ADA* EVALUATION PROJECT



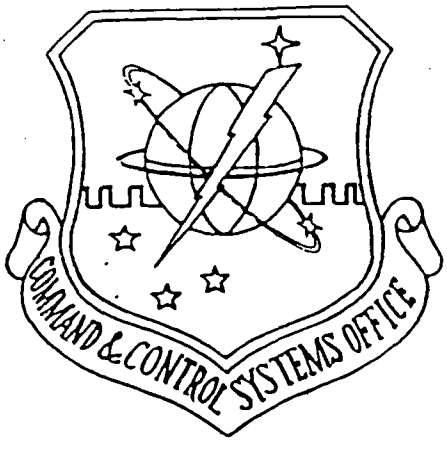
# ADA* DEVELOPMENT ENVIRONMENTS



Prepared for

HEADQUARTERS UNITED STATES AIR FORCE
Assistant Chief of Staff of Systems for Command, Control,
Communications, and Computers
Technology & Security Division






Prepared by
Standard Automated Remote to AUTODIN Host (SARAH) Branch
COMMAND AND CONTROL SYSTEMS OFFICE (CCSO)
Tinker Air Force Base
Oklahoma City, OK 73145 - 6340
COMMERCIAL (405) 734-2457 / 5152
AUTOVON 884 - 2457 / 5152



9 December 1986

**THIS REPORT IS THE FIFTH OF A SERIES WHICH DOCUMENT THE LESSONS LEARNED IN THE USE OF ADA IN A COMMUNICATIONS ENVIRONMENT.**

## ABSTRACT

This paper reports on the findings of a software development group on the state of two specific Ada software development environments.

The report first goes into a little background on the project and people involved in order to give the reader a perspective on the validity of the findings.

Next, the overall goals we had established for using an Ada software development environment are described. It was basically envisioned that a set of integrated tools would allow us to be more productive, produce a more reliable product, and easily, but effectively, control our software configuration.

The next section describes the process used to identify and select our environments. We started out to use the SofTech ALS system for the primary development environment, and the Alsys PC AT Ada environment for final testing and integration. Because of numerous problems with the ALS environment, we ended up almost exclusively using the IBM PC AT environment for all the work.

After problems and advantages for each of the environments are discussed, we then make some specific recommendations for other groups starting out with Ada software development.

Finally, the paper gives some sources for obtaining additional information on Ada software development tools and environments.

## Ada Evaluation Report Series by CCSO

| | |
|---|---|
| Ada Training | March 13, 1986 |
| Design Issues | May 21, 1986 |
| Security | May 23, 1986 |
| Micro Compilers | December 9, 1986 |
| Ada Environments | December 9, 1986 |
| Transportability | Winter 86-87 |
| Modifiability | Winter 86-87 |
| Runtime Execution | Winter 86-87 |
| Module Reuse | Spring 87 |
| Testing | Spring 87 |
| Project Management | Spring 87 |
| Summary | Fall 87 |

# TABLE OF CONTENTS


1. INTRODUCTION....1
   1.1. BACKGROUND....1
   1.2. PURPOSE....2
   1.3. ASSUMPTIONS AND CONSTRAINTS....2

2. OUR GOALS....4

3. CHOOSING AN ENVIRONMENT....5
   3.1. BACKGROUND....5
   3.2. TESTING AND INTEGRATION ENVIRONMENT DECISION....5
   3.3. THE APSE ENVIRONMENT DECISION....6
   3.4. OVERALL ENVIRONMENT PLAN....6

4. THE RESULTS....7
   4.1. THE ADA LANGUAGE SYSTEM....7
      4.1.1. ADVANTAGES....7
      4.1.2. DISADVANTAGES....8
      4.1.3. SERIOUS PROBLEMS....8
      4.1.4. ALS SUMMARY....9
   4.2. THE ALSYS PC AT ENVIRONMENT....10
      4.2.1. ADVANTAGES....10
      4.2.2. DISADVANTAGES....10
      4.2.3. SERIOUS PROBLEMS....11
      4.2.4. ALSYS SUMMARY....11
   4.3. OUR DEVELOPMENT ENVIRONMENT....11

5. RECOMMENDATIONS....13
   5.1. GENERAL, OVERALL RECOMMENDATIONS....13
   5.2. FIVE OR MORE PEOPLE....14
   5.3. RECOMMENDATIONS FOR SPECIFIC ENVIRONMENTS AND TOOLS..15
      5.3.1. OVERALL ENVIRONMENTS....15
         5.3.1.1. RATIONAL....15
         5.3.1.2. PC COMPATIBLE....16
      5.3.2. EDITORS....16
      5.3.3. PRODUCTIVITY TOOLS....17
      5.3.4. NETWORKS....17
      5.3.5. CONFIGURATION MANAGEMENT....18

6. FOR FURTHER INFORMATION....19


Appendices


A. REFERENCES....21

# 1. INTRODUCTION

This paper documents the experiences and work with several Ada environments on an Ada language project -- a "real world" project that has a tight schedule. If the reader wishes to read more on the background and theory of Ada environments, volumes of information are available. Some good sources of information are: The Department of Defense "Stoneman" document[1], the Software Engineering Institute paper on evaluation of Ada environments[2], or the STARS (Software Technology for Adaptable, Reliable Systems) Strategy document[3]. All of these documents should be available through the Ada Information Clearinghouse or the National Technical Information System (NTIS). The address is given in the "Further Information" Chapter.

## 1.1. BACKGROUND

The Standard Automated Remote to Automatic Digital Network (AUTODIN) Host (SARAH) project is a small to medium size project (approx. 40,000 lines of source code) which will function as a standard intelligent terminal for AUTODIN users and will be used to help eliminate punched cards and paper tape as transmit/receive media.[4] The development environment for SARAH consists of the SofTech Ada Language System (ALS) hosted on a Digital Equipment Corporation VAX 11/780, ALSYS Ada compilers for the IBM PC AT, and several IBM compatible PC XT and PC AT microcomputers. For some of the early Ada language training, a Burroughs XE550 Megaframe with a Telesoft compiler was employed.

Because the system must be reliable, maintainable, and reusable, it was decided to use modern software engineering concepts and methodologies to the greatest extent possible. The draft Software Volume of the Air Force Information Systems Architecture also recommends the use of Ada, software development environments, and formal software engineering methodologies in the development of Air Force software systems.[5]

The Department of Defense (DoD) language that best enabled implementation of software engineering concepts was Ada. What remained to be chosen was the design methodology and the development environment. The design methodology problem is discussed in a separate CCSO report.[6]

Many of the members of the SARAH software development team have had some experience or exposure to several different Ada compilers and software development environments. Some of these environments are:

- Telesoft/Unix: This was an early Telesoft Ada compiler

hosted on a Burroughs XE-550 Megaframe computer. There was no specific Ada environment for software support. The software development tool set consisted of the standard Unix System tools.

- ALS/VMS: We have used two versions of the Ada Language System (ALS) from SofTech. We had an evaluation copy of a very early release, and we now have the most current commercial release from SofTech (version 3). This environment has the recommended tools that fulfill the requirements specified in the Stoneman[1] document for a Minimum Ada Programming Support Environment (MAPSE).

- Verdix/Unix: One of our training courses provided by Intellimac used the Verdix Ada Development System (VADS) hosted on a Unix 68000 machine. Only the Unix program development tools were available (other than the compiler).

- Rational R1000: Several of our team members have seen brief demonstrations of the Rational Environment on the Rational R1000 hardware. This environment includes the integrated APSE type tools.

- Alsys/MSDOS: We are currently using (heavily) several IBM PC AT computer systems running PCDOS 3.1. These machines host the Alsys Ada compilers. The programming environment consists of the typical tools available in the MSDOS/PCDOS world.

## 1.2. PURPOSE

So that potential Ada developers could gain a practical insight into what was required to successfully develop Ada software, the Air Staff tasked the Command and Control Systems Office (CCSO) with evaluating and reporting on the Ada language while developing real-time digital communications software. The evaluation was to consist of a number of evaluation papers, one of which was to deal with Ada environments. CCSO chose the Standard Automated Remote to AUTODIN (Automatic Digital Network) Host (SARAH) project as the basis for this evaluation.

## 1.3. ASSUMPTIONS AND CONSTRAINTS

The assumptions and constraints are as follows:

- One possible constraint is the size of the SARAH project. Since the SARAH project team is small (10-13 persons), and since it will only be some 40,000 lines of code, some of the experiences reported in this paper

may not be appropriate for larger groups and larger projects.

- The SARAH team members have a variety of previous experience. Some members have had very little software experience. Others are very experienced in system design and development, and have a good working knowledge of different software development environments. Therefore, many of the observations presented come from different and varied perspectives.

- Since the SARAH project is at early coding phase of development, the final effectiveness of the concepts presented in this paper cannot be fully evaluated now. At the completion of the SARAH project, a summary paper will reflect on how well the selected environment worked out.

- SARAH is not embedded software.

## 2. OUR GOALS

We had two reasons for looking to use an Ada software development environment:

- Increase the productivity of development teams. A totally integrated software development environment provides sets of tools that allow designers and programmers to increase their productivity. Some of the tools one would expect to see are:
  - -- symbolic debuggers
  - -- program profilers/analyzers
  - -- complete configuration management/control
  - -- documentation generators
  - -- syntax oriented editors

  Ideally, these, and other, tools would work in concert with each other to allow programmers to produce more code with less time and effort.

- Improve software quality. Some of the tools mentioned above inherently improve the quality of software systems if they are used properly. For instance, profilers and analyzers can help programmers spot sections of code that are intensively used as potential targets for code optimization. Configuration management tools can help ensure that the software team knows exactly what software version and what changes are implemented in product baselines. Syntax oriented editors can help both productivity, and assist in producing standardized code.

# 3. CHOOSING AN ENVIRONMENT

## 3.1. BACKGROUND

At the beginning of the project we knew basically what we wanted. It was necessary to now decide what items we should attempt to obtain for the SARAH project. Since the final product would have to run on IBM PC AT compatible equipment, it would be necessary to have something that produced target code for that type equipment. At the time the initial product planning was being done, nothing was commercially available. We did know that several companies were working on IBM PC hosted compilers that would produce IBM PC code. It seemed the best approach was to buy a compiler hosted on an IBM PC or AT to use for the final testing and integration since this was the only possible product that would target to the PC.

Although there were several companies that anticipated releasing that type product, it appeared none of those products would be more than the basic compiler/linker. If we wanted to use an Ada Programming Support Environment (APSE), we would have to obtain another package. Since we had a Digital Equipment Corporation VAX available for use in the Ada project, it seemed most expedient to obtain something that would run on that machine. Our choices were several (Alsys, Verdix, SofTech, DEC, etc.). Because of other CCSO usage of the VAX that required the VMS operating system, we had to eliminate those systems that only ran under Unix (e.g. Verdix). We were constrained to a very small budget (under $25,000.00), so any product costing over that (e.g. the DEC Ada environment) would be eliminated. This was unfortunate since both the DEC Ada compiler and the Rational compiler were reported to be very fast, and of good quality.

## 3.2. TESTING AND INTEGRATION ENVIRONMENT DECISION

In the PC based Ada world, we had primarily been tracking a compiler from Alsys, and one from General Transformations. At the 1985 Boston SIGAda, several of our team members saw a prototype of the Alsys PC AT compiler running through the most current Ada validation test suite. From our observations, and verbal confirmations from Alsys, we decided that their product would likely fulfill our compiler needs for integration and final testing. The General Transformation compiler was reportedly still far from validation.

We started procurement action for several IBM PC AT compatible compilers. At the time procurement action was taken, Alsys was the only vendor providing a validated product.

## 3.3. THE APSE ENVIRONMENT DECISION

We had a pre-release copy of the Army/SofTech Ada Language System (ALS) for evaluation purposes. It was obviously not a production quality product. There were many bugs, we had no training, and the environment was not intuitively easy to learn. We therefore had no solid experience for judging the value of the programming environment.

From the literature available, and some limited contact with vendors via SIGAda, we knew of only three environments that seemed to offer an integrated approach to the Ada programming environment:

1) Rational Environment supplied with the Rational R1000 hardware,

2) SofTech ALS hosted on the VAX, and

3) The DEC Ada environment hosted on the VAX.

Although the early ALS system did not seem to be of production quality, we were assured by Army personnel, and SofTech people, and literature that the newer releases of ALS were much faster, more complete, and of production quality. The Rational environment was very new at the time. Although we had heard some good things about it via the "grapevine", we had no solid basis to judge it's merit. The same was true of the DEC Ada environment.

The primary consideration would then be cost. The commercial version of the ALS was in the range of $6,000.00 for government customers. The R1000 environment was in the $800,000.00 range, and the DEC Ada was around $50,000.00. Because of the budget contraints, we never were able to seriously consider the DEC or Rational Environments.

## 3.4. OVERALL ENVIRONMENT PLAN

The ALS environment was planned to be the focal point of our Ada Programming Support Environment. The source code developed on the microcomputer workstations would be maintained by the ALS configuration control system and would be transferred to the PC ATs for final compilation and testing. The SARAH software targets are the IBM compatible PC AT and PC XT microcomputers.

This approach would have the disadvantage of having a lot of file transfers between the PCs and the VAX. The big advantage would be the integrated APSE type environment for the majority of the software development work. Since we anticipated the speed of the AT based compilers would be much slower than that of the ALS hosted on the VAX we didn't anticipate using the PCs for much except text editing and the final integration and testing.

# 4. THE RESULTS

## 4.1. THE ADA LANGUAGE SYSTEM

Our ALS system was delivered and installed by SofTech Corp. This section will describe our actual experiences with the two systems described in the previous section. We contracted for and received nine days of SofTech training on the ALS system utilizing our DEC VAX 11/780 computer with four terminals.

### 4.1.1. ADVANTAGES

The SofTech ALS has attempted to implement much of the Ada environment thinking proposed by the Ada community, and specifically by the Stoneman document.

As such, it certainly has many good points:

- The configuration management and control functions could be extremely valuable to software projects -- especially larger projects. The environment gives the impression that you know exactly where everything is, who can use it, where it came from, etc. In general, we found this to be true.
- Access control for code can greatly enhance the manager's control of the developmental or baselined environments. Some of the highlights of file control are:
    - Files have attributes used to identify and control access. The ability to change attributes is controllable.
    - Files can be shared. It is possible to link across Ada libraries ("acquiring containers" in ALS terms). This prevents having to have multiple copies floating around in the system.
    - Locks and keys to files can be used to prevent more than one user from modifying the same file at the same time.
    - Files can be "frozen" by the configuration manager to prevent changes from being made.
    - ALS provides revision sets to track changes made to the file.

- The system resides in a multi-tasking environment. It is possible to start a compile in the background, and edit another program concurrently.

- By having a centralized system, programmers can examine their peer's code for ideas, suggestions, confirmation of interfaces, etc. without being able to change that code.

#### 4.1.2. DISADVANTAGES

The implementation of the ideas proposed by Stoneman did not appear to be well done. We encountered deficiencies that detracted from the potential of the ALS:

- The system is SLOW. Creating a new Ada library can take 30 minutes. Loading the debugger with a program to debug takes around 15 minutes (which is forever when a programmer is trying to think through a problem and debug a program).

- The ALS is hard to use (at least compared to today's very user-friendly programs). To be able to have the "nice" features in the environment (like keeping track of program versions, protecting baseline versions, etc.), the ALS is buried in its own Environmental Data Base. It seemed that everything we did had to have some kind of special attribute or file or directory structure, and we had to have some kind of special procedure or code or instruction to use it. After two weeks of classes, many people felt that they could not sit down and enter an Ada program, compile, and run it without extensive reference to the manuals.

#### 4.1.3. SERIOUS PROBLEMS

With all software, we must usually endure some minor problems. Usually, these are soon identified, and people learn to live with them, or work around them. This is only true if the system is overall stable and predictable. With the ALS, we found this predictablility and stability lacking.

During our ALS training class, while talking about the roll-in and roll-out functions, the instructor stated that it wasn't reliable -- it only worked some of the time. When we got to the section on the debugger, we asked to see a demonstration using one of our class problems that would not run. The debugger, after taking nearly 30 minutes to load and get ready to run with

the subject program, promptly "died" upon execution.

We had many instances where commands and tools just "didn't work". Sometimes calls by the instructor to SofTech Headquarters would bring an explanation (usually "Oh, that tool has some problems -- we don't recommend you use it"), and sometimes we never found out what happened. We had to reboot the VAX numerous times during the class.

By the time the classes were complete, the entire group was in agreement that the overall ALS system was still in a pre-production state. It was not a debugged or complete product that should be sold or released. The unpredictability of the system cannot be tolerated in an environment that must manage an extremely expensive resource -- software.

#### 4.1.4. ALS SUMMARY

Overall the ALS does have many of the tools required for the activities normally associated with software development. The configuration management facility does seem to work. In general, the tools needed to write and test programs are present. However, considering the cost of the system to the government, and the time spent developing the system there are several notable things not present:

- An Ada browsing capability, and a syntax sensitive editor that would assist in the writing and overall checking of code before compilation.
- Debugging and program analysis capabilities that provide features comparable with commercial products available for other environments and languages.
- An editor with a multi-window capability. By today's standards, only being able to see and work with a single file at a time is very primitive. This feature alone can make a dramatic difference in programmer productivity.
- A consistent, integrated, easy to use human interface.

Our experiences lead us to believe that the ALS(version 3) has not reached a level of maturity, stability, or performance to recommend its use. Furthermore, we believe that it is not very likely to reach an acceptable level of maturity without major rework.

## 4.2. THE ALSYS PC AT ENVIRONMENT

We initially obtained two copies of the first release of the Alsys PC AT compiler, and two IBM PC AT computers. We were, in general, very pleased with the package. Some of the salient points were:

### 4.2.1. ADVANTAGES

- The speed of compilation was very good. The same program that would take 20 minutes to compile on our VAX 11/780 with the ALS would compile on the AT in less than two minutes. In all fairness, we should mention that later we added more memory to our VAX, and that brought the ALS compile times to within a few minutes of the AT times.

- The product was very stable. We encountered very few problems, bugs, or "strange occurrences" with the compiler. It was a solid and predictable product.

- The compiler was very easy to learn and use. There is an on-line help facility that enabled us to successfully use the compiler without first reading the users guide. The compiler environment syntax is Ada-like.

- With the second release of the compiler, we were able to use some of the common productivity tools available for the PC environment (e.g. SideKick, SuperKey, etc.)

### 4.2.2. DISADVANTAGES

Although the section above reads very positively, there are some definite drawbacks with using the Alsys PC AT environment:

- Keeping up with programs and development work is the sole responsibility of the programmer. There are no configuration management tools provided with the Alsys package. For example, there is nothing to prohibit one programmer from writing to another programmer's area, other than management policy.

- There is no APSE-like environment at all, save for that specified by the Ada Language Reference Manual. If tools are needed other than those directly related to the compiler, they must be obtained and "integrated" by the end user.

- The compiler does not come with a debugger. This is a necessity for serious software development. (We

understand that a debugger is being developed -- but it will cost extra!).

- Alsys has several predefined packages that make programming easier on the IBM PC compatibles. However, the source code is not included! This amounts to an extension of the Ada language. Users should be wary of this arrangement.

### 4.2.3. SERIOUS PROBLEMS

Alsys is charging a royalty to distribute systems compiled and linked (binding) with their Ada runtime executive system. In the world of microcomputers, the concept of a compiler vendor charging royalties on the runtime environment linked to the generated code is archaic, and a concept that is, indeed, ugly to many computer professionals. Many of them tend to view this concept as greed. Indeed, when the people involved in the SARAH project (many with considerable experience in the private sector) learned that Alsys was extracting a royalty for all copies (except for the first 10) of a developed system, the general thinking changed from "what a great product - from a great company" to that of "as soon as another product is available, we will move to that - even if it is not as good otherwise".

### 4.2.4. ALSYS SUMMARY

Overall, we found the compiler a very good product. It is usable and speedy. It can form the core of a program development environment on IBM PC AT compatible workstations.

The compiler is, however, only a compiler. It does not in any way provide a complete, integrated program development environment. As we will explain below, this is not a total disaster since many of the existing products available for the PC can be used together to create a reasonable development environment.

## 4.3. OUR DEVELOPMENT ENVIRONMENT

Because of the problems discussed above with the ALS, we have had to, for the most part, abandon our original plans for using it. We had still, even in the end, hoped to use it for at least the configuration management function. This would necessitate up and downloading source code from the PCs. We found that without high speed local area networking, this process is very time consuming and awkward. There is no Air Force requirements contract or equivalent for acquiring local area networks. With this fact, and the marginal possible usefulness of the ALS, we have not been

able to devote much time to further use of the ALS. The fact of the matter is: If we want to finish the SARAH project in a timely manner, we must focus on the reliable, solid Alsys compiler to get the job done.

# 5. RECOMMENDATIONS

We would like to note that the recommendations given are based on our experience. Your needs, background, or funding posture may dictate different solutions.

We have some different groups of recommendations. Some of them are based on the size of the software development team, and the software development project. The different classes for our recommendations are:

- General, overall recommendations, including small programming projects.
- The project with five or more people.
- Recommendations for specific environments and tools.

## 5.1. GENERAL, OVERALL RECOMMENDATIONS

These recommendation are applicable to all development efforts regardless of the environment used, compiler used, or project size.

- Based on our experience in developing several thousand (so far) lines of Ada code, we would strongly recommend that each designer and programmer have their own personal workstation. The increased productivity seen would easily pay for the cost of the hardware many times over every year.
- Each workstation should have (or have access to) an Ada compiler. Nothing can slow down a programmer's momentum like being ready to test some new code, and have to "get in line" to compile and test it. We have found that both designers and programmers tend to get very frustrated when they are on a productive "roll" and the hardware and resources are not available to allow them to move forward with their ideas.
- Each workstation should have (or have access to) a symbolic debugger that works in conjunction with the Ada compiler.
- For the very small project, it is possible to keep track of source code development configurations, and baselines manually. For example, a pseudo-shared environment can be obtained by keeping current copies of programs on a shared (manually) hard disk. A possible additional item of value for this small development group would be a single user version of a configuration management (source code control) system to be available to at least the project manager. This

manual type environment is not our recommendation of ideal even for only a few programmers. However, it is probably the minimum that is workable.

## 5.2. FIVE OR MORE PEOPLE

For a project of this size, significant returns can come from having a shared environment. The environment should have the following characteristics:

- Networking or sharing of some type should be employed. Packages and programs must be available from each workstation. Walking around trading diskettes would be unacceptable in an environment of this complexity. As explained below in the section on Overall Environments, there are several ways to achieve the sharing.
- There should be a number of tools available to the users of each workstation. Some of the tools need to be shared, and some of them could be dedicated (resident on the workstation). Our definition of shared for the purpose of this section is a program that resides and runs on a central CPU (file server or multiuser computer).
    - Shared:
        - Database/Configuration Management type tools. Configuration Management tools are discussed in the "further information" section.
        - Electronic Mail, or equivalent. Smooth flowing communications can greatly enhance productivity. Generating paper is not efficient.
        - Ability to share (or at least easily transfer) source code files with other programmers.
    - Dedicated. It is not necessary to have dedicated versions of these tools. However, if it is more appropriate (e.g. memory resident utilities on a PC) they may be dedicated.
        - An editor, or combination of editors that allows viewing and manipulating of more than one file at a time. An Ada oriented editor that assists with writing and debugging syntax would be an additional benefit.

- Other Ada specific tools such as tools for program analysis, program debugging, program documenting, etc.

## 5.3. RECOMMENDATIONS FOR SPECIFIC ENVIRONMENTS AND TOOLS

In order to provide more information than supplied in the general recommendations, we have attempted to provide some specifics that may be of assistance.

### 5.3.1. OVERALL ENVIRONMENTS

As mentioned above, we found the SofTech ALS to not be a good choice for serious software development. This is not at all to say the idea of an integrated APSE does not work.

The two environments that we now see as reasonable options from our direct experience are the Rational environment, and the IBM PC compatible environment. Many of the other vendors environments may well be suitable (DEC, Data General, Harris, etc.). We do not have the experience to comment on them.

#### 5.3.1.1. RATIONAL

Another APSE type environment that seems to be quite successful is the Rational Environment hosted on the Rational R1000 processor. We have seen several short demonstrations of this system, and have been very impressed.

The entry price is now down to about $300,000.00 to get started with this environment. The compiler is very fast. In fact, this environment uses true incremental compilation to greatly reduce the compilation overhead when code changes are made. The environment includes a complete set of tools that include:

- Interactive syntactic and semantic analysis of programs or program fragments. Work can be checked while editing.
- The environment, for example, can create private part and package body skeletons. This kind of automated assistance can greatly reduce the number of programmer keystrokes.
- Multi-window editor.

- Interactive program cross reference ability to allow instant reference to variable declarations, etc.

From what we have seen, this product series may be very close to the type integrated Ada programming environment that was envisioned by the authors of the Stoneman document.

Any group doing large scale projects in Ada, or any programming house with many people using Ada should certainly seriously investigate this environment. The Rational address is in the "Further Information" section.

#### 5.3.1.2. PC COMPATIBLE

With the price of IBM PC AT compatible computers falling monthly, an organization must certainly consider the economic advantage of using PCs to enter the Ada world.

A number of AT compatible machines, AT hosted compilers (right now, Alsys would be our recommendation -- for more information, see the CCSO paper on microcomputer compilers), a small amount of PC compatible software, and a local area network can result in an inexpensive, yet reasonably effective Ada development environment.

The remainder of the "tools specific" recommendations are aimed at making this PC based environment as productive as possible.

### 5.3.2. EDITORS

There are many, many word processors and text editors available for IBM PC compatible machines. Most of them will satisfactorily work for producing Ada code.

When our project started out, the most used software was WordStar. However, most of the programmers have transitioned to using the memory resident editor supplied with SideKick. This allows them to "pop up" the source code while running the executable version of the same program. It also allows viewing two source code files at the same time. In the SARAH programming area, it is quite common to find a screen with a program opened with Wordstar, and then another file overlayed in SideKick, so that the programmer can work on both files at the same time.

In the "we'd sure like to try it" category, there is at least one Ada oriented editor available for the IBM compatible world -- the Xinotech Program Composer. If it is well done, this would be an even greater tool for added productivity. The full reference can be found in the "Further Information" chapter.

### 5.3.3. PRODUCTIVITY TOOLS

In the PC environment, we have found that some of the commonly used productivity tools can be well employed in the Ada programming environment.

SIDEKICK
As mentioned, the memory resident editor in Sidekick is used heavily since it (and the file loaded in it) are always "there" -- a keystroke and it's on the screen instantly. Some of the other functions in Sidekick are also very valuable. Probably the next most used item is the ASCII table. It's so efficient to not have to look for that sheet of paper to look up ASCII values, keystrokes, etc. Although those are the only two components that are heavily used to program Ada code, the other features such as the Calculator (it does binary and Hex too!) and Appointment Calender are used by some of the people to optimize their time utilization.

SUPERKEY
SuperKey is sold as a keyboard macro utility. Essentially, one can define any key to "contain" letters, words, phrases, commands, etc.

This allows programming shops to set up a macro that will define certain keystrokes as Ada constructs. For example, an Alt-P could be set up to contain the skeleton for a procedure. An Alt-F could contain the shell for an Ada function. Then the programmer need only type Alt-P and a complete procedure shell is entered into the current text editor just as if the programmer had typed it.

The program can even be set up to insert some text, pause and wait for programmer input, then continue with more "canned" text. It is also easy to load a "transient" variable name or phrase into a key as it is typed. Then only that key need be pressed every time the programmer would normally have to type the often occurring name or phrase. This tool can take a lot of the drudgery out of the business of writing code.

### 5.3.4. NETWORKS

There are a number of PC compatible local area networks available. From our contacts, it appears 3Comm and Novell seem to be the most mentioned as far as popularity and sales.

With this type network, many things are possible:

- Many program development tools are available in versions that can work for multiple users on such a network. The source code control system mentioned in the "Further Information" chapter is one such tool that would be of great value.

- Many very powerful and easy to use general purpose tools will work in this environment. For instance, many valuable tools can be quickly built with a relational database management system like dBASE III Plus from Ashton-Tate.

- These local area networks typically provide Electronic Mail services, which can really save time by eliminating all the paper floating about the program development shop.

- Programmers can have the power of individual workstations with individual compilers and Ada oriented tools, yet they can still share files, and access common databases, etc. Everyone has the ability to share, but no one has to worry about the degradation of compiler and system performance as the user load increases.

### 5.3.5. CONFIGURATION MANAGEMENT

There are several software packages available to assist with the configuration management function for software development projects in the IBM PC compatible environment. Since software development in this environment typically involves many different languages, these packages are not language specific, and will work fine for Ada. *The Polytron Version Control System* listed in the "Further Information" section below is one such system.

## 6. FOR FURTHER INFORMATION

PC BASED CONFIGURATION MANAGEMENT AND CONTROL
The Polytron Version Control System is a configuration management and control system designed for the IBM PC environment. It is capable of managing both source code and documentation. It tracks current version plus all increments, decrements, and notes needed to rebuild any version. It has privilege levels to control access. The system administrator can easily keep track of who is working on each module. Local area network versions are available. More information can be obtained from:

Polytron Corporation
1815 NW 169th Pl Suite 2110
Beaverton, OR 97006
(800)654-5301

ADA INFORMATION
The first and probably best contact point for obtaining most types of information related to the Ada language is the Ada Information Clearinghouse, sponsored by the Department of Defense.

Ada Information Clearinghouse
Rm. 3D139 (Fern St./C-107)
The Pentagon
Washington, D.C. 20301-3081
(703)685-1477 or (301)731-8894

RATIONAL ENVIRONMENT
This multiuser APSE type environment provides a set of well integrated tools with fast hardware for Ada program development.

Rational
1501 Salado Drive
Mountain View, CA 94043
(415)940-4770

ALSYS IBM PC AT COMPATIBLE ADA COMPILER
The Alsys compiler is a good solid product. If used with some IBM PC compatible general purpose programming tools, a reasonable, yet inexpensive Ada programming environment can be created.

Alsys Incorporated
1432 Main Street
Waltham, MA 02154
(617)890-0030

ADA ORIENTED EDITOR
The Xinotech Program Composer is an editor that has knowledge of Ada syntax so that it can assist in writing and debugging code.

Xinotech Research, Inc.
Technology Center, Suite 213
1313 5th Street S.E.
Minneapolis, MN 55414
(612)379-3844

SIDEKICK AND SUPERKEY
These are two very popular and much used general purpose aids to productivity. At prices usually under $50, they are one of the best software bargains available.

Borland International
4585 Scotts Valley Drive
Scotts Valley, CA 95066

LOCAL AREA NETWORKS
The two networks that seem to be selling the best in the IBM PC compatible market are the Novell network, and 3Comm.

Novell Corporate Communications
748 North 1340 West
Orem, Utah 84057
(801)226-8202

3Comm
(800)NET-3Com

## A. REFERENCES

[1] "Stoneman", Ada Joint Program Office, National Technical Information Service (NTIS).

[2] "Evaluation of Ada Environments", Software Engineering Institute, Carnegie Mellon University, August 1986.

[3] "Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy", Department of Defense, ACM SIGSOFT Software Engineering Notes, April, 1983.

[4] "SARAH Operational Concept Document", Command and Control Systems Office, US Air Force, September 5, 1986.

[5] "Draft Air Force Information Systems Architecture -- Volume VII ", HQ US Air Force/SITI, 18 February 1985.

[6] "An Architectural Approach to Developing Ada Software Systems", Command and Control Systems Office, US Air Force, May 1986.

GENERAL

"Proceedings of the Air Force Information Systems Architecture Workshop", Air Force/SITI, August 1984.

"Air Force Information Systems Architecture, Vol I -- Overview", Headquarters Air Force/SI, May 1985.