DANIEL F. STERNE

APPLIED RESEARCH IN ADA

A three-phase program consisting of applied research into the Ada* programming language, Ada-based design techniques, and the Ada Programming Support Environment has been developed at APL. The first phase established a foundation of initial Ada laboratory facilities, hands-on Ada experience, and basic literacy in Ada technical issues. The second phase is under way and is investigating the application of Ada to representative Navy shipboard systems. The third phase will develop spinoff products such as prototype reusable Navy software components, an advanced Ada training course, or new Ada Programming Support Environment tools to support development and maintenance of Navy Ada software.

BACKGROUND

What is Ada? Ada is a new state-of-the-art computer programming language developed by the Department of Defense for embedded computer systems.[1] But more importantly, Ada is also a modern approach to reducing software life-cycle costs. The approach encompasses a design philosophy that departs from the traditional emphasis on computational efficiency alone and recognizes that software must be designed to reduce software maintenance costs. In addition, the approach calls for the use of an integrated set of special utility programs to facilitate the designing, coding, testing, maintaining, and managing of software. This set of tools, called an Ada Programming Support Environment, is being created in conjunction with the development of compilers (translators) for the Ada programming language.

The Department of Defense developed Ada to become the single standard high-order language for the Department's embedded computer systems, and the armed services have embarked on plans to adopt Ada. Nevertheless, the transition to Ada involves a number of technical and management issues. The Navy has made substantial commitments to existing software standards and has millions of lines of code invested in such older languages as CMS-2. Thus, for the Navy, a transition to Ada entails not only technical risks but also a major shift in software policy and resource utilization.

The Fleet Systems Department of APL has been monitoring and participating in Ada developments since early 1981. During 1981, an in-house Ada document library was established, and an informal newsletter was issued to increase the scope of Ada awareness within APL. The first useful Ada translator, developed by New York University, was installed on computer facilities managed by the Advanced Systems Design Group and made generally available. Members of the Group joined the national Ada special-interest group sponsored by the Association for Computing Machinery and became participants in the Ada Programming Support Environment standards committee, known as the Kernel APSE Interface Team. In late 1981, the Group's growing interest in and knowledge of Ada culminated in a plan for a three-phase program of applied research.

*Ada is a registered trademark of the U.S. Government (Ada Joint Program Office).

PROGRAM OBJECTIVES AND PROGRESS

Phase 1

The objectives of the first phase were to gain hands-on programming experience and to acquire familiarity with the central technical issues surrounding Ada and the Ada Programming Support Environment. This phase was carried out during 1982 and 1983 and was supported by the Independent Research and Development Program at APL. The first task was to acquire an Ada program development system capable of supporting larger programming experiments than could be carried out using the New York University translator. A survey of Ada compilers was undertaken. Based on the survey, a Telesoft compiler and an Intellimac computer (Fig. 1) were purchased.

Next, a well-understood example application, suitable for redesign in Ada, was needed. An existing simulation of a distributed processing system was chosen for its value as an educational vehicle. Although the simulator is not a real-time tactical program, it consists of many components that execute in parallel and addresses many of the central issues of real-time tactical programs. These include synchronization, buffering, resource sharing, deadlock prevention, and error recovery. The simulator was completely redesigned and programmed in Ada using the Telesoft/Intellimac system. Every attempt was made to exploit Ada's most significant new features.

The resulting 2500-line program was analyzed and compared with the earlier version, which had been programmed using the Pascal language and the Digital

Figure 1—Telesoft/Intellimac computer system used in Ada programming experiments.

Equipment Corp. RSX-11M executive system. The Ada version of the simulator was judged more reliable, more transportable, simpler, and superior in overall organization. These improvements were attributed to the use of Ada's advanced features: packages, tasks, and exception handlers. The team was favorably impressed with Ada but felt that its large size and the complexity of certain features would make comprehensive programmer training relatively difficult. The language features for intertask communications were found to be powerful and easy to use but may be less natural or less efficient than other mechanisms for some applications. Ada's large variety of program building blocks allows designers to construct elaborate program architectures whose structures are difficult to envision from program listings. Consequently, some form of pictorial representation is needed as supplementary design documentation. A combination of Ada-oriented graphic notations proposed in the literature[2,3] was found useful. Figure 2 illustrates the use of this notation to depict one component of the simulator.

The remaining element of Phase 1 was a literature survey that concentrated on unresolved technical issues considered of interest to the Navy. These included the use of Ada as a program design language, use of Ada in distributed and multiprocessor systems, the suitability of various computer architectures for executing Ada programs, Ada educational approaches, and the compatibility of existing Navy software with Ada.

Phase 2

The objectives of Phase 2 are to explore the issues of applying Ada to Navy tactical software systems and provide lessons learned for future combat system upgrades. Phase 2 encompasses two efforts, one funded by the Aegis Complementary Research and Development Program and the other by the Tomahawk Cruise Missile Program. Each began in late 1983 and builds on the base of experience acquired in Phase 1.

In the Aegis investigation, a simplified model of a large tactical computer program is being redesigned in a top-down manner. The Aegis Command and Decision System Computer Program was chosen as an example from among six candidate tactical systems. The selection was based on a number of factors including representativeness, familiarity, availability of documentation, and potential for illuminating particular Ada issues. The Command and Decision Program is one of three computer programs forming the core of the Aegis Weapons System. It is responsible for coordinating the activities of the Aegis sensor and weapons subsystems in response to operator commands and automated doctrine rules. The objective of redesigning the program is not to produce operational tactical software but to identify problems and successful techniques of applying Ada to a typical shipboard system.

The investigation probes the aspects of program architecture in which Ada-based designs will differ most from traditional ones. These aspects include scheduling of functions in response to elapsed time or input/output events, the exchange or sharing of data among functions, error detection and error handling, and the encapsulation of processor and device-dependent information. Various system-level Ada design techniques discovered during Phase 1 are being evaluated.

The redesign of the Command and Decision Program is proceeding in three steps, or "cuts." During each cut, a simplified executable model of the program is built that is completely redesigned, based on the requirements described in the Command and Decision System Program Performance Specification document. Each model is of wider scope and higher fideli-

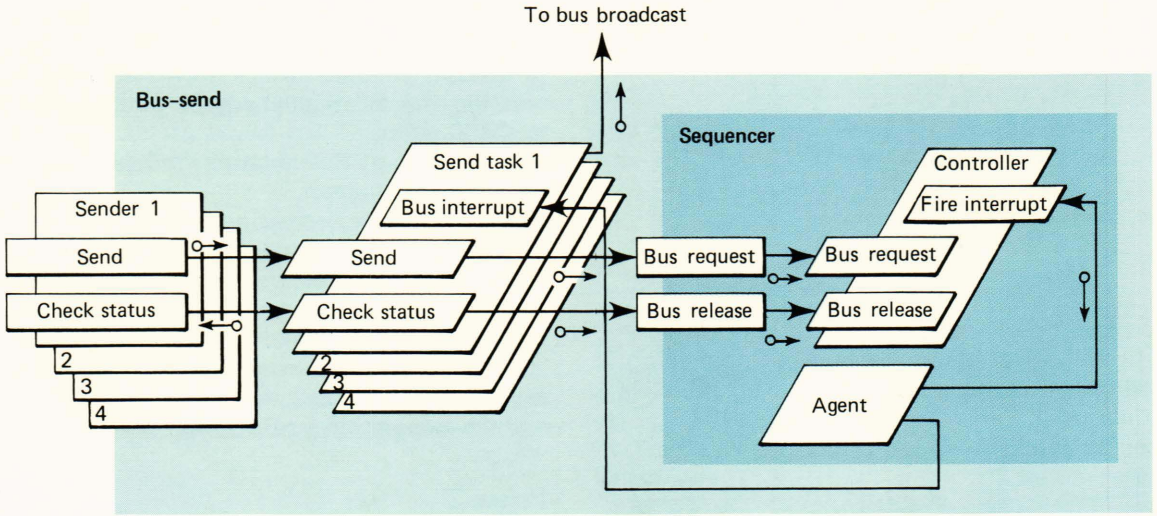

Figure 2—An example of an experimental Ada-based graphic notation, used here to describe a component of a simulator program. Rectangles and parallelograms represent Ada code modules: packages, procedures, tasks, and entries. Large arrows represent execution flow paths; small arrows represent data flow paths.

ty than its predecessor. At the beginning of each cut, only those design decisions and techniques that worked successfully in the preceding cut are retained; others are discarded. Similarly, choices among software development computers, compilers, execution computers, and peripheral devices are reexamined and new choices are made.

The three cuts are expected to take approximately 3, 9, and 30 months, respectively. The first cut has recently been completed. It uses a Digital Equipment Corp. VAX-11/780 computer and the UNIX operating system for both program development and program execution, and an Ada subset compiler developed at the University of York, England. A Navy standard UYA-4 console is the program's primary input/output device.

The first cut model implements only a few essential, highly simplified command and decision functions. The functions include managing a small database of track reports from a single simulated sensor; displaying track positions, velocities, and identification symbols on the UYA-4; and responding to operator commands to engage targets, drop tracks, and change track identifiers. Subsequent models will support multiple operators, additional operator functions, track reports from multiple sensors, doctrine-driven automatic system response, and increased detail in all implemented functions. Each cut will also include a wraparound simulation program to create input stimuli and record output responses. As shown in Fig. 3, the program provides simulated communications from the Aegis Weapon Control System and Spy Radar Control System.

The other Phase 2 Ada activities are sponsored by the Tomahawk Cruise Missile Program, which is considering using Ada for future upgrades of the Tomahawk Weapon Control System software. As the Technical Direction Agent to the Tomahawk program, APL plans to support future upgrades by conducting

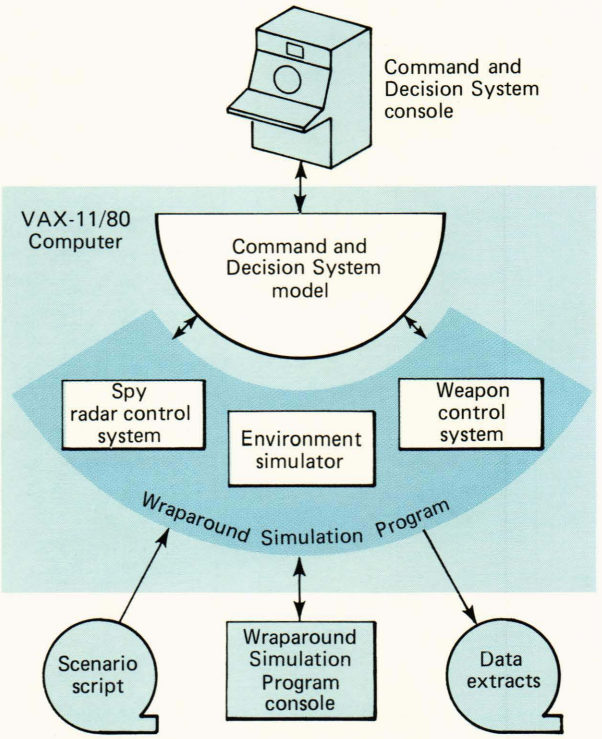

Figure 3—Configuration of the software used in Phase 2 of the initial Ada investigation. The software includes a simplified model of the Aegis Command and Decision System and a wraparound simulation program to create input stimuli and record output responses. The wraparound simulation program provides simulated communications with the Aegis Weapon Control System and Spy Radar Control System and creates a simulated environment of friendly and hostile forces.

software design experiments for the weapon control system. General areas of interest include experiments to improve the man/machine interface and to incorporate Ada's design philosophy into the overall software structure by using Ada as a program design

language. To date, APL has assisted the Tomahawk program in identifying Ada risks and fallbacks and has participated in an Ada advisory committee for the Tomahawk Weapon System.

In addition, the Tomahawk and Aegis programs have supported planning efforts to establish an APL Ada laboratory facility and a series of Ada awareness and education seminars.

Phase 3

By capitalizing on the insights gained in the Phase 2 experiments, it should be possible to develop some specific spinoff products. Several potential spinoffs have already been identified. One is derived from what is perhaps Ada's greatest potential benefit: its support for constructing reusable software components. Truly reusable components will allow new software applications to be built from elements of earlier applications, resulting in substantial cost savings. If Ada proves successful in this respect, the Navy will eventually assemble a library of reusable components specialized for Navy applications. By generalizing and enhancing some of the Ada components developed in the Phase 2 experiments, APL may be able to contribute prototype reusable components to a Navy Ada library.

Another potential spinoff concerns the need for more extensive and specialized Ada training. In particular, real-time system architects, having years of experience using traditional techniques and philosophy, may be ill-prepared to use a radically different technology like Ada to design large tactical programs. Although consultants and educational institutions offer numerous courses on Ada and on software engineering with Ada, these courses tend to be introductory, directed specifically at programmers, and concerned only with small, academic example applications. At present, there are no educational courses that address the needs of experienced real-time system architects. Another potential spinoff is an advanced educational course using the Phase 2 prototype systems as case studies. Because these prototypes are based on representative Navy applications, an analysis of their strengths and weaknesses may have significant pedagogical value.

Another potential spinoff concerns the need for specialized Ada Programming Support Environment tools to support development of Navy tactical software. The experience of building representative tactical systems may lead to the identification or development of such tools.

REFERENCES

[1] *Reference Manual for the Ada Programming Language,* U.S. Department of Defense, ANSI/MIL-STD-1815A-1983 (1983).

[2] R. Buhr, "A Graphical Design Notation for Ada with Realistic Examples," *ACM AdaTEC Meeting,* San Diego (24-25 Feb 1983).

[3] G. Booch, *Software Engineering with Ada,* Benjamin Cummings Pub. Co., Menlo Park, Calif., pp. 49-52 (1983).

ACKNOWLEDGMENT—The author wishes to thank the following for their contributions to the results described in this article: M. E. Schmid, R. A. R. Pearce, T. A. Grobicki, and M. J. Gralia.