SUPPORT TECHNOLOGIES FOR AVIONICS OPERATIONAL FLIGHT PROGRAMS

ROBERT L. HARRIS

WRIGHT LABORATORY
AVIONICS DIRECTORATE
WRIGHT-PATTERSON AFB OH 45433-6543

ABSTRACT

As software grows in airborne weapon systems support problems of operational flight programs (OFPs) become a critical technology challenge. Five important technologies are discussed from the viewpoint of how they might be applied to controlling the "software tail" of OFPs. They are: 1) Ada; 2) Hypermedia; 3) Automatic Programming; 4) Software engineering environments; and 5) Advanced Verification and Validation.

THE SOFTWARE EXPLOSION

The amount of software in airborne weapon systems has been growing exponentially for the past three decades and this growth is expected to continue at an even accelerated rate for the foreseeable future. Among the reasons for this phenomenon are the pervasive applications of digital technology, innovations in computer science and engineering resulting in the migrations of former hardware functions to software, and the ease of implementing more complex and sophisticated capabilities in software. The transition of hardware functions to software is illustrated in Figure 1. Prior to the digital revolution, avionics were, for

the most part, implemented by dedicated, standalone hardware units with very little software. Digital technology today is promoting major transfers of avionics' functions from hardware to software, a trend that is expected to continue into the foreseeable future. Figures 2 and 3, taken from a paper delivered at the 1990 Air Force Software Technology Support Center's annual conference dramatically reflect this phenomenon by its effect on the growth of software in Air Force weapon systems. (1)

WEAPON SYSTEM DEPENDENCY THE GROWTH OF SOFTWARE

Figure 2. Avionics functions performed in software

TRANSITION OF AVIONICS HARDWARE TO SOFTWARE

Figure 1. Hardware is migrating to software.

Figure 3. Growth in demand.

OPERATIONAL FLIGHT PROGRAMS

The growth of software-intensive avionics has had a profound effect on Operational Flight which resides in avionics systems and performs functions integral to that system. In addition to becoming more abundant, this software is increasing in its complexity and pervasiveness. The support of OFPs is done at a facility known as an Avionics Integration Support Facility (AIFS). A typical AIFS makes changes to OFPs including: additions and/or changes in system capability; deletions, additions, and/or changes in operational modes; changes in operational functions;

changes and/or replacement of system equipment; correction of errors; reliability/maintainability improvements; and testing, validation and documentation of all changes. Tools to accomplish these and other tasks are shown on Figure 4. (2) The OFPs themselves represent only a small portion of the total computer memory, hence information and data listings; avionics development equipment software; test equipment software; AIFS tools; automatic test equipment software; and software engineering environment tools. Figure 5, adopted from Reference 1, is illustrative. If one uses as the measure of software memory, a personal computer's "floppy disk," the average OFP can reside in less than the space of one-third of a floppy disk (64Kbytes). Multiple OFPs would fit on a single floppy...750kbytes of memory. It is necessary to add to this, supporting software descriptions and listings that total approximately 45 floppy disks. Next is the required software for OFP development

equipment plus its documentation, which would typically total 94 floppies. This would be followed by test equipment and its documentation plus workstation support software totaling around 250 floppies. Finally it is essential to add AIFS software, and automatic test equipment, plus its documentation, that together average approximately 900

Figure 5. The "Software Tail" of Operational Flight Programs

floppies. This brings the collective total to more than 1,290 floppies for a modest avionics suite that is deployed today. If one extrapolates this to future systems that include integrated avionics, fault-tolerant behavior, distributed processing, and advanced system architectures, it is not unreasonable to arrive at several orders of magnitude increase. This realization carries the imperative for technologies to focus its attention on software logistics in order to support these systems once they are deployed.

SOFTWARE SUPPORT ACTIVITIES REPRESENTATIVE TASK BREAKOUTS

Figure 6. Support data compiled by AFLC (Ref. 4)

SOFTWARE SUPPORT

According to Glass and Noiseux (3), software support can be divided into three categories: corrective, adaptive, and perfective. Corrective support is the correction of software errors, and typically accounts for only 17 percent of total software support activity. Adaptive support, the act of changing software to conform to environmental changes (e.g., new hardware, etc.) accounts for about 18 percent, and perfective support, the improvement of software because of new system requirements, accounts for over 60 percent of the total support effort. Based on data compiled from various Air Force OFP support activities, a typical distribution of these tasks is shown in Figure 6. (4)

The Air Force Systems Command, Wright Laboratory, Avionics Directorate, Avionics Logistics Branch (AAAF), at Wright-Patterson AFB has focused technology on improving the supportability of OFPs. The OFP support process generally follows that shown in Figure 7 which has been adapted from Reference 2. This process is very similar to the one

Figure 7. Typical "maintenance" of Operational Flight Programs.

used during weapon systems conceptual and acquisition-development phases. Because of the processes similarity, software-support technologies developed in the laboratory can apply not only to systems currently in their conceptual stage, but also to existing systems that are routinely updated via software block changes. Among the classes of technologies being developed by AAAF are: the Ada language; hyper-media; automatic software generation; software engineering environments; and software testing and validation.

ADA TECHNOLOGY

The Ada language was developed and introduced by the Department of Defense as its single, preferred higher order programming language (HOL), in an effort to mitigate the

rising costs and difficulty of sustaining software-intensive weapon systems. Air Force policy has recently reinforced Ada as the only approved HOL for new weapon system development unless technical reasons dictate otherwise. The application of Ada to OFPs cannot be separated from Ada compiler performance. Ada compilers are suitable currently for applications that run on general-purpose computers with no severe memory or time-critical performance constraints. OFPs however reside in memory-constrained and time-critical environments. Each OFP application therefore must evaluate the quality and maturity of its compiler based on compile-time efficiency, object-code efficiency, additional compiler services, and support for real-time embedded system requirements. The Avionics Directorate's Ada Compiler Evaluation Capability (ACEC) program, funded by the Ada Joint Program Office, developed the technology of evaluating Ada compilers. The ACEC consists of test benchmarks that automatically produce more than 1,200 performance-tests of Ada compiler-target machine combinations. In addition, the ACEC provides an evaluation tool to interpret test results. The performance-test programs provide three capabilities: 1) performance comparisons of multiple Ada compilers and target-computers using a uniform set of measurements that are tailored for a particular application; 2) determination of changes that have been made between different versions of compilers; and 3) performance prediction of alternate coding styles. It provides measures of execution-time efficiency, code size efficiency, and compile-time efficiency. ACEC's evaluation tools provide, in addition, a set of tools and procedures that assist in preparing the test suite for compilation; in extracting results from the data of executing the test suite; and in interpreting and analyzing the performance measures obtained. This capability has been transitioned from the laboratory to industry for general use. It is available from the Air Force Data Analysis Center for Software, Rome Laboratory/COED, Griffiss AFB, NY. (5), (6), (7), (8)

HYPERMEDIA

Hypermedia is a contraction of the words "hyper-text" and "multimedia." Its technology combines text, audio, and video information with a computer for greater presentation efficiency and much greater user effectiveness in managing large amounts of changing data. Hypermedia "threads" non-linear or non-sequential text along with other media. Information is "chunked" into small units called nodes and connected by links, thereby enabling users to navigate through the information by selection of these links. The technology is driven by avionics-software-maintainers' need for rapid and comprehensive access to documentation and information of all types, plus a simple capability to manage it. Under pressure to carry out changes quickly, maintainers often form only a local and partial understanding of software, which often leads to incorrect and inefficient modifications. Hypermedia technology hopes to

provide simple, natural styles of information access, including no "key words/phrases" to remember, or queries to compose. Users would control the level of detailed presented at any time by "point-and-click" interactions with the computer screen. Maturation of this technology is aimed at decreasing information/knowledge access time; and its initial focus is documentation updates, technical order maintenance, and reduction of software distribution times. A technology assessment is underway that will establish technical requirements and specifications for a hypermedia system for OFPs.

The introduction of each new generation of fighter has resulted in an order of magnitude increase in the amount of documentation. To maintain each block version of contemporary fire control computer software, for example, support personnel must rely not only of the source code listings, but the Computer Software Product Specification, Interface Control Document, and Prime Item Development Specification. Each of these documents contains hundreds of pages and reference numerous other documents. As new software is developed for next-generation aircraft or upgraded for currently fielded systems, it is required to perform many more functions. Therefore, the amount and need for more documentation increases. Currently, the documentation required for delivery to the Air Force is explained in DOD-STD-2167A. This standard identifies 22 deliverable products that could result from a software development. With the Air Force faced with supporting weapon system software amid such a large amount of documentation, the need arises for a capability for quickly accessing and easily managing enormous quantities of information. Hypermedia technology is one solution to this challenge.

AUTOMATIC PROGRAMMING TECHNOLOGY

Automatic software-generation technology has a potential to treat the escalating amount of software required for OFPs, by increasing productivity of OFP support personnel; and it holds a promise of reducing weapon system block-cycle update time. As new system requirements emerge and as OFPs become larger and more complex, the "software-tail" associated with their maintenance must be handled by some form of automation of the code-generation and documentation process. The Avionics Directorate's major program for developing technologies in this areas is the "Automatic Programming Technologies for Avionics Software" (APTAS) effort. (9)

The focus of APTAS is to synthesize real-time avionics application software, focusing initially on radar tracking systems. APTAS is to provide high-level specifications and design capabilities from which executable Ada code can be synthesized automatically. Users of this system are eventually intended to be avionics systems and software engineers. The APTAS knowledge base will contain precise, codified definitions of radar tracking terms, data structures, functions, and synthesis rules to relieve an avionics engineer

from having to specify such. The engineer will consequently be able to synthesize code directly from specifications expressed in a high-level avionics specification language. Four APTAS requirements have evolved. First, it must be easier for system engineers to use the automatic programming system than to write code directly. This means using modern human-machine interaction principles. The system must shift the burden from what the user enters to produce code, to reusable application and design knowledge, mediated by inference on the part of the system. This infers use of an expressive language to simplify input for the user. General specification languages have had mathematical orientations and have typically been difficult to learn and use for most people. This will be one of the major challenges of the program. Second, APTAS will be domain-specific...avionics. Since the design of OFPs is specialized, considerable specialized knowledge is needed in order to understand specifications and to produce valid executable designs that satisfy the specifications. This knowledge comes from many areas and ranges from very general to very specific. This requires that APTAS be extensible rather than domain-independent. Third, APTAS will be interactive rather than completely automatic in its synthesis. Users will interact with the synthesis process and intermediate steps in the process will be accessible and meaningful to users. Allowances will be made for the user to override design decisions made by the system. Fourth, APTAS will be designed to result in quality Ada code. Its end code may be required to be optimized for either readability or efficiency. APTAS' design will not preclude either of these dimensions.

SOFTWARE ENGINEERING ENVIRONMENTS

The Advanced Multi-Purpose Support Environment (AMPSE) is an AAF dynamic test simulator used to test, modify, verify, and validate OFPs. It uses distributed processing with multiple, plug-in card computers that communicate over a high-speed, real-time network. The hardware and software design use a modular, building block approach that has many advantages over traditional architectures. While advantages such as: acquisition and support cost savings, reduced size, improved reliability, and greater flexibility are each significant, perhaps its most significant advantage is expandability. The AMPSE architectural concept allows computers to be added incrementally as additional processing power is required. This permits new technologies to replace old technologies gradually, thus avoiding the need for expensive upgrade efforts. (10), (11). The importance of this technology to future OFP support cannot be overstated.

ADVANCED AVIONICS VERIFICATION AND VALIDATION (AAV&V)

Advanced verification and validation technology addresses four problem question areas. First, how does one verify and validate that mission-critical software will function correctly in a multiple embedded processor environment that is distributed and fault-tolerant? Second, how does one test such software? For example, how does one know when to stop testing, and how can one establish a level of confidence associated with any given amount and type of testing for advanced systems? AAF's AAV&V effort will attempt to answer these questions. Its objective is to enhance software verification and validation testing techniques and methodologies via formal statistical analyses for real-time software. The effort aims at developing a capability to quantify software quality and reliability, and to enable software engineers to predict how reliability (e.g. the probability of failure-free operation for a specified time) will vary as testing progresses.

Testing is concerned with both verification and validation of OFPs. Verification is intended to demonstrate that a program meets its specifications while validation is intended to demonstrate that a program does what the user wants it to do. (12) Many textbooks on software engineering agree that testing can only demonstrate the presence of errors. It cannot prove their absence. The testing process involves unit testing, module testing, subsystem testing, integration testing, and acceptance testing. Testing and debugging are distinct. The aim of testing is to discover the presence of faults in a program while debugging is the process of locating erroneous code and repairing it. Real-time systems are difficult to test because of timing dependencies between processes in the system. System errors can be time dependent, only arising when the system processes are in a particular state. The exact state of the system processes when the error occurred may be impossible to reproduce.

SUMMARY

This paper has reviewed five technology areas that could significantly improve the supportability of operational flight programs. Each area is significant in terms of managing the proliferation of software that is occurring in avionics systems. As future avionics becomes even more complex and software-intensive, these areas will assume a critical importance.

REFERENCES

- (1) Bates, Wayne, "Introduction to the Software Technology Support Center (STSC)." A talk presented at the Joint HQ USAF/SC-STSC Software Technology Conference, Salt Lake City, UT. to the plenary session on 24 April 1990
- (2) Patterson, Alton, "Air Force Integration Support Facilities: Their Total Utility," IEEE/AESS 1980 NATIONAL AEROSPACE ELECTRONICS CONFERENCE PROCEEDINGS, (NAECON80), p. 126.
- (3) Glass, Robert L., and Noiseux, Ronald A., Software Maintenance Guidebook, Prentice-Hall Co, 1983, Englewood Cliffs, NJ.
- (4) Ferens, Daniel V., Defense System Software Project Management, 31 January 1990, Air Force Institute of Technology, School of Systems and Logistics, Wright-Patterson AFB, OH., p. 16-2.
- (5) Pitarys, Marc J., "Avionics Laboratory Ada Compiler Evaluation," IEEE/AESS 1990 National Aerospace Electronics Conference Proceedings, (NAECON90), 22-26 May 1989, Dayton OH, p.610.
- (6) "New DACS Product: Ada Compiler Evaluation Capability, Software Product," DACS Newsletter, September 1968, p. 2, Data and Analysis Center for Software, Rome Laboratory/COED, Griffiss AFB, NY 13441-5700.
- (7) Weiderman, Nelson H., Ada Adoption Handbook: Compiler Evaluation and Selection, Version 1.0, Software Engineering Institute report no. CMU/SEI-89-TR-13, ESD-TR-89-21, March 1989, pp. 79, 95.
- (8) Leavitt, Tom; Terrell, Kermit; and Decker-Lindsey, Barbara; "Ada Compiler Evaluation Capability (ACEC): An Overview," AIAA COMPUTERS IN AEROSPACE VII CONFERENCE PROCEEDINGS, October 3-5, 1989, Monterey CA.
- (9) Williamson, James S., Jensen, Paul S., et. al., "Automatic Programming Technologies for Avionics Software," IEEE, AIAA, NASA 9th Digital Avionics Conference Proceedings, October 15-18, 1990, Virginia Beach VA.
- (10) Stephenson, Mark, "An Advanced Multi-Purpose Support Environment," IEEE/AESS 1989 National Aerospace Electronics Conference Proceedings, May 1987, Dayton OH, p. 1282.
- (11) Howell, Gary T., and McCord, James W., "Support Environment Methodologies for the Rapid Reprogramming of Operational Flight Programs," Air Force Systems Readiness Technology Conference Proceedings, San Antonio TX, 1990.
- (12) Sommerville, Ian, Software Engineering, 3rd ed., Addison-Wesley Publishing Co., Reading MA, 1989., p. 432.