LABORATORY CONCEPTS IN AVIONICS SOFTWARE

BY

ROBERT L. HARRIS

WRIGHT RESEARCH AND DEVELOPMENT CENTER
AVIONICS LABORATORY
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-6543

ABSTRACT

The growth of digital electronics has spawned dependence on software as the enabler of performance in avionics. As such, laboratory software concepts have technical leverage on deployed, developmental, and conceptual systems. Seven concepts are the focus of software technology at the Air Force Avionics Laboratory: 1) automatic programming, 2) software fault tolerance, 3) reusable software, 4) expert code modification, 5) common Ada run-time systems, and 6) modular embedded software for distributed systems. Each is discussed within the context of avionics.

INTRODUCTION

This paper addresses concepts in the avionics software domain for tactical military aircraft. As digital avionics have grown, mission performance and reliability have become totally dependent upon the evolving science of software engineering. Software dependency has been fostered by the growth of computers and extensive use of embedded processors. Research within the Air Force Avionics Laboratory involves concepts that are necessary for solving problems associated with what has been identified as "The Software Crisis." These concepts are critical to knowing the limits of avionics operational capabilities, costs, and mission effectiveness in tactical weapon systems.

THE AVIONICS SOFTWARE DOMAIN

Avionics software has three characteristics that distinguish it. First, it is real-time-embedded. It must simultaneously satisfy timing-deadlines and computational accuracy. Typically avionics have processor context-switching time requirements of 30 to 100 microseconds. Its embedded characteristic results in constraints on the weight, size, cooling, packaging, and environmental specifications of the computer resources that are employed. Second, this software typically lasts a long time and undergoes many changes during its long life. Software updates and enhancements occur with the aircraft as new operational requirements emerge, as new capabilities are added, and as obsolete functions are dropped. It is not unusual for such software to last more than 20 years, with as much as 80 percent of the code replaced by the end of the period. Third, avionics software is part of a system with a "man-in-the-loop." Consequently, user-interfaces, documentation for training, technical orders, and version-control are necessary as this software ages.

SOME KEY AVIONICS SOFTWARE CONCEPTS

With the growth of computers, software is playing the enabling role for advanced system architectures, avionics functional integration, and use of shared resources. Six concepts are the focus of much software work at Wright-Patterson Air Force Base's Avionics Laboratory: 1) automatic programming; 2) fault-tolerant software; 3) software reuse; 4) expert code modification; 5) common Ada run-time systems; and 6) modular embedded computer software. These concepts are applied to three classes of systems. First are current aircraft in the inventory. Typically the designs of these aircraft are fixed and retrofitting constitutes the major task. Second are aircraft in their development stage. More flexibility in technology-insertion and software creativity is permitted and involved. These systems have not yet gone through Program Management Responsibility Transfer to the maintenance community. Third is the conceptual system representing the future where neither hardware nor software technology is constrained. Software is characterized in this situation as being unprecedented each of these system-classes software has a leverage effect on operation, performance, costs, and reliability. Laboratory research addresses the three life-phases of avionics software: the development
phase, the operational (use) phase, and the post-development (maintenance) phase.

AUTOMATIC PROGRAMMING

The goals of automatic programming are: 1) increased productivity of software development by removing much of the mechanical toil and drudgery of coding and documentation; 2) increased software reliability by capturing reusable parts that have been previously tested; 3) reduced cost of software production; 4) reduced time for software creation and change; 5) improved software version-control and documentation; 6) promotion of coding standards and maintenance efficiency; and 7) improved rapid turnaround for system changes. The Laboratory's instantiation of this concept is the Automatic Programming Technology for Avionics Software (APTAS) program. (1)

A prototype automated programming environment for development of Ada avionics software is being developed. APTAS is to provide high-level specifications and design capabilities from which executable Ada code can be synthesized automatically. Users of this system will be avionics systems and software engineers. With the APTAS system, engineers will be able to quickly compare various architectural designs, along with hardware and software trade-offs, by testing instrumented, operational prototypes. A prime goal of the program is improvement of avionics' quality and reduction of development costs and time. In order to accomplish this, automatic programming techniques, rapid prototyping, avionics design methodologies, productivity tools, knowledge-based avionics, acquisition requirements and analysis concepts are being applied. The APTAS effort has the following design requirements: a) an avionics system description language; b) automated design synthesis capability; c) reusable avionics design templates; d) abstract graphical design methodology, and e) ease-of-use. It should be noted that a completely automatic and domain-independent automatic programming system is generally considered not possible. APTAS uses rules to successively transform a specification into code and incorporates backtracking to resolve design conflicts. A major objective is to demonstrate that significantly-improved Ada programmer productivity can be attained. The conceptual approach combines graphical programming, automatic code generation, incremental analysis and evaluation of code, common data base management and software reuse, with a programming environment that supports programming-in-the-large.

SOFTWARE REUSABILITY

No concept has had more attention paid to it, yet been more difficult to implement inter-organizationally than software reuse. Most organizations employ some form of software reuse internally. However when attempts have been made to institutionalize reuse across organizations the advantages of reuse have been lost in non-technical issues. Among the problems associated with reuse are the following: (2)

a. We do not know the critical component attributes that make a component reusable are. For example designs, speci-
ficiations, algorithms, and software configuration items, as well as code modules could constitute reusable parts. Assessing reusability of components across different application domains is difficult.

b. Developing generalized components is more expensive than developing a component for a specific purpose; and how does one assess return-on-investment for the extra costs?

c. Many software engineers feel they can beat the costs, time, and performance of "generic" software with their own which is optimized.

d. There is no agreed upon means of classifying, cataloging and retrieving software components. The critical attribute is that component retrieval costs should be less than component development costs.

The goal of the Avionics Laboratory's Reusable Ada Avionics Software Packages (RAASP) effort is to develop a flexible, user-friendly software reuse system that can be repeatedly used for Ada-based avionics applications. This effort will exploit "lessons learned" from the CAMP program and provide a sound approach to software reuse in the avionics domain (radar, electronic warfare, navigation, identification, communication, fire-control, and flight controls). Explicit tasks include: a) domain analyses for evaluation of difficulties, risks, cost effectiveness, and functions involved in implementing reuse concepts in these domains, and selection of candidates for reuse; b) identification, design, test, implementation, and documentation of RAASP life-cycle objects (i.e., designs, algorithms, test cases, code, etc.) for use in the selected avionics domain; c) provision of a RAASP reuse library that is user-friendly, efficient, flexible, and organized for easy access and retrieval using a optimum classification schema; d) development of an automated configuration management system to track utilization and population of the library, and e) use of expert-system library managers and component reuse metrics, to develop a prototype avionics reuse system for demonstration. The RAASP system proof-of-concept will be applied to a radar system. The metrics will address confidence and reliability factors for potential users of reusable components, and a management perspective regarding the amount of reuse and associated productivity gains.

EXPERT AVIONICS CODE MODIFICATION

The Expert Avionics Code Modification (EACM) system is to provide the maintenance programmer with project-specific, maintenance-related data, and with a suite of tools that perform maintenance functions. The goals of EACM are to reduce the time and other resources needed for Ada avionics program maintenance, and to reduce the quantity of resources needed to modify Ada avionics. This is being done by using knowledge-based systems and knowledge acquisition techniques applied to individual avionics projects. Knowledge-systems' techniques are used to build in EACM a "corporate memory" of the development and maintenance processes. Four aspects of EACM are of particular note. First, EACM learns about individual projects by "doing"...the longer it is used on a project, the more it comes to know about it, providing an automated "team member" that remembers how and why things were done. Second, EACM is built around a "society of agents" architecture, minimizing the difficulty of extension, adaptation, porting, and even of multi-host operation. Third, EACM has a flexible hypertext interface, allowing the user maximum freedom in controlling the assertiveness and level of detail of the program, and making access to vital program information extremely flexible and responsive. Finally, EACM embodies considerable knowledge about Ada and avionics.Expert Avionics Code Modification (EACM), has the objective of reducing time and money required for the maintenance of Ada avionics software.

COMMON ADA RUN-TIME SYSTEM (CARTS)

In any avionic software effort, the Ada run-time system is an essential component for project. Each Ada compiler uses a run-time system for tasking, scheduling, reconfiguration, memory-management, system-initialization, and other executive functions. Currently, run-time systems are specific to each compiler and provide limited support for avionic software. Implementing a custom run-time system for each compiler inhibits avionic software reuse and portability. In addition, the Air Force will soon be faced with supporting and revalidating numerous Ada compilers thru the life of many weapon systems. Extensive testing and knowledge is required for each run-time system.

CARTS is aimed at developing a common Ada run-time system for embedded, real-time applications. The CARTS will provide common interfaces, features, and options to avionic software developers and compiler implementors. This will be accomplished by building upon the considerable work done by the Association for Computing Machinery's Ada Run-Time Environment Working Group (ARTEWG). This group of over 38 companies and universities have already developed a catalogue of interface features and options for the Ada run-time environment, in addition to a model run-time system interface for Ada. The CARTS project will implement a common run-time system for two compilers targeted to different processors. This will prove
that the CARTS is transportable. Furthermore, with the CARTS built and demonstrated with avionics software, performance can be proven. With these goals met, in addition to the nonproprietary nature of the software, it is anticipated that the CARTS will be widely accepted by the Air Force and compiler users and builders. Without it avionics software reuse and transportability are severely limited. It is expected that CARTS will save time and money during software development on systems that use several processors and compilers by eliminating the need for understanding, modifying, and testing more than one run-time system. Furthermore, software support cost savings will be realized because one nonproprietary run-time system will have to be maintained when compilers are updated or changed.

MODULAR EMBEDDED COMPUTER SOFTWARE (MECS)

This concept is an attempt to create an integrated design information and representation scheme and support tools for distributed, fault-tolerant, multiprocessor avionics. Entitled Modular Embedded Computer Software for Advanced Avionics Systems (MECS), it will provide quantitative measures of the impact of system modifications, along with standard consistency checks. The approach will develop a system-level design methodology that enhances supportability. As a result of research into design information representation technology, operating systems, and reusability-approaches for distributed, fault-tolerant, multiprocessor avionics architectures, there was created a technology concept for an Avionics Software Information Representation Scheme and the associated development environment. An integration support environment is being designed that can span multiple facets of development/maintenance activities and treat the different perspectives of the avionics system held by different types of users. (3)

CONCLUSIONS

Digital avionics and the migration of traditional hardware functions to software are increasing the importance of software logistics in the research laboratory. Several software concepts in the forefront of research for avionics systems have been reviewed in this paper. These concepts apply to three phases of avionics' system life: conceptual, developmental, and operational. Breakthroughs in any one of these concepts will result in an order of magnitude improvement in the operations, performance, and maintenance of avionics in tactical weapon systems.

REFERENCES

