SOFTWARE TECHNOLOGY FOR
NEXT-GENERATION STRIKE FIGHTER AVIONICS

Major Lynne Hamilton-Jones
United States Air Force
Joint Strike Fighter Program

Kenneth Littlejohn
Marc Pitarys
United States Air Force
Wright Laboratory

ABSTRACT

The mission of the Joint Strike Fighter (JSF) Program is "to facilitate evolution of fully developed and validated operational requirements, proven operational concepts, and mature demonstrated technologies to support successful development and production of next-generation strike weapon systems for the U.S. Air Force (USAF), Navy (USN), Marine Corps (USMC), and our allies".[1] Commensurate with its mission, the JSF Program endeavors to reduce the cost of developing strike fighter software by maturing and demonstrating software technology, standards, and processes. Several critical technology areas have been identified. The areas are Programming Languages, Secure Real-Time Operating Systems, Software Engineering Environments, and Software Architectures. In each of these areas, technologies have been identified to accomplish the overall goal of affordable software for next-generation strike fighter avionics. During FY95, the JSF Program sponsored demonstrations as part of the JSF Risk Reduction Studies/Demonstrations (RRSD) phase. Throughout this phase, emphasis has been placed on leveraging accomplishments made from several sources including the Defense Advanced Research Projects Agency (DARPA), and USAF and USN scientific and technical efforts. The corresponding technology demonstrations featured Ada 95, Portable Operating System Interface (POSIX) standards, software fault-tolerance, and several technologies demonstrated under DARPA-sponsored Domain Specific Software Architectures (DSSA) program. This paper describes the accomplishments made to date by the JSF Program, and includes future plans.

INTRODUCTION

The mission of the Joint Strike Fighter (JSF) program is to "facilitate evolution of fully developed and validated operational requirements, proven operational concepts, and mature demonstrated technologies to support successful development and production of next-generation strike weapon systems for the U.S. Air Force, Navy, Marine Corps, and our allies." In support of this mission, an affordable avionics "building block" of matured technologies will be developed and demonstrated for transition into the Engineering and Manufacturing Development (EMD) phase for the next-generation strike weapon system. The maturation of the technologies depend heavily on the early formation of the JSF avionics...
architecture. The architecture, the "building codes" of the avionics, describe all aspects for the resulting avionics system, as well as the system software and its environment. The achievements made in demonstrating software technology to date, are the focus of this paper.

**JSF SOFTWARE DEMONSTRATIONS**

During FY95, the JSF Program sponsored technology demonstrations as part of the JSF Risk Reduction Studies/Demonstrations (RRSD) phase. The demonstrations focused on major risk areas that would impact the availability of affordable technology by the JSF EMD phase. The software technology demonstrations featured Ada 95, POSIX, software fault-tolerant techniques, and several technologies demonstrated under the DARPA-sponsored DSSA program. The results of the technology demonstrations are discussed in the following sections.

**Programming Languages**

The JSF has identified Ada 95 as its preferred programming language for avionics software. Ada 95 offers several improvements over its predecessor, Ada. Ada 95 has introduced several new core features for the development of distributed, fault-tolerant, multiprocessor software applications. How these features will impact affordable software development and meet the needs of next-generation strike fighter software are of primary concern to the JSF Program. To this end, the "Distributed Ada 95 Strike Fighter Simulation" demonstration was developed.
The goal of the Distributed Ada 95 demonstration was to demonstrate features of Ada 95 that support distributed, multiprocessor, fault-tolerant software. The simulation illustrates Ada 95 features from the “Special Needs” Annexes including system programming, real-time systems, distributed systems, safety and security, and numerics. The simulation features Weapons Delivery capability, a graphical Avionics Controls and Displays, and an Out-the-Window display (developed under separate science and technology efforts), and an analog joystick and throttle control.

The simulation software, developed exclusively in Ada, is hosted on a distributed platform of single board computers (SBCs) in a VME chassis. The software was ported to a distributed platform and modified to incorporate Ada 95 Remote Procedures Calls (RPCs), dynamic reconfiguration for fault-tolerance, and parallelism. Support for fault-tolerant multiprocessor techniques is supplied by a mechanism called “Virtual Nodes”.

A virtual node (VN) is an atomic unit of distribution for application software. The VN functions in the same manner as the Ada 95 “partition”. The VN executes independently of other software, unless it is communicating with other virtual nodes. Also included in the VN concept is support for remote procedure calls (RPCs), which are also defined in Ada 95. Partitions communicate only via RPCs, and both partitions and VN support synchronous and asynchronous calls.

The results of the Distributed Ada 95 Simulation demonstrated the benefits of Ada 95 for JSF avionics software. First, Ada 95 supports fault-tolerance through dynamic reconfiguration. Second, Ada 95 can perform in real-time. The simulation software executed on the distributed target according to the same timing constraints as the original software. Therefore, the lines of code which were added to the simulation software did not adversely affect the execution of the software. Finally, and most important, Ada 95 saves money, in terms of software development and integration. The original simulation software contained over 15K lines of source code. The task of porting the software to the new target involved approximately 100 lines of code being either modified and/or added, and less than 80 labor hours being expended. Thus, minimal code added means less time and cost to develop. Ada 95 also provides explicit support for object-oriented programming, programming in the large, and real-time/parallel programming.

Secure Real-Time Operating System

The JSF Program must identify technology to control the execution of avionics software and management of mission essential data. This control will be provided primarily by the avionics operating system (OS). The operating system maintains a partition between the hardware and the software that minimizes disruptions when one partition undergoes a modification. The JSF operating system will provide an interface to the avionics application software, commonly known as an application program interface (API). The JSF is leveraging commercial technology in order to meet this
requirement. The leading candidate is the IEEE POSIX 1003.n family of standards, where n represents a particular POSIX standard. The POSIX standards define a set of common interfaces to operating systems. The interfaces allow application software to be easily ported to different architectures. In addition, reuse of tests, designs, specifications, and knowledge is simplified by using standard operating system interfaces.

Although the POSIX standards are not specifically designed for real-time avionics software, several features are useful for next-generation strike fighter software. To this end, the JSF sought to assess the applicability of POSIX features for JSF avionics software. The Ada/POSIX Maturation (APM) program focused on assessing the performance, security, real-time data communication, and control mechanisms of the selected POSIX features, and comparing the results with an existing implementation. The POSIX features demonstrated were semaphores, timers, real-time signals, and message queues. For each of the features demonstrated, information was collected on timing, executable size, complexity (in terms of source lines of code), security, and implementation difficulty. The Ada Operating System (AOS), developed by Hughes Aircraft Company for the F-22, was modified to support the POSIX features. The AOS is currently used on the F-18 and F-22 weapon systems.

The preliminary results, in general, showed that the POSIX features produced acceptable service times such that they can be used in an avionics system. However, several general findings for using POSIX services in a real-time avionics system are noted below:

- POSIX features tend to occupy more memory than their AOS counterparts
- POSIX features lack the security features needed in a multi-level secure environment
- POSIX real-time signals introduce long latencies for asynchronous notifications
- POSIX needs a real-time distributed interprocess communication (IPC) interfaces.

Additionally, a number of concerns related to POSIX were highlighted. These include:

- Ada vendors support of POSIX bindings
- Completion of POSIX standards
- Ada 95 Special Needs Annexes obviate the need for a POSIX Ada API

These issues need to be resolved before POSIX is recommended for use in JSF avionics software.

Software Architecture

The JSF software architecture must have the ability to compensate for system and software errors to achieve mission effectiveness. Furthermore, this capability must be provided at low cost. Fault-tolerant techniques enable the software to continue to operate in the presence of software errors and failures caused by either hardware or software faults. Several fault-tolerant techniques have been effective in ensuring mission completion. The most common of these techniques are N-Version Programming
(NVP) and Recovery Blocks (RB). However, a new affordable fault-tolerant technique has been developed. The technique is known as the Data Fusion Integrity Process (DFIP).

The DFIP is an embedded algorithm whose function is to check for the reasonableness of data into or out of a software application unit. The DFIPs can modify the data if a discrepancy is found. The DFIPs are flexible in that they can be built based on simple rules, statistical analysis, or duplication of application rules. In addition, the DFIPs can be developed as a stand-alone module which can be easily added to existing software.

In order to exercise the DFIP capability, the Software Fault-Tolerant Demonstration (SFTD) system was developed. The SFTD system integrates the fault-tolerant modules with a typical strike fighter avionics application. A multi-media user interface was integrated with the application software to clearly and completely demonstrate the fault-tolerant technology as applied to avionics software. The demonstration system executes in one of four modes, each mode corresponding to the use, or non-use of, the DFIP. The modes correspond to execution of the algorithm under normal circumstances and in the presence of erroneous data. In the presence of erroneous data, the DFIP is used either on an algorithm input or output data item.

The scenario selected for the demonstration system was a simulated Air-to-Ground (AG) bombing mission. In particular, the Continuously Computed Release Point (CCRP) algorithm was used. The CCRP algorithm consists of approximately 5000 lines of Ada code. The algorithm computes the miss distance to the target. When the miss distance nears zero, the weapon is released. The time to weapon release is computed based on this computed miss distance and the aircraft dynamics. The time to release data is the data item used to trigger the bomb release.

The SFTD demonstrated that the DFIP technology is an affordable method of embedded software fault-tolerance, capable of improving mission effectiveness in the presence of faults generated by hardware or software. This capability is inexpensive to develop. On the average, a DFIP comprises approximately 100 lines of Ada code. Moreover, this technology is especially suitable for meeting the mission and affordability goals of the JSF avionics.

Software Engineering Environment

The JSF Program must define a software engineering environment which enables avionics software designers to rapidly specify, simulate, and develop avionics software. The environment must be supported by advanced software tools which enable reuse, domain specific software engineering, schedulability analysis, and performance analysis in order to reduce software costs.

Several technologies, developed under the DSSA program, were demonstrated as part of the Executable Architecture demonstration. The goal was to show how cycle time between design and implementation of software can be reduced by employing domain specific
software engineering, reuse, and automation to the software development process. The architecture is supported by a collection of tools that enable the developer to successively generate application-level Ada code modules from a generic, domain specific, functional specification. A separate collection of tools incorporate the Ada code modules into a generic system architecture to perform schedulability analysis, and are integrated to produce a real-time, executable image of the system architecture. The executable image is instrumented such that performance data can be collected during execution. The performance data can then be used to tailor the functional and system architecture specifications to meet overall design objectives.

The Executable Architecture provides several key benefits for developing affordable software for the next-generation strike fighter. With emphasis on automation and reuse, the Executable Architecture demonstrated technology which offers significant cost savings for software development. In addition, the ability to rapidly specify and simulate the software will yield a measure of confidence that the software satisfies cost and performance goals.

**SUMMARY**

The JSF Program endeavors to reduce the cost and risk factors associated with developing software for the next-generation strike weapon system. In an effort to realize these reductions, the JSF Program defined a software architecture that emphasizes such factors as modularity, open system standards, adaptability, reuse, and the leveraging of commercial technology. These factors were established by the JSF Program to identify viable candidate technologies for next-generation avionics. The JSF identified four risk areas associated with software. Technology demonstrations that proved the viability of the technology were developed. The next step is to mature these technologies for incorporation in the JSF Preferred Weapon System Concept (PWSC). For example, the maturity of Ada 95 compiler and Ada 95/POSIX bindings for the JSF operating system API will have to be ensured so that their use in the PWSC will be cost-effective. Their use, or non-use, could have a dramatic impact on the affordability of the JSF avionics.

**REFERENCES**


