Implementing COTS Open Systems Technology on AWACS

By Lt. Col. Michael K. J. Milligan
U.S. Air Force

The U.S. Airborne Warning and Control System (AWACS) E-3 weapons system required a modernization program for its aging mission computing system. Due to the significant technological and cost advantages of using commercially available hardware and software, a distributed, object-oriented, open systems, commercial off-the-shelf (COTS) approach was taken. This article presents lessons learned from the development and preproduction of the U.S. AWACS Step 1 Mission Computing Upgrade Program.

The purpose of this paper is to share experiences the AWACS development team encountered during the integration of COTS technology into the legacy E-3 weapon system. Sharing these lessons with similar programs may be helpful in avoiding some of the problems the AWACS team experienced.

The mission computing upgrade program was initially conceived in the Fall of 1995 as a result of the System Program Office's and Air Combat Command's desire to fix critical maintainability shortfalls and at the same time, get three critical operational capabilities aboard AWACS. These capabilities were required to enhance operational situational awareness, and include a more accurate tracker (fusing radar and IFF data), improved symbol definition, and more detailed (and useful) map displays.

Due to the high operations tempo and funding constraints, the acquisition approach used in the mission computing modernization effort injects technology in two fundamental steps, U.S. Step 1 and U.S. Step 2. Step 1 injects fundamental open systems technology, migrating the mission computing system from a vendor unique or closed legacy system to an open architecture and provides a path for future migration and growth. Step 2 completes the modernization effort. An open system implies that the system interfaces are public domain, so the selection and integration of components should be analogous to the concept of plug-and-play. Open systems provide cost savings by allowing a number of vendors to compete for the various hardware and software components in the broader commercial market. The AWACS architecture will no longer be tied to a specific vendor selling unique components built to proprietary or closed interface standards. By opening the architecture, future upgrades and new mission capabilities may be integrated with minimal integration and testing requirements.

The computer modernization development program was a joint effort among the AWACS System Program Office, Hanscom AFB; Air Combat Command/ 552 ACW and Air Logistics Center, Okla.; MITRE Corp., Lockheed-Martin Federal Systems, Boeing Space and Defense Systems, and GEC-Marconi Hazeltine. Many COTS vendors also participated.

In 1999, after approximately three years of development, the U.S. Step 1 production program was cancelled in favor of a larger, more comprehensive upgrade program. This program would continue developing the same technologies and COTS strategies as Step 1 while expanding the overall effort. There are many useful lessons learned during development and testing of the U.S. Step 1 program that can be applied to future modernization programs within the defense community.

COTS Considerations

With the introduction of COTS into the E-3, several aspects of traditional military weapon system design were modified or eliminated. Key areas include physical and environmental characteristics of the various COTS hardware components.

COTS hardware used in the U.S. Step 1 architecture is not specifically designed to operate in an airborne environment. In order to take full advantage of COTS, the design team needed to determine if certain components could be used. For example, the temperature range specified in the original AWACS system specification required that all electronics operate within the operating range of -55°C to +85°C (-67°F to +185°F). After flying the E-3 more than 20 years, Air Combat Command determined that such a wide operating temperature range was not necessary in most cases.

The requirement was modified to specify use in the 0°C to +50°C range (typical for most COTS electronics) and included changes to some existing operational procedures. This modification to environmental requirements provided the opportunity for use of an increased number of hardware components from various vendors. COTS hardware used in the U.S. Step 1 architecture includes single board computer cards, graphics accelerator cards, power supplies, cabinets, VERSA Module Eurocard (VME) enclosures, network interface cards, network switches, fiber optic cables, SCSI disk drives, 1553 I/O cards, and solid state memory devices. In addition, several COTS software components are used, including a real-time UNIX compliant operating system, map generation software, compilers, graphical user interface generators (X-Windows, for example), debuggers, and network interface software drivers. In addition to the many COTS components, some custom hardware and software was required to interface the U.S. Step 1 architecture to the remaining legacy system.

Legacy Software

In terms of life-cycle costs, software upgrades and maintenance are the most expensive component of the overall mission computing architecture. The current mission software consists of a single computer program called the Airborne Operational Computer Program (AOCP), which consists of approximately 370,000 lines of code written in Jovial and assembly language. The AOCP is responsible for all functions, including basic operating system services, timing and scheduling, and all applications, including weapons control, surveillance, display control, communications, internal simulation and system maintenance. Since the AOCP is based on a complex cyclic executive, any changes or upgrades made to the AOCP requires exhaustive functional and temporal testing to ensure

new functions operate correctly—logically and within specific timing constraints.

Mission Computing Hardware

The U.S. Step 1 Mission Computing Hardware Architecture is shown in Figure 1. The shaded areas indicate those components being added or modified. As illustrated, there will be a mix of new and legacy hardware. The new architecture is designed as a client-server network, distributing functions among a number of processing nodes. Each node consists of a processor, Local Area Network (LAN) Interface Cards (NIC), and other specialized cards. All are based on the industry standard VME bus design. The processor family of choice is PowerPC due to its performance, support of real-time operating systems, and large market share for embedded real-time applications. The LAN protocol chosen is switched Fibre Channel, based on bandwidth and real-time support requirements.

One of the key differences between the legacy system and the new U.S. Step 1 design is the use of a client/server architecture. Client/server is a relationship between processes running on separate machines (processors). The server process is a provider of services, while the client is a consumer of services. In essence, client/ server provides a clean separation of functions based on the idea of service [1]. The overall goal of this new architecture is to ensure the ability to provide inexpensive, timely upgrades and/or modifications to any hardware or software component without directly affecting the overall architecture.

U.S. Step 1 Software

Due to the many limitations and costs associated with development, maintenance, and testing legacy software, a modern software design consisting of a three-layered, object-oriented architecture was chosen for the U.S Step 1 Program. This new architecture is designed to allow migration of new software applications to a completely object-oriented design. The Distributed Software Infrastructure (DSI) is designed to isolate the application software from the operating system and allow encapsulation of all applications. This eliminates any application program dependencies (data or timing) on the operating system or specific hardware, and allows software to be developed and tested independently. This middleware is built according to open industry standards, ensuring that all present and future applications will communicate directly without any special, unique code changes. These open interface standards are called Application Programming Interfaces (APIs) and are based on the Object Database Management Group's (ODMG)'s Common Object Request Broker Architecture (CORBA) standards. By adhering to the ODMG's defined APIs, code developers can ensure their applications will interface correctly in any CORBA compliant environment.

The architecture supports this approach by being implemented as a collection of objects, and providing a framework in which objects can be shared among the distributed components of the AWACS computer system. For example, the display system application interacts with the tracker application by invoking well-defined methods on the tracker's interface. This hides the details of the complex tracking subsystem to the rest of the system. Most importantly, these interfaces are defined by an Interface Definition Language (IDL) that gets precompiled, enforcing the interface definition and allowing large amounts of automatic code generation. By communicating via these well-defined interfaces, all dependencies of the display on the tracker (and vice versa) are eliminated [2].

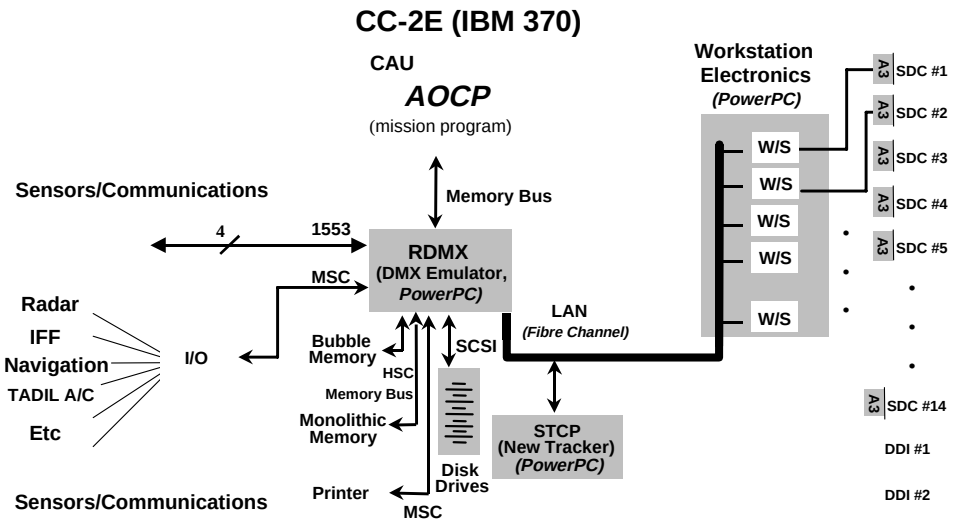
Figure 2 illustrates the software architecture. It is divided into three distinct layers: a real-time UNIX POSIX compliant commercial operating system,[1] software middleware (the DSI—information manager, encapsulated scheduler, and real-time database), and encapsulated applications layer. This design isolates all applications from the underlying operating system and any hardware dependencies, thereby ensuring platform independence for all applications. The DSI performs three vital functions:

1. It acts as communication pipeline, transferring objects between various component applications and database.
2. It schedules processes according to *a priori* priorities and pre-empts any lower priority processes if necessary.
3. It provides a real-time database.

A modified AOCP, with reduced functionality) continues to execute on the CC-2E (IBM 370) computer.

This software architecture is based on the philosophy of incorporating real-time design attributes into the architecture at all levels. The attributes of a real-time system may be characterized by the predictable response times; priority-based scheduling, and pre-emptive control. These three “P's” of real-time design allow the developer a great deal of control and flexibility, critical in designing today's complex real-time systems. Predictable response times under all load conditions ensure that applications respond to external events in a predictable fashion, regardless of what other tasks the system is handling. It requires consistent and prompt priority-based scheduling of time-critical tasks and low system overhead [3]. Pre-emptive control ensures deadlines are met by allowing lower priority process-

Figure 1. U.S. Step 1 Hardware Architecture (Functional Diagram)



es or threads to be pre-empted by high priority ones. The operating system, including the kernel, must also be able to be pre-empted. All three "Ps" are a function of the underlying operating system chosen to support the real-time applications [4].

Lessons Learned

There were many technical and programmatic lessons learned that can be derived from the U.S. Step 1 development program.

User buy-in to use of COTS is crucial.

While recognizing the implementation of COTS-based systems as an enabler to modernization, AWACS operational users are extremely cautious of COTS in their day-to-day operations because the hardware was not designed to harsh environments. To ensure program success, it was critical to have their full support of the program and participation on a regular basis to ensure operational requirements were understood and met. There were many opportunities during program development for the users to partner with the developers to devise a solution, and their commitment to a COTS-based solution was critical. Educating the user and support community also was important. Since the operational user is not normally in the business of developing technology, there were many opportunities for misunderstanding, especially in the area of acquisition reform. This bold DoD initiative blends COTS-based solutions with streamlined management to produce superior products within tight fiscal and schedule constraints. Since the user and support depot were not necessarily current on this acquisition philosophy, conflicts often arose over development practices and perceived shortcuts. Some caution is also advised in the requirements area, since heavy user involvement at all stages of the program provides the opportunity for some requirements growth. This can lead to attempts to specialize the COTS away from the pure COTS baseline. Critical requirements must be solid.

Establish a baseline with COTS.

Since the COTS market is fast moving and ever changing, it was difficult to establish baseline architecture with COTS components if the program development schedule was longer than 18 months.[2]

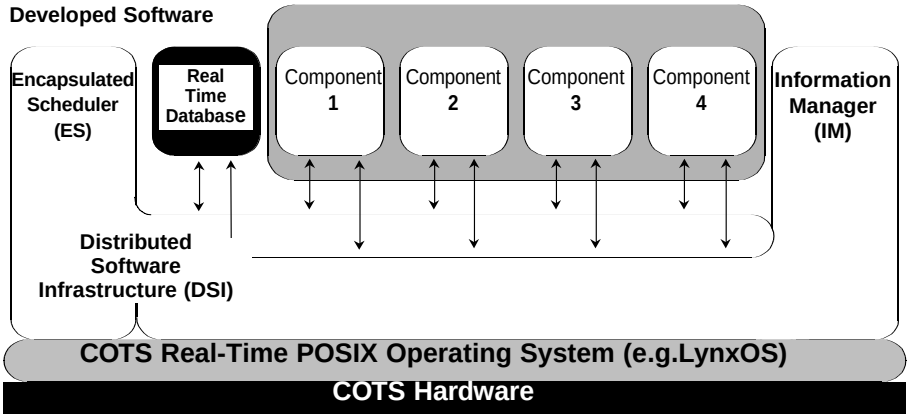


Figure 2. *U.S. Step 1 Software Architecture*

New products are introduced every six to 18 months, and often components chosen for the AWACS upgrades were phased out or no longer supported. As a result, it is critical to team with COTS vendors to ensure that the components chosen will last as least as long as the program's development and preproduction phases. This problem was especially apparent in the software area, where frequent new releases aimed at solving a select set of problems often produced new ones. Also, there was minimal documentation available from one software release to the next, so application developers had little insight into changes until anomalies were discovered.

Transitioning contractors is not simple.

Teaming with contractors with vast AWACS experience is vital to the success of any E-3 upgrade program. They have made a long-term commitment of building a specialized team of engineers, software specialists, and managers to support the unique requirements of a custom hardware and software system based on proprietary designs and technologies. However, object-oriented design techniques and implementation of COTS-based technology demands a change in design philosophy, subject matter expertise, and in many cases management structure. It was difficult for some contractors to effectively change and adapt in a timely manner. This was especially apparent in the software development area, where object-oriented software engineering techniques were not always well understood or implemented. This was primarily due to the nonavailability of modern software engineering and management skills within the company. Some contractors found it very difficult to hire new software engineers who possessed the necessary skills, due to the general market shortage of software professionals. The net effect was that software development and integration took longer than anticipated, which in turn severely impacted the overall development schedule.

Be on the leading edge, but not the leader.

More often than not, modern weapon system development is associated with leading edge technology. In the COTS-oriented world, while it is desirable to take advantage of the latest products, it is often painful to be one of the first users of a particular technology. You become the *de facto* beta tester of a product. Oftentimes, even if you are not the first user, you may become the first to stress the product's capabilities, sometimes discovering subtle deficiencies. This translates to additional time and money to troubleshoot and integrate. It is best to use COTS products that have some level of demonstrated maturity.

Carefully investigate tool availability.

The U.S. Step 1 program is based around the use of three programming languages: C, C++, and Ada. While implementing these languages for different parts of the design is not a problem, availability of advanced development tools was a challenge. For instance, since C and C++ are widely used, there are numerous software tools readily available to the software developer. However, while Ada was widely adopted by primarily government contractors over the past 15 years, other commercial developers did not embrace the use of Ada. As a result, development of advanced software tools for Ada lagged behind its C and C++ counterparts. During the development of the U.S. Step 1 program, there was only one vendor that offered a development suite (compiler, debugger, etc.) for

Ada 95. When problems, or suspected problems, occurred with the development suite, there were no alternative software tools to use. This was especially frustrating when the team was in the process of optimizing the software. There were no other compilers against which the efficiency of compilation could be compared.

Interfaces are not guaranteed compatible.

One of the primary advantages of using COTS is the adherence of industry standard interfaces. However, often a large integration effort is required—in some cases, larger than would typically be the case for custom solutions. The development team found that the compatibility of vendors' products depended heavily upon their implementation. For example, a Single Board Computer (SBC) drives the workstation displays with graphics support via a dedicated graphics accelerator. During the selection process, several vendors offered both products, but the team's assessment showed that the best performing SBC and best performing graphics card were offered by different manufacturers. The team did not see this as a problem since their electrical and mechanical interfaces were dictated and followed by industry standards. However, when assembled together in their intended configuration, the combination SBC-graphics card did not work. The compatibility problems were eventually resolved, but it involved intense and diligent efforts over a six-week period by a large team of engineers. Each vendor's implementation of the industry standards was slightly different, causing the unintentional conflicts.

Consider a COTS subsystem approach.

Choosing multiple vendors to supply common parts is good for competition. However, any advantage can quickly be erased by increased costs due to additional integration and debugging efforts. By choosing a solution based on a developed and tested product subsystem (for example, the SBC-graphics accelerator pair), the majority of the risk of integrating the subsystem is assumed by the vendor. In choosing this path, the AWACS team would likely save significant schedule and cost.

Model subsystems to understand systems.

Early in the design process, decisions must be made regarding the capabilities and performance of various components (i.e., computational efficiency, LAN bandwidth, etc.). Although product decisions are usually made after a system design is complete, our experience shows that a system model in early stages of development would have had a significant benefit in product selection, system architecture, and overall cost and schedule. Unfortunately, with most new COTS products, functional models (including performance information) do not exist. In addition, since the development team does not have insight into the internal architecture of the various COTS components, it is very difficult for them to develop their own models for such components. This modeling is critical to prove out the design. It is worth the additional time required upfront to either work with the vendors to help develop models or their components, or take the additional time required to measure the COTS product's performance in critical areas and characterize it.

Thorough market research is important.

Adaptation to the COTS world means market research is required—sometimes extensively. One of the problems with leading-edge COTS products is the lack of good, objective benchmark data. It often is inadequate or does not exist. This makes the selection process much more difficult. It would be wise to develop a set of benchmark programs that exercise all critical attributes of the desired system, and independently run those on products under consideration. Only then can one vendor's product be fairly judged against another. During development of the U.S. Step 1 program, this type of assessment would have likely led to the choice of alternate components.

An example involves the choice of Fibre Channel as the LAN for the U.S. Step 1 network architecture. When the decision was made, it appeared that Fibre Channel was making serious inroads into the LAN market, and would fit the performance requirements dictated by the U.S. Step 1 architecture. Unfortunately, a key requirement—the need to operate in Class 1 Mode (dedicated data transfer path)[3]—was available through only one vendor, which decided to exit the Fibre Channel market in the middle of the U.S. Step 1 development effort. More extensive research, coupled with asking the right questions, may have averted some problems experienced on the AWACS program.

Choose vendor as carefully as technology.

All COTS vendors are not alike, and their commitment to the program and their defense contractor partners vary. For example, some vendors excel at providing timely technical support, assist in troubleshooting problems, and provide early insight into new products. Others have a take-it-or-leave-it approach. Based on more than three years of working with various vendors, the AWACS team developed a list of highly desirable qualities for COTS vendors.

1. Choose vendors that have prior military experience. Those with some experience working with defense contractors were more likely to understand the unique customer requirements. They also were more likely to support the program for the long haul, as this is typical of existing defense programs.
2. Choose vendors that want to be on the team. For high volume manufacturers, the AWACS business represents a small portion of their market, so it is often difficult to receive adequate technical support on a timely basis.
3. Choose vendors that are committed to the market. Experience showed that some vendors would quickly change their business strategy regardless of any existing customer commit ments. It is best to try and select a vendor who plans on staying in their current market for the foreseeable future (if they are willing to share their corporate vision with you).

Partner with COTS vendors, meet regularly.

A formal face-to-face meeting on a regular basis with the contractors, vendors, and government provides a great opportunity to exchange ideas, concerns, and future plans by the government and vendor, etc. It gives all parties a comfortable sense of their current and future role in the program. It was during such meetings that the Fibre Channel vendor discussed their intent to exit the Fibre Channel switch market. Similar sessions with the other vendors allowed the government, prime contractors, and vendors to express their concerns on every issue ranging from manufacturing capability and schedules to future roadmaps for continued AWACS modernization.

A careful approach to DMS is required.

The use of COTS provides many technological advantages, but it also introduces new challenges. One of the most challenging tasks is that of logistical support of the fielded system. For example, numerous versions of COTS products may be identified with the same part number, but it is likely that the exact configuration of parts is not identical. One of the great advantages of using COTS is that you do not necessarily care about what is on the inside, as long as the device performs satisfactorily. Unfortunately, it has been determined from experience that this is not always the case, especially with respect to firmware. Since COTS technology changes at a very rapid pace, a philosophy must be developed that allows an affordable, flexible process with the potential to grow as COTS technology progresses.

Proactively plan technology insertion.

Although not implemented, the preliminary strategy planned for the AWACS computer modernization is centered on planned technology insertion. For example, the installation of production kits into operational aircraft was scheduled to take between five and six years. As a result, it is very likely that unless a lifetime buy of production hardware and software was made, different components would be installed within the fleet of 32 aircraft. Since a lifetime buy was not desirable from either a technology or cost-effectiveness point of view, the AWACS team needed to devise a strategy to address this issue. This strategy involved appointing a single manager take charge of tracking COTS, and making recommendations on product choices. This is a multifaceted project involving a number of specialized areas: market research, hardware evaluation, software evaluation, systems integration, etc. The manager would maintain a laboratory and cadre of people dedicated to tracking product changes and improvements, assessing new technologies and market trends. They would evaluate new hardware and software or updated version/releases in the system integration laboratory to determine compatibility, qualification testing, etc. Since COTS design information is generally proprietary, vendors are hesitant to provide detailed design data for performance assessment or debugging. Properly managing this effort requires working closely with manufacturers and vendors, and involves using nondisclosure agreements and other legal and managerial arrangements to ensure the project manager was well informed of any projected changes to the manufacturer's product line. Final approval for introducing new products/technology would involve the use of a configuration control board process specifically designed to address COTS integration.

Conclusion

The E-3 AWACS System Program Office attempted to incorporate COTS technology into a legacy weapon system. A brief overview of the existing mission computing system and upgrade program were given, along with specific considerations associated with COTS equipment on the E-3. Several challenges arose during development, including:

- Building user support for the COTS approach.
- Establishing baseline architecture with specific COTS components.
- Working with defense contractors transitioning from proprietary practices to the open systems approach.
- Availability of robust software development environments.
- The advantages of using entire COTS subsystems.
- Compatibility problems associated with commercially accepted interfaces.
- The need for modeling COTS behavior.
- Rapidly changing market dynamics.
- Choosing the right vendors.

Although this upgrade program did not lead to production, many valuable lessons were learned.◆

Acknowledgements

I would like to thank Thad Russell, MEI Technology; Elise Locker, Enterprise Systems Incorporated; and Joseph Bradley, Enterprise Systems Inc. for reviewing the original draft of this paper and providing comments.

References

1. Orfali, Robert; Harkey, Dan; Edwards, Jeri, *Essential Client/Server Survival Guide*, John Wiley & Sons, New York, 1994.
2. *DSI User's Manual*, Lockheed Martin Federal Systems—Owego, N.Y., 1997.
3. Sohal, Vik and Bunnel, Mitch, A Real OS for Real Time, *Byte*, vol. 21, no. 9, September 1996, pp. 51-52.
4. Milligan, Michael K., *U.S. Mission Computing Modernization Program 1998*.

Notes

1. POSIX is an industry standard that defines the interface between program code and the operating system. By adhering to the POSIX standard, program code is more portable—thereby avoiding many dependencies on specific operating systems or target hardware.
2. An additional constraint was the long deployment cycle into the field, since only about five aircraft (out of a fleet of 32) could be modified with U.S. Step 1 in any one year.
3. Fibre Channel defines three different classes of service—1, 2, and 3. Class 1 is a circuit switched connection in which an end-to-end data patch must be established before any data transfer can begin. When two devices are using Class 1, that path is dedicated to those devices and is not available otherwise. As a result, access (times) between those devices are guaranteed without any unpredictable interruptions. Class 2 and 3 are also switched services, but without first establishing dedicated paths. In the Class 2 and 3 configurations, data may flow between any two nodes, but the physical path is unknown (and therefore transfer times not predictable).

About the Author

Lt. Col. Michael K. J. Milligan is an assistant professor of electrical engineering at the U.S. Air Force Academy, Col. He previously served as lead engineer and program manager of the AWACS U.S. Step 1 Computer Modernization Program at Hanscom AFB, Mass. He holds a doctorate degree from the University of Texas at Austin, a master of science degree in electrical engineering degree from the University of Massachusetts-Lowell, and a master's degree of business administration from Western New England College. He also has a bachelor of science degree in electrical engineering from Michigan State University. His primary research interests include high-performance computer architecture and real-time systems.

Lt. Col. Michael K.J. Milligan
Department of Electrical Engineering
USAFA/DFEE
2354 Fairchild Hall, Suite 2F6
U.S. Air Force Academy, Colo. 80840
Voice: 719-333-6766 DSN 333-6766
Fax: 719-333-3756, DSN 333-3756
E-mail: michael.milligan@usafa.af.mil