2011 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
VEHICLE ELECTRONICS AND ARCHITECTURE (VEA) MINI-SYMPOSIUM
AUGUST 9-11 DEARBORN, MICHIGAN

RAYTHEON COE: MIDDLEWARE ENABLING THE TACTICAL PLUG AND PLAY FRAMEWORK

Al Stuessy
Combat Systems
Raytheon
Plano, TX

ABSTRACT

The Raytheon Common Operating Environment (RTN-COE) is a TRL-9 level integrated real-time operating environment that has been utilized in 40 plus Army, Navy, and Air Force programs. RTN-COE was created in 2000-2001 as an open architecture instantiation of the US Army's Weapons Systems Technical Architecture Working Group (WSTAWG) Operating Environment API and has since been propagated throughout the company. This paper will describe the evolution RTN-COE and specifically how RTN-COE is a key enabler of the Raytheon Tactical Plug-and-Play Framework that meets the low latency requirements associated with closed-loop operations with sensors, while also providing a gateway to the C2 applications within this framework. Finally this paper will elaborate upon the design considerations addressed by RTN-COE that have enabled it to: facilitate digital backbone integration, maximize scalability, enable multi-company software integration to shared processors, and promote software reuse and portability.

INTRODUCTION

The concept of middleware has been and continues to be a long-standing approach to solving several problems in the application software domain. The use of middleware at Raytheon and in industry as a whole is now accepted as a standard practice. However, the actual middleware products used in industry and even inside Raytheon are many and varied. Some organizations develop custom middleware on a program-by-program basis, while others adopt commercial products to meet their needs. This is precisely the issue that we ran into at Raytheon in the 1990-2000 timeframe. Just within the Texas region of Raytheon we experienced a situation where almost every new program was creating its own custom middleware solution. This created issues on a number of levels including: the inability to reuse code, the extra scope required to develop middleware on a program basis, the need to re-train engineers on how to use the new middleware, and the need to maintain the new middleware once it was created. These provided the impetus for creating a single middleware solution that could be used to address these issues – the Raytheon COE.

This paper will describe the evolution of the Raytheon COE, its role in enabling the Tactical Plug and Play Framework, the design attributes that are addressed by this product, and its use in multi-company integration efforts.

EVOLUTION

The US Army WSTAWG organization, of which Raytheon was involved with since its inception, developed an operating environment API specification during the 1997-2005 time period. WSTAWG came into being in 1997 to address the Army's problem of dealing with proliferating middleware approaches and hardware/software obsolescence. To the Army's perspective, seemingly every contractor was inventing their own custom middleware solution and there was little to no software reuse going on between programs. (Very similar problems to what we were experiencing inside of Raytheon.) In addition, the development of the WSTAWG API specification allowed the Army to mandate the use of middleware on future contracts to mitigate against the problems of COTS hardware and software obsolescence. This broad specification is still included as part of the JTA-Army.

As an outcome of Raytheon's participation in the WSTAWG standards body and the need to stop the proliferation of internal middleware products, the Raytheon

COE was developed in 2000-2001. The business case at that time for the creation of RTN-COE was Raytheon's participation on the Future Scout/TRACER vehicle development program. Raytheon, as a key subcontractor to BAE, had the role of Vetronics integrator for the demonstration vehicle. In that role, Raytheon in conjunction with BAE agreed that the Vetronics architecture would adhere to the WSTAWG standard and that the Raytheon COE would be the instantiation of the standard that would be used for the integrating the vehicle.

Although the eventual fate of the Future Scout/TRACER program was cancellation, the use of RTN-COE on this program was quite successful. The RTN-COE served as the backbone of the Lancer team demonstration vehicle (Figure 1) which made it through user trials in England successfully in 2003. To successfully integrate this vehicle required 1.5 million lines of code, 81 software engineers, and 12 development teams spread across two countries.

Figure 1: Future Scout Demonstration Vehicle - 2003

In 2001, just as RTN-COE was beginning to be used as the backbone of the Future Scout/TRACER program, the 9/11 tragedy occurred which spurred on the development of new drone programs such as the Predator (Figure 2). Raytheon had a key subcontracting role for General Atomics as the sensor developer for the Predator. That sensor, the Multi-Spectral Targeting System (MTS), required a software infrastructure product to facilitate the integration of its software and hardware subsystems. The RTN-COE was chosen to fill that role. This provided the first opportunity to leverage RTN-COE as a software reuse enabler, as RTN-COE was used to facilitate the reuse of a Feature-based Tracker software component from the Line of Sight Anti-Tank (LOSAT) program to run within the Predator platform. Leveraging the fact that both LOSAT and Predator were running RTN-COE as a backbone infrastructure, this Tracker was operational in one month, through flight tests in two months, and in combat with Hellfire missiles in 4 months. Since 2001, an entire Raytheon drone sensor product line for the Air Force, Army, Navy, and CIA has sprouted from this initial success. The RTN-COE is running in each of these now more than 20 programs.

Figure 2: Predator / MTS - 2001

During 2006, a cross-company initiative focused on software reuse was initiated from the Raytheon Missile Systems division. The objective of this initiative was to identify a common software infrastructure that could be used across new missile development programs within Raytheon. As a part of this effort a trade study was launched to determine whether this common infrastructure should be a Raytheon internal product or an available COTS product. The result of this trade study was that RTN-COE was selected over multiple COTS products for several reasons including: real-time performance, usability, availability of source code, and cost. Following this trade study, RTN-COE was successfully utilized on a pilot program – Small Diameter Bomb II. Subsequent to that success, use of the RTN-COE has proliferated into many other missile programs including: JAGM (Figure 3), SM-3, KEI, MRM, AMRAAM, and Maverick.

Figure 3: JAGM – 2010

ENABLING THE TACTICAL PLUG AND PLAY FRAMEWORK

As a continued part of the evolution of the RTN-COE, we became aware of the need to think in bigger terms. RTN-COE served the infrastructure framework portion of the problem well, but we were seeing the need for something more – a comprehensive architecture framework that

addressed the integration of Command and Control, C4ISR, and other electronics packages with crew stations. This was the impetus behind the creation of the Tactical Plug and Play Framework – a series of loosely-coupled sub-frameworks (Presentation, Sensor Control, Command and Control, and Communications) that tie together across a digital backbone to provide a cohesive vehicle integration solution.

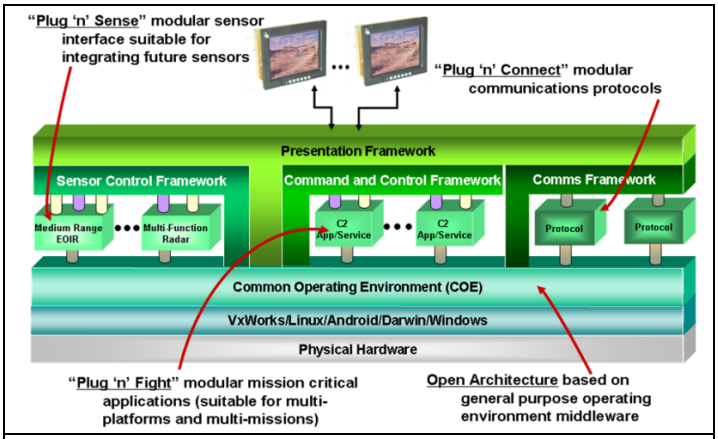

Figure 4: Tactical Plug and Play Framework

As shown in Figure 4, the RTN-COE enables the Tactical Plug and Play Framework by integrating the real-time Sensor Control framework with the other frameworks. The communication between the applications within the framework takes advantage of RTN-COE's publish/subscribe messaging capability which transfers data independently of the hardware and network topology. RTN-COE, however, does not underpin all of the sub-frameworks within the architecture as evidenced by the C2 framework which is based on Service Oriented Architecture (SOA). The architecture is defined this way because the most popular C2 applications are written to work with SOA infrastructures. To seamlessly integrate the C2 framework with the rest of the architecture, a SOA Gateway was developed. The SOA Gateway is necessary to bridge communication between the RTN-COE middleware and whatever native SOA infrastructure is being used to support the C2 Framework. The SOA Gateway adapts application message interfaces from their Standard COE Interface (SCI) format to an industry standard interface definition format that is more compatible with SOA called ICD 101.

In realizing a productized implementation of the Tactical Plug and Play architecture, another product named the Integrated Mission System-Platform (IMS-P) was born. The IMS-P (Figure 5) is a tactical open standards-based C4ISR system integration product that enables integration of discrete systems into a common framework. The IMS-P provides the "digital integration backbone" consisting of both software and hardware components that enables sharing of information between users on the same vehicle platform and between vehicle platforms over a tactical network. The software portion of IMS-P facilitates the creation of crew station Graphical User Interfaces (GUIs) and their integration with processing electronics and other associated software applications. The hardware portion of IMS-P hardware consists of the computers, displays, and networking equipment needed to integrate discrete systems together with vehicle user interface(s). RTN-COE is used as the infrastructure within the IMS-P that ties together all of the software components on the digital backbone.

Figure 5: Integrated Mission System-Platform

DESIGN ATTRIBUTES

Many attributes have driver both the initial design and upgrades to the RTN-COE over the past 11 years. These attributes include:

- **Real-Time Performance** – means the ability to address the performance constraints of real-time embedded systems. This attribute was important because Raytheon's Texas region primarily produces Electro-Optical and Radar sensors - types of products whose software content is microprocessor-based and subject to real-time constraints (33 msec or less).
- **Scalability** - means the ability of the middleware to run within a variety of processors from desktop computers to single board computers to digital signal processors (DSPs). This led our development team down the path of creating a single "core" for the product that could be deployed across the entire range of supported platforms instead of creating multiple "cores" or "editions" of the product. We wanted to design the product so

that it could scale to run within a $5 DSP just the same as it could run on a multiprocessor board costing tens of thousands of dollars (Figure 6).

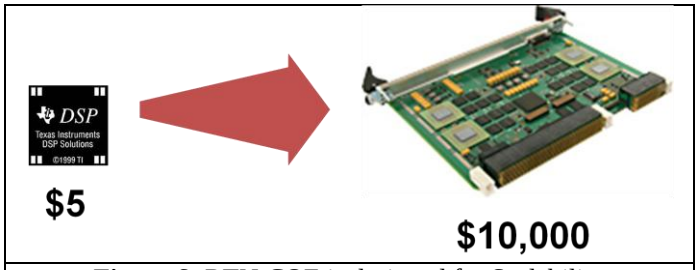

Figure 6: RTN-COE is designed for Scalability

- **Usability** – means providing a product that is easy for developers and integrators to use. This attribute manifested itself in the design of the product through both its APIs and internal architecture. As an example of this, when RTN-COE was ported to C++, an abstract API was created on top of WSTAWG which took advantage of features within C++ that decreased the number of lines of code that developers needed to write, therefore increasing their productivity.
- **Interoperability** - means the ease with which software and systems that utilize the RTN-COE would be able to communicate with each other. Examples of actions that we took in the design to address this attribute were the adoption of the standard WSTAWG wire protocol for message transfers and the addition of a RTN-COE standard Proxy service to interface to external components not running on the RTN-COE.
- **Portability** – means the ability to run software applications on the RTN-COE that can be moved from processor type to processor type or from operating system to operating system without modifying the application source code. Realizing this attribute was considered highly important as it would enable host-based testing of applications and most importantly would provide protection for the software applications from the obsolescence of the processor hardware and operating system software upon which they run within production systems.
- **Testability –** means the ease with which developers can test their components and systems using the RTN-COE. The need to address this attribute led to the addition of integration tools to the RTN-COE product. These tools include the COE Message Injection Tool (CMIT) and the COE Image Injection Tool (CIIT). These tools have been used on multiple programs and have increased developer productivity through facilitation of repeatable and automated testing of software and firmware components. CMIT (Figure 7) provides the ability to inject message data into and to capture instrument data from a RTN-COE-based system.

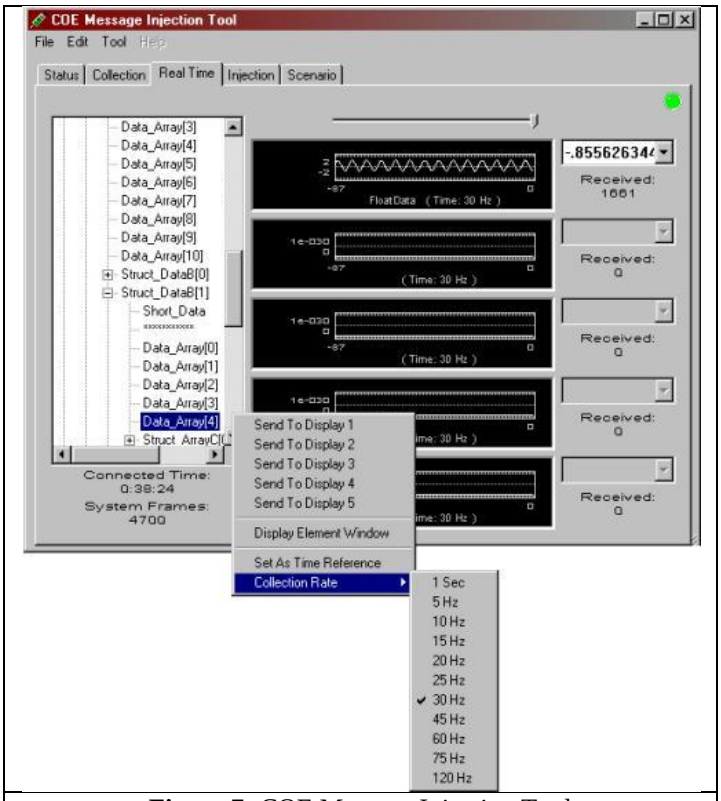

Figure 7: COE Message Injection Tool

MULTI-COMPANY INTEGRATION

To enable RTN-COE to function in the role of a multi-company integration middleware, the approach by which application data is defined for transmission within a system has changed since the inception of RTN-COE. Originally, all data to be transmitted by the RTN-COE was defined in terms of C/C++ source code header files or Ada packages. That approach worked on Future Scout/TRACER, but required hundreds of hours of developer labor to write the source code that defined the data for a large system. To optimize the entire data definition process including the amount of labor spent upon defining data, we added the ability to define system data using XML for the RTN-COE. To accomplish this we defined an XML Schema and a tool to parse the XML that we called the COE Message Automation Tool (CMAT) in 2006.

CMAT was designed to parse the developer's input XML data definitions and to generate source code in C, C++, C# or Ada from the input. This output source code is then compiled with the application into binary structures that are passed between applications. For efficiency reasons, ASCII XML is not transmitted between components or across the wire.

This approach of using data defined in XML and processed by CMAT into source code data representations is one that we have deployed across two multi-company integration efforts – FCS GSI and VIVID. With this approach, as illustrated in Figure 8, the system's XML becomes the artifact that is under configuration control – not the source code. For example, if Vendor X is supplying an INS component to the system, a set of XML is written that defines the interfaces to that component. That XML is then utilized by the software build process of everyone that needs to integrate with the INS including Vendor X.

This idea, once adopted and embraced by the all the software engineers, system engineers, and suppliers, worked well to successfully facilitate both the development and integration efforts on these programs.

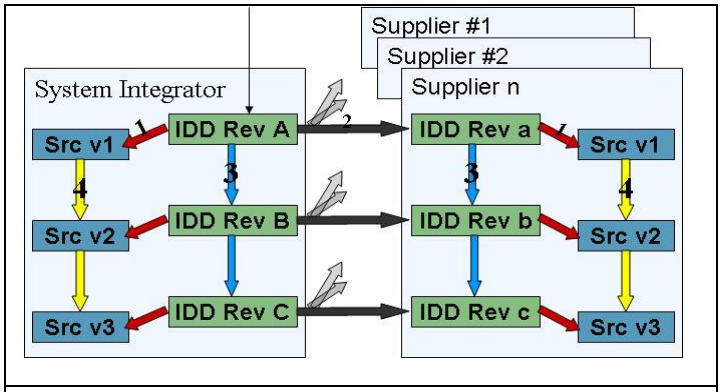

Figure 8: Multi-Company Integration with XML

CONCLUSION

At Raytheon, through our experience in middleware development and involvement with the US Army WSTAWG organization, we have proven that a single middleware product can be successfully applied to a variety of military programs. We have taken something that originated with WSTAWG and successfully propagated it for use on over 40 programs within Raytheon. We have also built upon these efforts to create a more encompassing integration solution called the Tactical Plug and Play Framework. RTN-COE enables this Framework by tying together the various sub-frameworks contained within the Framework into a cohesive whole. This Framework has since been realized in an implemented product called IMS-P. The IMS-P exists today and has been deployed upon several demonstration vehicles.