Joint Common Architecture Demonstration Lessons Learned – Honeywell Perspective

Alex Boydston
Electronics Engineer
US Army ADD/JMR
Redstone Arsenal, AL, USA

John Cunningham
Lead Software Engineer
Honeywell-Aerospace
Albuquerque, NM, USA

Matt Warpinski
Lead Verification Engineer
Honeywell-Aerospace
Albuquerque, NM, USA

ABSTRACT

Under the US Army (Aviation Missile Research, Development & Engineering Center) AMRDEC's Science & Technology (S&T) Joint Multi Role (JMR) Program, the Joint Common Architecture (JCA) Demonstration (aka, JCA Demo) project was established to exercise the Future Airborne Capability Environment (FACE™) Standard and Modeling Tools, validate the JCA concept and reduce risk for follow-on efforts. This was achieved by procuring a software component called the Data Correlation and Fusion Manager (DCFM) along with a Reusable Verification Component (RVC) from multiple vendors, built to the same specification and integrated on multiple undisclosed operating environments (OEs) designed to host FACE conformant Units of Portability (UoPs). Interaction between the software developers and the system integrator was tightly controlled to evaluate whether the information provided was sufficient to define a component that is modular, portable, and interchangeable. A representative acquisition approach was used whereby the interface requirements were provided in the form of a data model. This paper will discuss the experience, significant issues, and lessons learned.

INTRODUCTION

The complexity of avionics and mission systems is increasing at an exponential rate. This is due in large part to the proliferation of software-intensive capabilities and increasing demand for greater systems integration and superior performance. References 1, 2 and 3 estimate that the percentage of total cost related to software during the development of new aircraft is approximately 70% and likely to be higher in the future. These same studies suggest that the cost for developing the software is as much as $10B per aircraft with approximately 50% of this cost attributable to software rework. The complexity and cost is outpacing the Government's ability to develop, qualify and field safety-critical systems. The user community is clamoring for greater combat capability at the same time the United States Department of Defense (DoD) budget is shrinking. The DoD acquisition community must find new ways to procure systems that can affordably address these development and sustainment challenges.

In June 2010, the DoD partnered with Industry and The OpenGroup™ to establish the Future Airborne Capability Environment (FACE™) Consortium. The FACE approach is a Government-Industry software architecture standard and business strategy for acquisition of affordable software systems that promotes innovation and rapid integration of portable capabilities across global defense programs, initially intended for the aviation sector. The FACE Technical Standard (Ref. 4) defines software computing environment architectures and interfaces intended for the development of portable software applications targeted for general, safety, and security purposes.

The DoD is also developing a Joint Common Architecture (JCA) definition for the Future Vertical Lift (FVL) family of systems to increase affordability, reduce time to field, enable new technology integration, accommodate mission performance requirements, and address applicable mandates. JCA is an implementation-agnostic Reference Architecture (RA) that provides a conceptual framework describing a set of avionics subsystems and a functionally decomposed mission computer subsystem comprising a functional model and a semantic data model. JCA will provide a common vision and taxonomy, serve as a starting point for the design of avionics architectures, and support the development of an avionics software product line for FVL. It is envisioned that transitioning from a document-driven procurement process to a model-driven process will reduce the impact of software rework. The FACE Technical Standard and JCA RA enable a model-driven process by providing a framework and tools that can be represented in a model based language for solicitation and utilized for design, development and analysis through the lifecycle of software intensive systems.

JCA Demonstration Project

The US Army AMRDEC established the JCA Demonstration (JCA Demo) project to exercise the FACE Standard and Modeling Tools, validate the JCA concept, and reduce risk for follow on efforts. The JCA Demo approach was to procure a single software component from multiple vendors that would be integrated on multiple undisclosed Operating Environments (OEs). Interaction between the software developers and the system integrator was tightly

Presented at the AHS 71st Annual Forum, Virginia Beach, Virginia, May 5–7, 2015. Copyright © 2015 Honeywell International. All rights reserved.

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

controlled in order to maintain integrity of the experiment to the maximum extent possible.

The Modular Integrated Survivability (MIS) system in the Aviation Systems Integration Facility (ASIF) at AMRDEC Software Engineering Directorate (SED) on Redstone Arsenal, AL was selected as the target system and system integrator for JCA Demo. MIS comprises multiple OEs, multiple sensors, multi-function displays, and software aligned with the FACE Standard. JCA Demo sought to procure a Data Correlation and Fusion Manager (DCFM) software component that could be easily integrated into the MIS system. The DCFM would gather source track information from the MIS Situational Awareness Data Manager (SADM) and return a list of correlated tracks, combining those that are identified as a single entity. The MIS system executed a common scenario for each DCFM on each OE selected. Composition of the MIS system was withheld from DCFM developers until after software delivery. A depiction of the MIS System with the DCFM is shown in Figure 1 and a UML activity diagram with the DCFM and MIS SADM from the DCFM Data Model Description is shown in Figure 2. Note that only two operating environments were utilized in the JCA Demo due to schedule constraints.

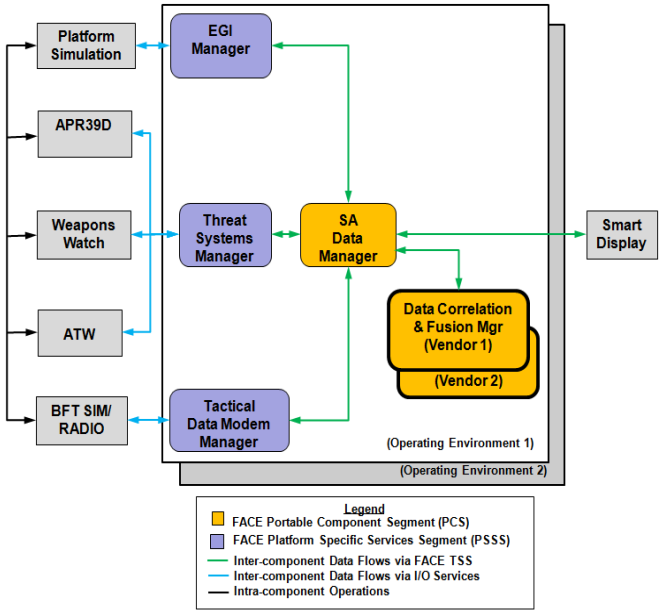

Figure 1 – MIS System with the DCFM

The DCFM was designed as an implementation of functionality from a preliminary version of the JCA RA. It was defined as a FACE Unit of Portability (UoP) where the interface was provided as a data model developed in anticipation of the FACE Standard Edition 2.1 Shared Data Model (SDM) plus two behavior component interaction diagrams. One of those interaction diagrams is shown in Figure 2. A minimal set of functional requirements were provided for demonstration purposes only. Processes, tools, and lessons learned were more important than component performance.

A FACE UoP describes a set of executable software which provides one or more services or mission-level capabilities required for complete and correct execution of that capability (e.g., programming language run-times and application frameworks). UoPs are a helpful method of packaging in cases where incremental capability is desired on an existing war fighting platform that already contains the required infrastructure (e.g., programming language run-times or application frameworks). The UoP can be bundled with other UoPs to provide a set of capabilities to a system. The DCFM software component is a UoP at the FACE Portable Component Segment (PCS) layer. Other UoPs within the MIS system are shown in Figure 2 and are at the FACE Platform Specific Services Segment (PSSS) layer.

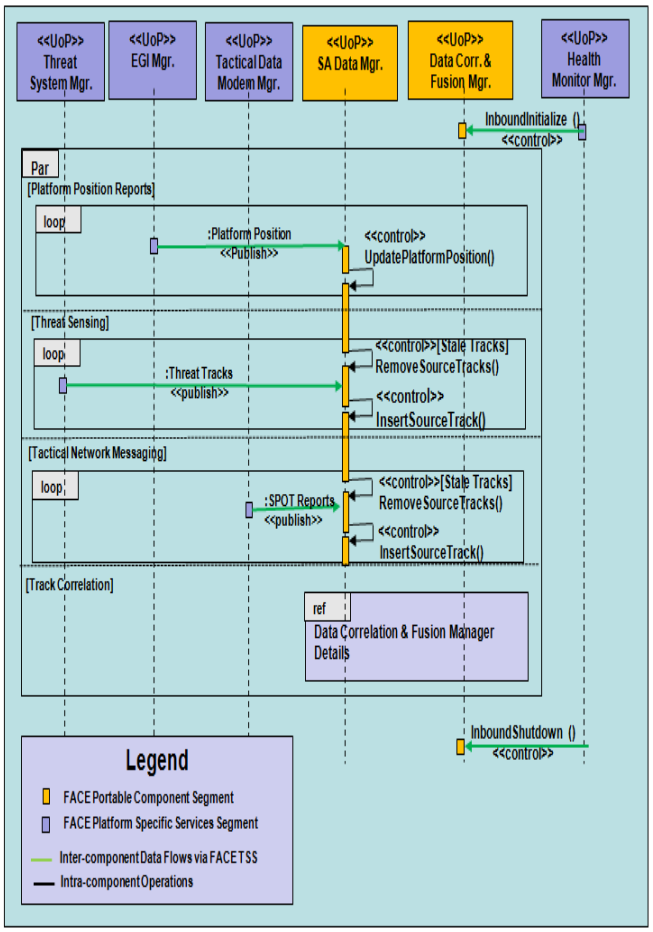

Figure 2 – System Activity Diagram of the MIS and DCFM

A Reusable Verification Component (RVC) was also required to provide unit test capability for any OE. The RVC was defined as the full suite of capabilities required to perform verification that the DCFM operated as intended and satisfied its software requirements when executed within any OE. The RVC captured input conditions, expected results, and traceability data that mapped RVC test cases to

corresponding DCFM requirements. The RVC ensures compatibility with host hardware and provides verification of the platform integration as life-cycle changes are made to each OE.

Finally, the DCFM was evaluated by the ASIF's FACE Verification Authority (VA) for conformance to the FACE Technical Standard. As was previously stated, the JCA Demo DCFM Data Model was developed in anticipation of the FACE Standard Edition 2.1 SDM. This version of the SDM was not published in time for its use in JCA Demo; thus, those requirements failed verification.

Solicitation

A representative model-driven acquisition approach was followed for JCA Demo. A Request for Information (RFI) sought comments on the approach and technical content of the planned solicitation (Ref. 5). Responses from Industry were positive on the approach with minor recommendations on the technical content. Based on the feedback the specification was updated and a Broad Agency Announcement (BAA) solicited proposals to procure the DCFM (Ref. 6).

Honeywell® Aerospace was one of two teams selected to enter into a Technology Investment Agreement with the US Army. The scope of the effort was to develop and deliver the DCFM and RVC software, submit FACE verification evidence, exercise the FACE Tools, provide on-call integration support, consider airworthiness implications, evaluate the overall approach and capture lessons learned.

DCFM REQUIREMENTS DERIVATION

Providers of avionics and mission software often develop software for specific systems and hardware platforms that meet requirements related to system integration, hardware capabilities, as well as software functionality. The JCA Demo project, in part, separates the system and hardware capabilities/requirements from the software functionality. As mentioned, the intent is to have software applications designed around the FACE Technical Standard such that they are portable. The effort to capture system and software requirements was based on supplemental material and supplied DCFM Data Model as the source.

Difficulties existed in extracting/deriving some requirements from the DCFM Data Model and supplemental material because information was either missing or vague. One reason for this lies in the use of the older FACE Standard Edition 2.0. With FACE Standard Edition 2.0, the meta-model relies heavily on the use of textual specification rather than explicit modeling elements to define the semantics of basic elements and entities. The text entered in the DCFM model was brief and non-specific. For example, in most cases the descriptions of elements at the logical and platform level were identical to the general description used at the conceptual level. Hence, this caused some confusion about the meaning of the logical measurements and their platform realizations.

As part of the JCA Demonstration experiment, information about the system and hardware platform was purposely withheld to control the experiment of acquiring a FACE software component (i.e., the DCFM) and test its portability to different OEs. Not having knowledge of the system and hardware led to some confusion. An example is the *period* attribute of message ports, which was unspecified in all cases. The *period* attribute is only allowed to be unspecified when a message is aperiodic; it must be specified for periodic messages. This, combined with the vague timing requirements in the supplemental material made it impossible to define temporal requirements of the DCFM without further specification.

Since FACE Standard Edition 2.0 defines only software architecture and the interfaces between components (e.g. DCFM and other Units of Portability (UoP) such as the SADM) and no system characteristics, such system information must be conveyed in documentation other than the data model. No guidance about system features was provided beyond that contained in the DCFM Data Model and the supplemental requirements. Examples of non-disclosed information included the possible number of concurrently executing instances of the MIS's SADM or the DCFM UoP, whether the timestamps on the source "tracks" coming from different sources used a common notion of system time, and the maximum number of concurrent active source or correlated tracks. In a safety-critical environment such information is required at design time because system resources (e.g., time and memory) must be allocated by design to insure deterministic operation at runtime.

The requirements were discussed during the technical coordination meetings between Honeywell and the Army. During those meetings specific missing information was identified along with assumptions as to the intended requirements, so that the Army would be aware of the shortcoming of the specifications and that Honeywell could get confirmation of the intended requirements.

Several timing requirements were, in fact, specified for the DCFM UoP in terms of frame rates and periods. For example, one requirement stated "the DCFM shall correlate or decorrelate 50 source tracks within 1 second". Timing requirements such as this can only be verified when implemented on hardware which was not made known, again due to experiment control to learn about the FACE Technical Standard. As mentioned, until software delivery, the only understanding of hardware by Honeywell was that the DCFM UoP was intended to run on "several target platforms".

Using hardware design as an example, integrated circuits (ICs) are acquired as commodity off-the-shelf elements and connected together to establish system-level functionality. The FACE Technical Standard intends to

addresses software in a similar fashion, with software elements being acquired off the shelf with the goal of interconnecting these software elements to achieve integrated system-level functionality. For the hardware designer, IC data sheets provide not only interface details (i.e. input X and output Y), but also a description of timing constraints and dynamic behaviors required to integrate the device with other elements using signals such as clocks and enables. The combination of static interface and dynamic behavior descriptions supports integration of the ICs into higher level systems. For ICs, timing constraints are defined parametrically as a function of clock cycles. For software, specifying real world timing constraints is challenging since it is dependent on the supporting hardware. Thus, an approach that parametrically specifies software timing constraints as a function of hardware specifications should be considered in the integration of the software on the target hardware.

It was intended for the FACE Modeling Tools to be used by the system integrator to insert the DCFM UoP into the system architecture. These tools would allow the system integrator to connect the DCFM UoP interfaces into the system and would also be used to specify various attributes such as the UoP execution rate, UoP priority level, UoP deadline type, etc. This gives the system integrator the ability to execute the software UoP at any rate that the system integrator desires within the limits of practicality. This implies that requirements related to the UoP's rates and priorities are the system integrator's responsibility and not the software UoP provider's. This begs the question then of how to specify desired performance in terms of throughput and time capacity for software UoP providers.

For the JCA Demo, Honeywell essentially provided a requirement back to the customer dictating the rate at which the DCFM UoP was designed to operate in order for the Kalman filter in the correlation algorithm to work properly.

Although the system/hardware requirements may not be the ultimate responsibility of the software UoP provider, they are necessary for functional testing of the application. In some cases, the DCFM rates were specified in the requirements but not in the DCFM Data Model. A question that arose out of this included how to ensure the receipt of SADM responses without knowing if the SADM and DCFM are in different partitions. The SADM provides source tracks to the DCFM as well as being the consumer of correlated tracks compiled by the DCFM UoP.

The success of the UoP software lies in some part with the validity of the provided data model; however, the provided data model was known to not fully comply with the final FACE SDM. Again, this was due to the fact that the SDM for FACE Standard Edition 2.1 was not finalized when the JCA Demo BAA was released. As a result, FACE conformance testing failed as expected.

The completeness of the defining requirements is not by default ensured by use of model based techniques. Systems engineering is still required. Requirements analysis prior to solicitation to ensure requirements consistency, completeness, and accuracy is still needed. In some cases it may be necessary for the system integrator to communicate with the UoP provider to ensure that integration concerns are addressed in the development of the UoP. For those UoPs already present in the FACE Repository, enough technical information about the UoP must be provided to allow for the integration considerations and analysis of the software. This would include certification and qualification information. Models of the software components containing information with interface and behavioral data would be useful in the system integration analysis.

DCFM ARCHITECTURAL DESCRIPTION

The sequence diagrams provided in the BAA DCFM Data Model Description document showed synchronous request/responses within the same loop, but the DCFM could not receive response data within the same partition time window in which the request was made if the SADM and DCFM are in different ARINC 653 (Ref. 7) partitions. If the DCFM has a single partition window per period, responses would be received in the period after the requests are made. The assumption made was that the MIS SADM and DCFM were in different partitions and that the DCFM has a single partition window. The assumption was also made that the SADM would execute between the DCFM time windows. Provided the SADM and DCFM are scheduled at the same rate, it is possible for the data/process flow of the original sequence to be as depicted in Figure 3. In this example it would take multiple periods, or time windows to complete the DCFM processing of a single set of data. As a software application provider, it is necessary to have a clear and detailed understanding of the data interface timing and latencies for an application like the DCFM. While the software could be architected with states to achieve the flow depicted above, it would not be efficient.

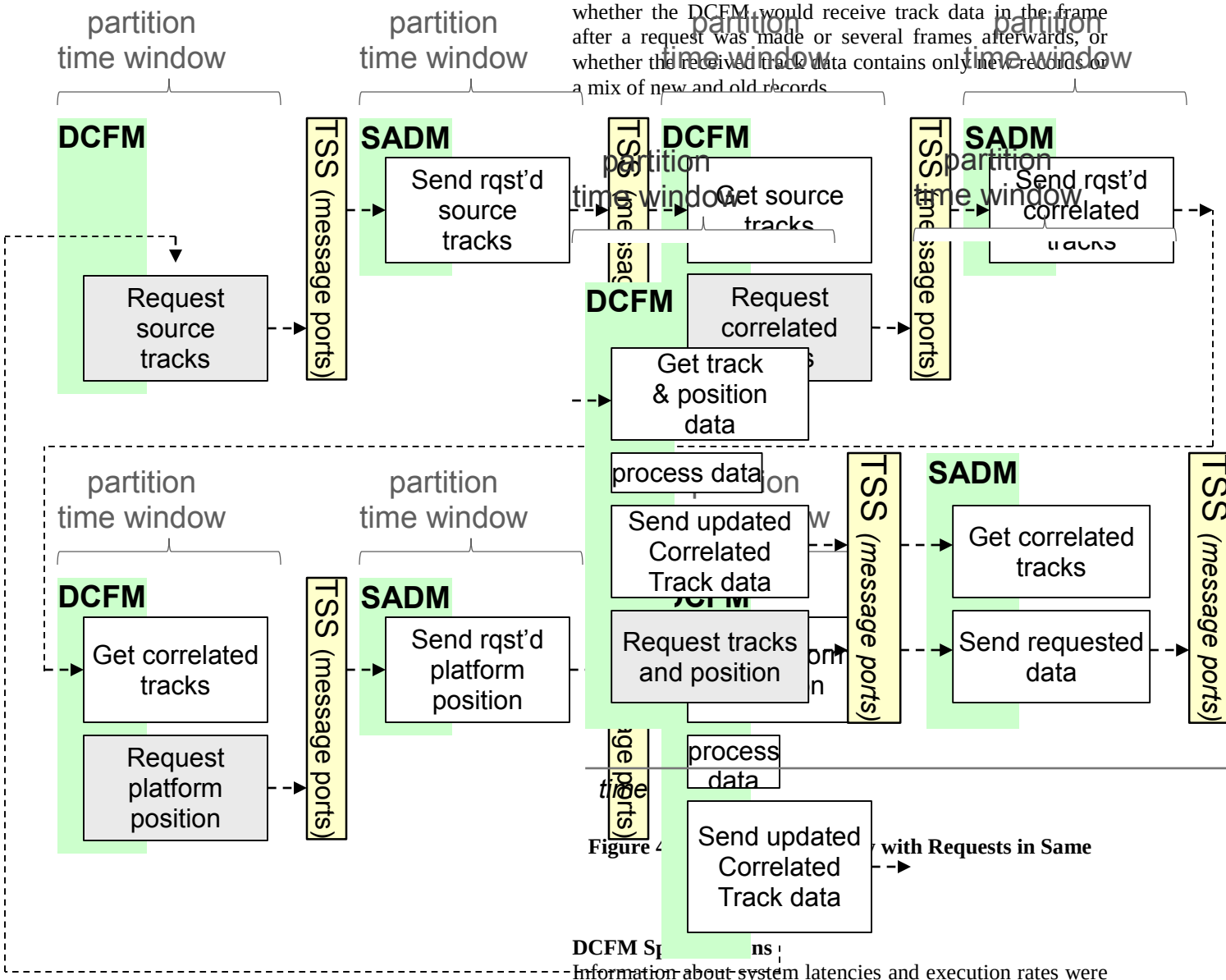

Figure 3 - Original Data Flow with SADM and DCFM

After Honeywell worked with the Army, it was decided to update the sequence diagram such that all of the requests could be made within the same ARINC653 partition frame period. With this new understanding of the interface, the DCFM could then receive responses to all requests, process the data, send all of the correlated track data back to the SADM, and issue new requests within each period. This new data/control flow is depicted in Figure 4. With this change, every DCFM period will process a complete set of track data for the SADM. As seen in Figure 2, the DCFM interfaces with the SADM UoP by retrieving track data, suppling updated track data back to the SADM, and then requesting track data all within the same partition frame period. System latencies, such as how long it would take for the SADM to receive and respond to the request, were unknown. It was also unknown if the track data would contain old or stale tracks. As a result, several questions came up including whether the DCFM would receive track data in the frame after a request was made or several frames afterwards, or whether the received track data contains only new records or a mix of new and old records.

Figure 4 - Data Flow with Requests in Same

DCFM Sp ns

Information about system latencies and execution rates were not provided in the DCFM Data Model or supplemental requirements material, which created additional challenges. For example, the correlation algorithm uses a Kalman filter and depends on a known frame rate. Large latencies in track data relative to the Kalman filter processing rate must be accounted for by the fusion algorithm or problems would arise.

Several observations made were: (1) the UoP interfaces need to be defined to a higher level of detail and clarity, (2) UoP timing and latency concerns need to be addressed and understood, and (3) working early with leading providers can provide details that may improve the overall system architecture and UoP requirements which may be critical to successful integration.

Current system providers that maintain both the system architectures as well as the software applications can more easily change system interfaces to improve efficencies. In a FACE architecture where multiple suppliers are involved, it

may be harder to change system interfaces without affecting multiple suppliers. In systems where a new UoP is being procured, working with suppliers early to identify detailed understanding of various interfaces may be of benefit, not only in defining the system architecture, but also in creating a clear and concise interface requirement for the software component.

DCFM Implementation

During implementation it was discovered that the Transport Services (TS) interface code generated from the DCFM Data Model appeared to compile and execute, but data was not being sent through the message ports. Investigation revealed that ARINC 653 requires message ports to be bounded in size. In order to support both POSIX (Ref. 8) and ARINC 653 operating systems all data ports must be bounded in size. As a result the DCFM Data Model had to be updated to specify the maximum number of records available in the data structure. Another field was added to indicate the number of records actually used. This provided the required determinism for message port sizing necessary for multiple targeted operating systems.

There are several ways to implement component rates. The FACE modeling tools have the ability to specify component priorities, frame rates/periods, and time allocation. The software application provider may also spawn tasks with specific frame rates. Requirement specifications should make clear who is ultimately responsible for software UoP timing, (i.e., the software UoP provider or the system integrator). When the system integrator takes on this responsibility they gain control of the software UoP execution periods giving them greater flexibility; however, when the software supplier is responsible for software UoP timing the system integrators lose that flexibility and adds more complexity for the software supplier by having them manage multiple tasks within the component. Either solution may be desirable and should be taken into account and made clear when writing the software UoP specification.

UTILIZATION OF THE FACE TOOLS

One of the core goals of the JCA Demo was using the Vanderbilt University Institute for Software Integrated Systems (ISIS) candidate tools and testing the ability of the tools to auto generate all applicable FACE aligned code, and testing the software with the FACE Conformance Test Suite (CTS). The DCFM Data Model generated by the Army and provided to Honeywell only modeled the data sent through TS to the DCFM. In addition, the data sent through TS from the RVC needed to be modeled using the same data structure, and regeneration of the data model was required for ease of software development.

After the Army modified the DCFM Data Model, Honeywell was able to create all the necessary FACE connections to model the lower part of the Honeywell DCFM architecture shown in the Figure 5. The FACE Software Development Kit / Integration Development Kit (SDK/IDK) then was able to generate the software, by creating stubs into which the Honeywell custom software could be inserted. This code was then moved to the Linux work environment. The code was compiled, linked, and executed in the provided Vanderbilt ISIS ARINC 653 Component Model (ACM 653) emulator, where it worked without issue. However, when this code was run through the FACE CTS, issues began to surface.

Code created using the FACE SDK/IDK is tailored for the ISIS ACM 653 emulator. The emulator needed specific non-ARINC 653 commands to operate. The tools created these function calls; however, they were not FACE conformant calls. Honeywell was able to manually comment out the non-conformant code to pass conformance testing. Commenting out this code partially negated the link between functional testing and conformance testing, but it allowed for the code to pass conformance testing.

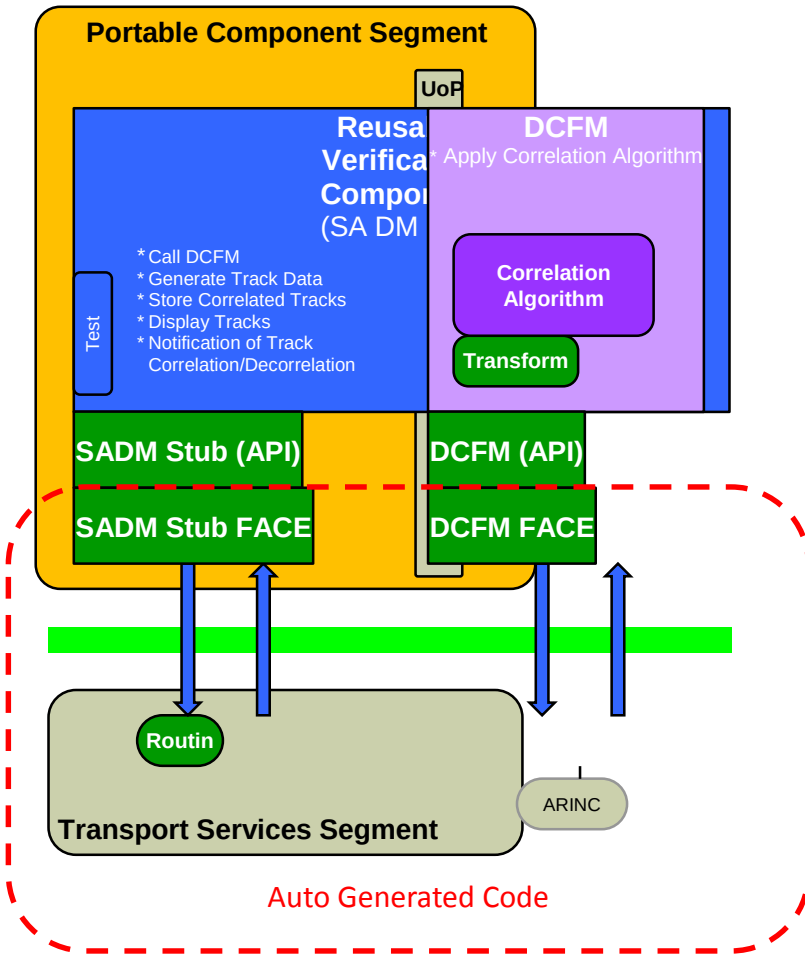

Figure 5 – FACE SDK/IDK Auto Generated Code

During development, these tools saved Honeywell considerable effort by generating FACE aligned code. Using the FACE SDK/IDK tools is an easy starting place for any company entering the FACE conformant software market.

DCFM AND RVC DEVELOPMENT

Early in system design the Honeywell team discussed data structures and how they were to be handled. The FACE SDK/IDK created data structures automatically; however, those data structures were tied directly to the data model. As a software supplier, Honeywell did not control the DCFM Data Model, which was generated by the Army. Due to the aforementioned shortcomings in the DCFM Data Model, another revision of that model was issued. Honeywell decided to create its own internal data structure for this effort for ease of development and testing, flexibility to add additional data as needed, and as to not rely on an external data model.

A question arose as to which software component was going to use this data structure, (i.e., would the RVC and DCFM each have their own data structure, or should they be combined). The desire was to ease the creation of a testing platform. This drove the decision to have a single data structure applied and maintained across the software development pieces (i.e., the DCFM and the RVC).

Having this additional data structure coupled with the FACE Standard introduced some new issues. Under the traditional acquisition process, one supplier would own both the data generator and data consumer (i.e., the DCFM in our case as the data consumer). Traditionally, these could also share a block of memory, allowing both to operate on one copy of the data. The data would then be transmitted to the Correlation Fusion Algorithm, with possibly some transforms creating a second copy of the data as seen in Figure 6.

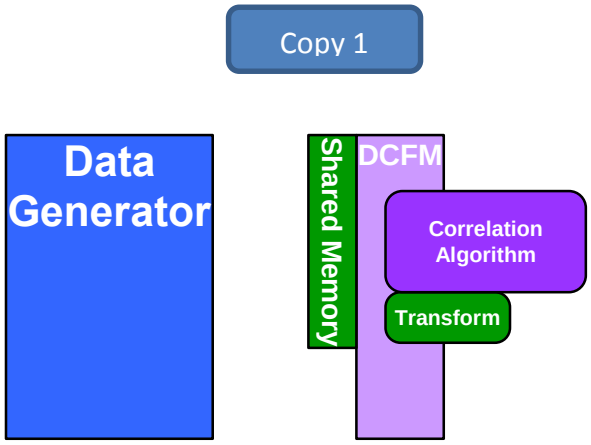

Figure 6 – Traditional Data Structures

Using the FACE Standard as well as the Honeywell data structure, additional copies of data were introduced as shown in Figure 7. First the RVC brings the data in and stores it internally (i.e., Copy1). Then the data needs to be copied to the FACE-aligned auto-generated data structure (i.e., Copy2). From there the data is transmitted to TS, and TS stores a local copy of the data (i.e., Copy3). That data is then transmitted to the DCFM (i.e., Copy4). Now the FACE-conformant auto-generated data structure converts the data into another data structure (i.e., Copy5). Subsequently, the transforms are applied to the data to send to the Correlation Fusion Algorithm (i.e., Copy6).

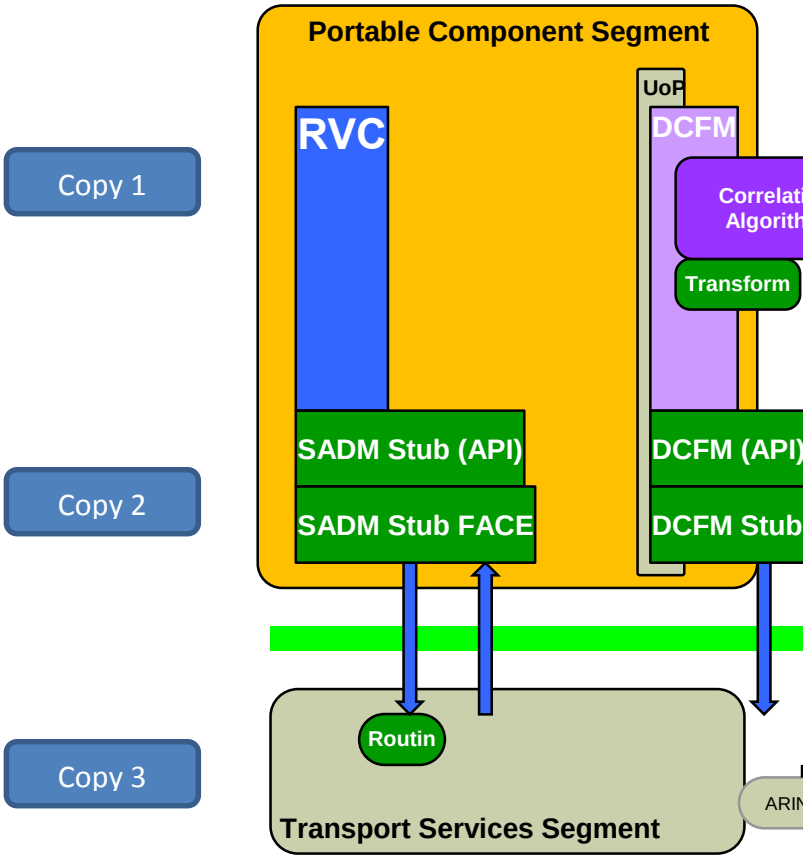

Figure 7 – Data Structure Copies

For applications with tight storage requirements, the design decisions might not bode well, given this memory allocation overhead. In this case, the concerns were the necessity of making six copies of the same data. Even without the decisions made in this case, any FACE conformant solution would require generation of at least three copies, as shown in Figure 8.

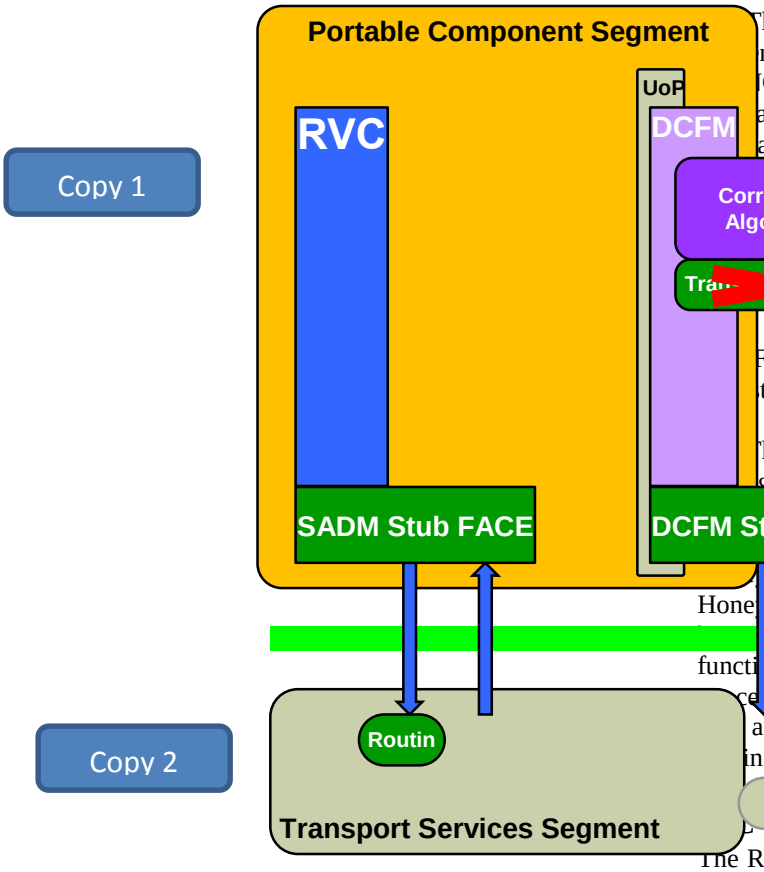

Figure 8 – Data Structures with FACE Conformant Implementation

DCFM

Three observations arose during DCFM development. The first two observations pertain to generation of the FACE Gold Standard Libraries for the ARINC 653 environment for the Safety Profile by the Candidate FACE Tools.

The first observation is that the code generated for the FACE TS interface includes "ACM wrapper functions" which are needed to execute in the ISIS ACM 653 Simulator. However, these functions would not be used when deploying to a Commercial off the Shelf (COTS) ARINC 653 Real Time Operating System (RTOS) or a future FACE aligned RTOS. Hence the code must either be conditionally compiled for the two environments (which is not allowed by the FACE Technical Standard) or the code must be edited before being submitted for verification or for execution on a COTS RTOS, which means that the verified code is different than the code used for development and testing in the environment supplied by the FACE Tools which was required by the BAA. This observation was reported to ISIS on Aug 29, 2014 and was discussed with ISIS during the meeting held with JCA Demo developers on Sept 11, 2014. During that meeting, it was agreed that the FACE Tools should be modified to eliminate the need to modify the generated code prior to delivery.

The second observation is that when the tools are used nerate the FACE Gold Standard Libraries for the C 653 environment for the FACE Safety Profile, the ated file named common.hpp tries to include the header amed stdint.h, which is not allowed for ARINC 653 s results in a compile error, because stdint.h nce the Gold Standard Libraries are used in nformance to the FACE Technical Standard, be an obstacle t formance. A d was identified w the generated to use FORMANCE_TEST_FACE_EXACT_TYPES.h rather tdint.h to define the standard data types.

The third observation is that of the Health Monitor ger (HMM) interfaces which were specified in the vever, an actual HMM was not supplied as Demonstration program. It was up to well to code the functionality of the HMM. Hone well modeled the HMM in the RVC component with to how it operates and what the functions actually do. With these assumptions there is the ce that when integrated onto the targeted hardware a real HMM, operation may differ both in functionality in terms of timing interactions.

The RVC has a wide range of requirements that drove early decisions to simplify the operator interface, test scripts, and data size. First, the RVC was required to send observation position every second for each source target. The first approach considered to meet this requirement was to have a database of observation points and to simply step through this database and send the data. This created a fixed data set with very repeatable outputs. This also made the scripts much larger and harder to tweak.

The other option was to create a starting position, heading, and velocity. From that starting condition, the current position would be incremented along the specified vector. This creates a scalable data set where it is easy to create new data points, but also makes it harder to create test repeatability. The test repeatability issue was resolved with the implementation of test scripts that defined those values. Most of the test scripts could be reduced to a handful of lines, given source track starting information, position update rate, and a specific source track lifetime.

In addition to starting and stopping the position track simulation, the ability to change the track's heading and velocity was added. One issue encountered with this implementation was the instantaneous change in velocity and heading, which is not realistic. Future improvements could include a model for how a target, aircraft, or ground vehicle would turn to depict inertial affects instead of an instantaneous change.

RVC as a FACE UoP

The goal of the RVC was to provide simulated data to the DCFM in order to verify proper operation of the UoP. The RVC was not intended to go into a real mission computer platform, but only to be used for integration and testing of the DCFM. As such, the design of the RVC was not constrained by FACE conformance requirements. However, the RVC needed to act as a UoP to feed data to TS and ultimately to the DCFM.

The first attempt at doing this was to take the DCFM auto-generated code and modify it to create another FACE UoP-like code segment. After a few weeks, another design path was chosen: start with the Army provided model and add the RVC and appropriate connections to this model before code was auto-generated. This allowed all the connections to be built into the DCFM, RVC, and TS. Repeating this process for a new data model or any other change took less than a day for one engineer, which is quite a substantial decrease compared with the initial strategy. This also allowed the RVC to be a UoP, and to pass FACE conformance testing if it were required.

FACE VERIFICATION AUTHORITY EXPERIENCE

After the Army's receipt of the DCFM software, the ASIF FACE Verification Authority (VA) conducted FACE conformance testing. The DCFM passed except where the data model did not conform to the FACE SDM. A few other minor issues from Conformance Testing will be addressed in the final delivery to the US Army. Note that the DCFM did not pass FACE conformance due to release of the DCFM Data Model in the BAA before FACE Standard Edition 2.0 SDM had been finalized. It was decided by the Army that the DCFM Data Model would not be modified to be updated to the FACE Standard Edition 2.1 SDM to pass FACE conformance due to time and resource constraints on the program.

There are more than 300 requirements in the FACE Standard. Conformance testing only verifies that the UoP links to the Application Programming Interfaces (API's) allowed for particular profiles and segments applicable to the UoP. The FACE VA is responsible to evaluate all of the other requirements manually.

In support for the FACE VA activity, Honeywell was asked to provide a FACE conformance statement answering several questions from the FACE VA. The information gathered was provided to the Army, acting as a surrogate Program Management Office (PMO), and forwarded to the FACE VA. Additionally, a mock registration of the software onto the FACE Registry was exercised. Suggested improvements to the process were provided back to the FACE VA and Registry authorities by the surrogate PMO. As part of these improvements, it was recommended that the FACE VA provide a workflow guidance document to aid the organization seeking FACE Conformance and UoP Registration. Also, there were several minor recommendations to the FACE Library Subcommittee with regard to improvements suggested for the registry web site. Another recommendation to the FACE conformance process was to provide some notification or status web page to indicate the current status of particular conformance verification.

DCFM INTEGRATION WITH MIS

As mentioned earlier, as a control on the experiment, Honeywell Aerospace was not provided insight into the system into which the DCFM would be integrated prior to the delivery of the DCFM. As later revealed, the system chosen to integrate the DCFM into was the Modular Integrated Survivability (MIS) system. The MIS components consisting of the Situational Awareness Data Manager (SADM), Sensor Manager, and Digital Map Manager and the supplied DCFM were contained in separate partitions. Those MIS components are shown in Figure 9 in the context of the FACE architecture on a representative mission processor.

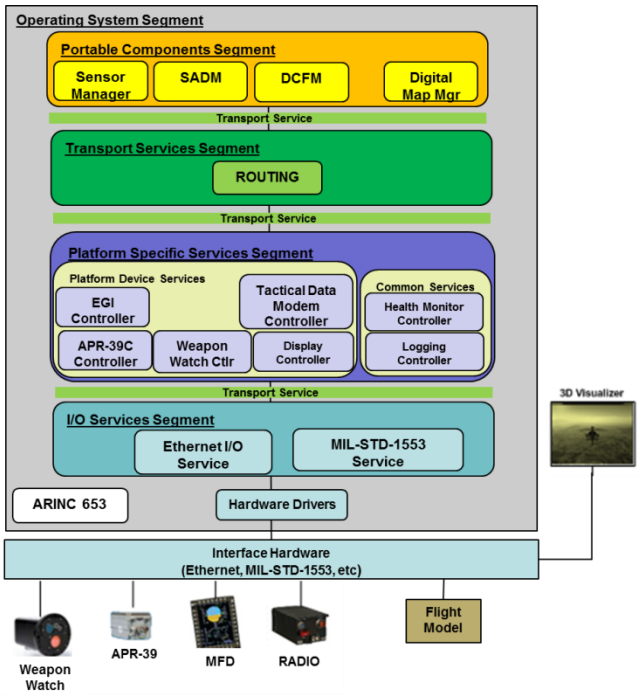

Figure 9 – MIS Lab Infrastructure

The MIS team's objectives, in alignment with the JCA Demo objectives, were to show portability of components to two different operating environments (OEs) and to demonstrate simulated threat correlation scenarios with concept of operations including hypothetical flights of Apache AH-64 and Blackhawk UH-60 helicopters. Detection of threats was to be observed on a moving map on a multi-function (MFD) display.

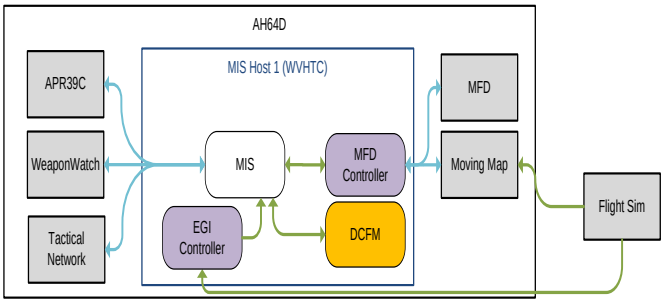

Figure 10 – MIS OE#1 Logical Configuration

Operating Environment #1 (OE#1), representing the AH-64D logical configuration shown in Figure 10, consisted of VxWorks653 Operating System (OS) running on a Curtiss-Wright SVME-183 (x2) computer which was provided by West Virginia High Technology Consortium (WHVTC).

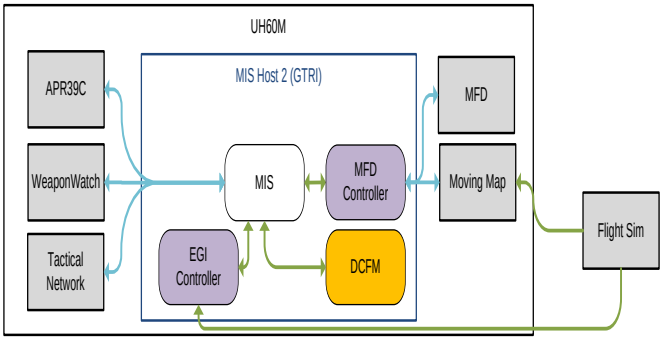

Figure 11 – MIS OE#2 Logical Configuration

Operating Environment #2 (OE#2), representing the UH-60M logical configuration shown in Figure 11, ran on LynxOS178 on a Kontron PENTXM2 (x2) computer provide by Georgia Tech Research Institute (GTRI).

Pre-integration on Simulated Linux ARINC 653 OE

The MIS Integration team conducted compilation of the DCFM software with very little difficulty. The DCFM was first recompiled in a Linux ARINC653 environment and the RVC tests were performed again by the system integrator, resulting in no errors.

The MIS team learned during compilation that there were some details in documentation, such as version of the compiler and build management system that were missing; thus, more complete documentation of the development tools is recommended. Also, the MIS team used a 64-bit version of Linux, and they ran into issues with message structure sizes between the Linux based RVC software and the external Windows RVC controller. This problem could have been avoided by the DCFM supplier providing a list of tools and their versions to compile and execute the DCFM component.

Lastly, during the pre-integration phase, the MIS team learned a lesson with regards to memory requirements. Typically, the system integrator would mandate memory requirements to a software provider. In this experiment, the system integrator did not provide such requirements. Based on experiences in this experiment, if memory requirements for the procured software are not defined by the system integration, memory usage should be documented by the software provider with delivery.

Operating Environment Integration

Integration efforts continued with the OE#1 (VxWorks) and OE#2 (LynxOS) with no issues found in the DCFM. A problem, not related to the DCFM software, with MIL-STD-1553 drivers on the OE#2 was encountered which would eliminate the capability of the MIS system to communicate to the APR39 sensor if not corrected. At the time of writing this paper, the MIS team is working to correct that issue. This issue caused a 2 month slip in this effort and affects the findings presented in this paper.

Integration Experience on OE#1 (VxWorks)

A variation between the simulated Linux ARINC 653 environment and the WindRiver VxWorks653 RTOS was identified by the MIS team. The VxWorks653 RTOS configuration defines uniquely named connections allowing data transfer between components. The connection names are limited to 30 characters in the VxWorks653 and some names for the DCFM were longer and had to be modified to make the integration work. Thus, either a limitation to fit all names within 30 characters or making the RTOS vendors support longer names is required.

Differences in the MIS TS implementation and the TS generated by the FACE tools forced some software changes to functions that set up the connection and message receipt. The TS does not require the size of the message to be set for these functions. Also, a recommendation was made by the MIS team to the FACE Technical Working Group to designate the data interface fields as input, output or input/output to ease porting between TS implementations. Some software function calls generated by the FACE Tools produced function calls specific to the simulated ARINC-653 environments that needed to be replaced with function calls provided by the RTOS vendor. Either the FACE Tools should generate the function calls per different RTOS vendor or the RTOS vendors should support the function calls generated by the FACE tools. Also, the VxWorks RTOS required a header file to be included which defined the function calls. The ARINC 653 standard does not standardize the name of this file. Lastly, for the OE#1 integration the MIS team had to modify the manner in which some objects were allocated in order to produce a functioning system.

Integration Experience on OE#2 (LynxOS-178)

A variation between configuration data requirements between the Lynx LynxOS-178 and the WindRiver VxWorks653 RTOS was identified. Each RTOS has

configuration data that defines uniquely named connections allowing data transfer between components. Using special characters such as '<' or '>' within the connection name was permissible within VxWorks653, but not within LynxOS-178.

The LynxOS-178 implementation of the CREATE_PROCESS function call differs in the manner it is implemented from both the simulated ARINC-653 environment and VxWorks653. The LynxOS-178 implementation expects a parameter in the API call to specify the path to an executable which it will execute as a new process. The simulated ARINC-653 environment and VxWorks653 expect a function pointer to be provided where the execution of the new process will start.

Integration Experience with the RVC

The RVC was clearly useful to provide a verification of DCFM functionality. Additionally, the RVC helped to compare behavior seen with the DCFM integrated with the MIS SADM against behavior of the DCFM integrated with the RVC. Although not conducted, porting the RVC to the embedded target and testing with the DCFM on the target would also reduce risk prior to integration into the target system. This method would isolate issues regarding porting to a differing OE from issues integrating software applications.

Integration Experience with the FACE Tools

Although the integration team did not use FACE Tools directly, the team did receive software generated using those tools from Honeywell. Porting software generated by the tools to an RTOS identified several differences between the ARINC-653 simulator and RTOS that is captured in previous sections. The benefit of the FACE Tools is a low cost starting point for the development of software intended for porting to an RTOS. Because they generate code encapsulating the software application, they provide common source code for the integrator which reduces integration effort.

Integration Perspective on Acquisition with FACE

The FACE Technical Standard defines rules for interfaces between components as well as architectural definitions about the role of a certain type of software component within a system. The JCA model provided a standardized method of defining interface data content and some methodology of how data will be exchanged (request, response, etc). Some interface details were not captured by the model and had to be defined separately (such as message timing).

Thus far, the integration issues encountered have been minor in nature. From the perspective of the system integrator, by applying the FACE Technical Standard the integration effort was improved over a traditional integration effort by providing independence of the software component from the underlying hardware. Corrections to the FACE tools, availability of a FACE Conformant operating system, and clear description of transport mechanism fields would further improve the integration process.

At the time of the writing of this paper the system integration and testing effort is ongoing and more lessons learned will be reported in the JCA Demo Final Report and in the briefing to the AHS Forum.

FUNCTIONAL TESTING OUTSIDE STANDARD

The FACE Technical Standard standardizes the interfaces for data sent from two different software pieces. Rapid development of the DCFM and RVC software in a high risk environment makes it imperative to perform functional testing early. As the FACE Technical Standard was the highest risk item on this project, Honeywell created a wrapper and software APIs to abstract away the FACE layer and allowed the DCFM software to live within a custom wrapper. This opened the way for the FACE auto-generated software for early integration. Figure 10 shows the architecture of the Honeywell DCFM and RVC.

The bottom of Figure 12 shows the Simulated TS which was implemented by Honeywell. Note that the MIS team developed and used its own TS. Choosing these common data structures allowed the system to be simplified. Honeywell was able to take an RVC structure and copy it over to the DCFM. The converse was true in taking a DCFM structure and copying it over to the RVC. Using these common data structures permitted a simulated TS to be written in a few days that allowed full operation outside of the FACE Standard Interfaces.

In addition to a working system, a system was created that was fully operational and could be compiled in Microsoft® Visual Studio®, giving Honeywell additional features such as debugging. On a Microsoft Windows® development system full end to end testing of the system was achieved. ARINC653 testing was not performed until testing in the development environment passed. This became crucial in the DCFM algorithm testing using the simulated TS to record data from the RVC and playing the data back to the DCFM. This allowed for precise repeatability required for algorithm testing.

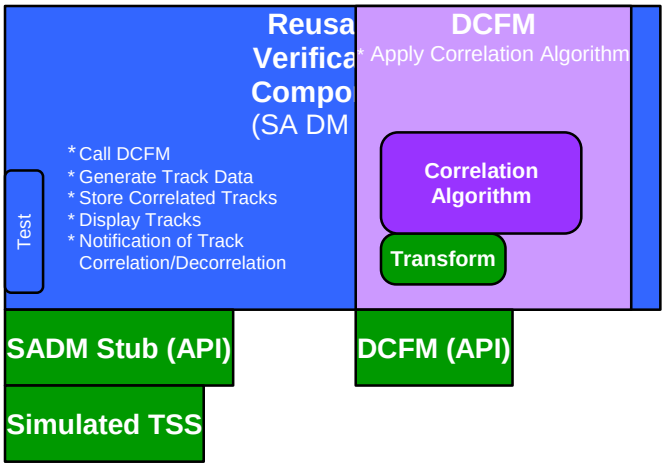

Figure 12 – Simulated TS

This approach worked well for the Honeywell team. The additional time during system design reduced schedule later on in the program. It also helped find integration issues early, and allowed for rapid debugging of individual UoP logic.

CONCLUSIONS AND RECOMMENDATIONS

The JCA Demonstration was an experiment between industry and the US Army to find and uncover issues with the procurement of FACE Standard components/UoPs. The JCA Demonstration has provided an excellent opportunity to test the procurement of a UoP as well as the concept of portability to support the ideas of the FACE Standard, and the JCA Reference Architecture.

As with any software procurement, detailed requirements about the interface must be provided. These include detailed requirements about system behaviors (e.g. latency, periods, time allocation, memory allocation, etc.) and functional coherency (e.g. interaction with other components, sequence diagrams, number of instantiations, 32-bit vs. 64-bit environments, etc.).

There were many successes and lessons learned from the JCA demonstration program which include the following:

1. Use of the FACE modeling tools enabled easy insertion and connection of the UoP into a system architecture.

2. Auto generation of TS layer interface between UoPs and health monitor functions saved time and reduced scope.

3. Use of the FACE Standard and data model added minimal additional integration and test time

4. Published verification matrix and tools allowed unambiguous verification of a FACE component.

5. Experience and knowledge in the FACE eco-system was gained.

6. Lessons learned and recommended Data Model changes were submitted to the FACE Technical Working Group for consideration for the FACE Technical Standard Edition 3.0.

7. FACE tools (CTS, and SDK/IDK tools) issues were indentified for the tool provider.

Thus, while there are advantages of utilizing the FACE Technical Standard and JCA standard in a model based acquisition approach, there are still system integration performance considerations that must be addressed. Also, analysis of requirements for completeness, and considerations in areas such as timing, data staleness, and data storage must be made prior to component implementation and system integration for an application involving situational awareness. While having information in a model can clarify expectations, it does not magically resolve requirements shortcomings.

The JCA Demo is still ongoing, additional lessons learned and issues around integration and testing will certainly surface. From a Honeywell perspective the Army's approach to JCA and model based acquisition was valid, the majority of the information needed for system design and integration were sufficient. As of writing this paper integration has been incredibly smooth with only minor issues on the systems integration side unrelated to the DCFM UoP software. No software changes to the DCFM have been requested since first delivery. Additional areas of the FACE Standard could stand to gain further insight with additional demo activities. It is recommended that further demonstrations such as with JCA Demo are executed to improve the FACE Standard and JCA reference architecture.

REFERENCES

[1] Boehm, B.W. *Software Engineering Economics.*: Prentice Hall, 1981.

[2] Galin, D. *Software Quality Assurance: From Theory to Implementation.* s.l. : Pearson/Addison-Wesley , 2004.

[3] NIST Planning Report 02-3, The Economic Impacts of Inadequate Infrastructure for Software Testing," May 2002.

[4] NAVAIR Public Release 2013-149 via The Open Group, "Technical Standard for Future Airborne Capability Environment (FACE™), Edition 2.0", https://www2.opengroup.org/ogsys/catalog/C137

[5] Department of the Army, Army Contracting Command, "Market Research to Refine the composition, data content, and acquisition approach for the Joint Common Architecture (JCA) Demonstration project as part of the Joint Multi-Role Technology Demonstrator (JMR TD) Phase 2 Program", Solicitation Number W911W6-13-R-0008, Location ACC-RSA-AATD-(SPS), Posted Date: June 5, 2013

[6] Department of the Army, Army Contracting Command, "A Joint Multi-Role Technology Demonstrator (JMR TD) Joint Common Architecture Demonstration (JCA Demo) Broad Agency Announcement (BAA)", Solicitation Number W911W614R0002, Location ACC-RSA-AATD-(SPS), Posted Date: January 27, 2014

[7] ARINC Standard, "Avionics Application Software Standard Interface: ARINC Specification 653 Part 0", June 2013

[8] IEEE, "IEEE Standard 1003.1-2008 - Standard for Information Technology - Portable Operating System Interface (POSIX(R))", 2008