Joint Common Architecture Demonstration Lessons Learned – Sikorsky/Boeing Perspective

Scott Wigginton
Computer Engineering
US Army AATD
Ft. Eustis, VA, USA

Thomas DuBois
Technical Fellow
The Boeing Company
Ridley Park, PA, USA

William Kinahan
Technical Fellow
Sikorsky Aircraft
Stratford, CT, USA

Andrew Bereson
Assoc. Technical Fellow
The Boeing Company
Seattle, WA, USA

ABSTRACT

The Joint Common Architecture (JCA) Demonstration project exercised the Future Airborne Capability Environment (FACE™) Technical Standard and tools as well as the JCA model-driven approach to gather lessons learned and mature the approaches. Sikorsky/Boeing was selected as one of two vendors to develop a software component and its reusable test component to be integrated onto multiple undisclosed Operating Environments (OEs). The model-driven approach was partially successful but required significantly more detail than was initially provided by the Government. The FACE Standard was effective in enabling portability of the software and the test component onto each OE. FACE Verification was pursued and process improvements were identified. Application of the FACE Transport Services did not introduce timing latencies that would impact safety-critical applications. The quality of the lessons learned throughout this effort highlight the importance of demonstrations involving independent teams in order to mature these approaches.

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 defense budget is shrinking. The United States Department of Defense (DoD) acquisition community must find new ways to procure systems that can affordably address these development and sustainment challenges.

In 2010, the DoD partnered with industry and The Open Group® 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. The FACE Technical Standard (Ref. 4) defines software computing environment architectures and interfaces intended for the development of portable software applications targeted for general-purpose, 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 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.

JCA Demonstration Project

The US Army established the JCA Demonstration (JCA Demo) project to exercise the FACE standard and 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 controlled in order to maintain integrity of the experiment to the maximum extent possible.

Presented at the AHS 71st Annual Forum, Virginia Beach, Virginia, May 5–7, 2015. Copyright © 2015 by the American Helicopter Society International, Inc. All rights reserved.

FACE™ and The Open Group® are trademarks of The Open Group.

The Modular Integrated Survivability (MIS) system in the Aviation Systems Integration Facility (ASIF) on Redstone Arsenal, AL was selected as the target system with the MIS team functioning as integrator for JCA Demo. The MIS system contains four OEs, multiple sensors, multi-function displays, and software built to the FACE standard. Based on this system JCA Demo sought to procure a Data Correlation and Fusion Manager (DCFM) software component. 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 selected OE. Composition of the MIS system was withheld from DCFM developers until after software delivery.

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 FACE Data Model plus two behavior component interaction diagrams. A minimal set of functional requirements were provided for demonstration purposes only. Processes, tools, and lessons learned were more important than component performance.

A Reusable Verification Component (RVC) was 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 ensured compatibility with host hardware and provided verification of the platform integration as life-cycle changes are made to each OE.

In addition to software components the Government sought verification of Vanderbilt University's Institute for Software Integrated Systems (ISIS) candidate FACE tools (Ref. 5). The FACE tools include candidate versions of a FACE Conformance Test Suite, FACE Modeling Tools, and a FACE Interface Definition Language (IDL) Compiler.

Finally, the DCFM was evaluated by the ASIF's FACE Verification Authority for conformance to the FACE Technical Standard edition 2.0. The FACE Shared Data Model was not published in time for its use in JCA Demo, so it was known that those requirements would fail verification.

Solicitation

A representative model-driven acquisition approach was followed for JCA Demo. A draft version of the FACE Contract Guide was followed to generate the solicitation language. A Request for Information (RFI) sought comments on the approach and technical content of the planned solicitation (Ref. 6). 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. 7).

Sikorsky Aircraft Corporation and The Boeing Company (Sikorsky/Boeing) partnered for JCA Demo and were one of two teams selected to enter into a Technology Investment Agreement with the US Army. The scope of the effort was to develop 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.

SIKORSKY/BOEING APPROACH

Sikorsky/Boeing proposed a basic technical approach to directly address the JCA Demo program goals and objectives. They also proposed two internally funded activities designed to provide additional lessons learned in line with the overarching goals of the JCA.

Basic Technical Approach

Figure 1 depicts the overall technical approach. Fusion functionality is provided by a core algorithm, known as Cohesion, that has been deployed on several production defense programs (Ref. 8). A software interface layer that conforms to the FACE standard and the provided DCFM data model was added to Cohesion. The candidate FACE IDL Compiler was used to generate portions of the software from the provided data model.

The RVC was developed as a FACE UoP that emulates the Situational Awareness Data Manager (SADM) interface to exercise DCFM functionality. A series of test cases were developed to show how the resulting software meets the performance requirements of the DCFM. The RVC was developed using a model-based tool that auto-generated much of the RVC software using models of the test cases.

FACE verification evidence was provided to the Army Verification Authority (VA) as a preliminary package during early development and as an initial package with the software delivered for integration. Conformance verification evidence included results from running the candidate FACE Conformance Test Suite, responses to applicable sections of the FACE conformance verification checklist, and additional artifacts that require manual inspections. Several iterations of the initial package were required to address VA concerns. The final assessment was not completed at the time when this paper was written.

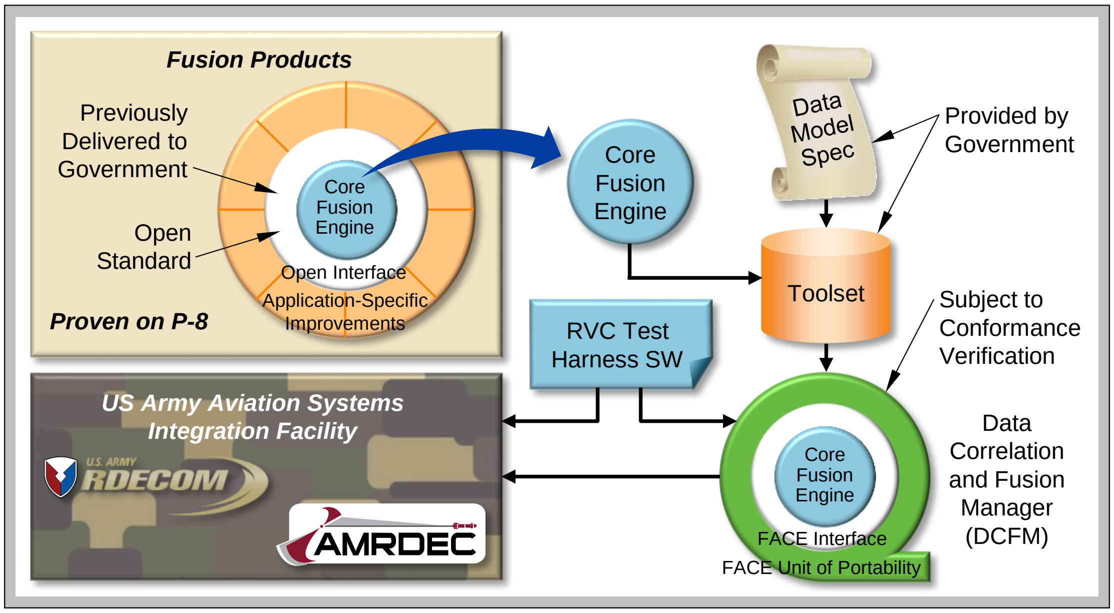

Figure 1. Overview of JCA Demo Basic Technical Approach

One of the most important activities associated with the JCA Demo technical approach was the collection, documentation, and disposition of lessons learned. Example types of lessons learned include application of the FACE standard, FACE tools, structure of the DCFM data model, interpretation of conformance requirements, and use of the FACE Library Portal. In many cases recommendations were either immediately addressed such as updates to the FACE Tools and DCFM data model, or were provided to the FACE Consortium. Other lessons learned were consolidated with all efforts relevant to JCA Demo for disposition by the Government to inform future research efforts and identify areas in need of further development.

DCFM on three additional OEs

The first additional activity performed by Sikorsky/Boeing integrated the DCFM and the RVC on three additional OEs. Figure 2 illustrates the reuse approach for this additional activity.

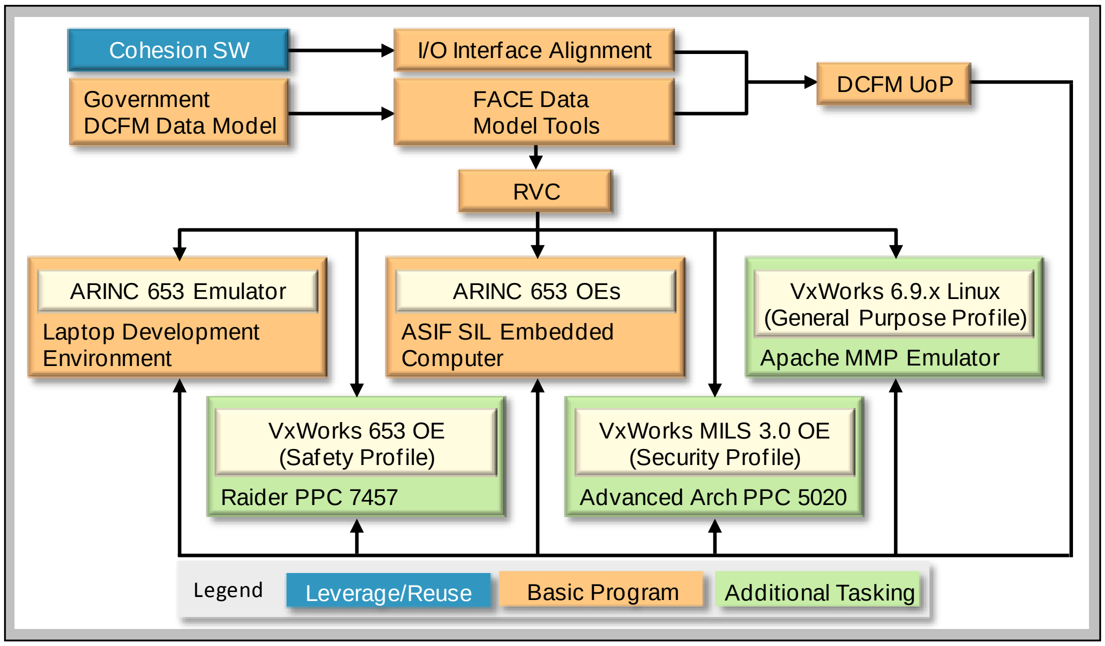

Figure 2. DCFM Reuse on Additional Rotorcraft Operating Environments

The OEs represent current, emerging, and future rotorcraft as represented by the AH-64E Apache Networked Common Operating Real-time Environment (NCORE) (Ref. 9), S-97 Raider^{TM} Cockpit Remote Processing Unit (RPU) (Ref. 10), and an advanced security-based OE using the Wind River® VxWorks® MILS 3.0 operating system hosted on a PowerPC 5020 dual-core processor.

Formation Flight UoP

The second activity developed a flight-critical application as a FACE UoP (Ref. 11). The objective of this activity was to assess the airworthiness implications of applying the FACE standard and identify concerns with performance resulting from the use of the Transport Services Segment (TSS). The principal result was a timing study comparing software processing latency with and without use of the FACE standard.

The selected application was Formation Flight, which uses positional information from multiple aircraft to direct auto-piloting functions on those aircraft to fly in formation along a flight route. The Formation Flight application interacts with the flight control system and is designed to operate within the timing limits established for those flight controls. It can be implemented as a pilot aid or with autonomous control of the aircraft. Formation Flight was chosen for its relevance and availability as a lab application with an established timing baseline. For use of the FACE standard, it was necessary to create a FACE Data Model, a test harness (performing the same role as the RVC for DCFM), and an auto-coding tool for interface with the control laws implemented in the lab for Formation Flight. The Formation Flight application was not taken through the FACE verification process. Initial results indicate that application of the FACE standard did not introduce latencies problematic for flight-critical applications. Expanded studies and analyses on this subject are under consideration for future technical papers.

DATA CORRELATION AND FUSION MANAGER (DCFM)

The DCFM developed by the Sikorsky/Boeing Team is based on Cohesion, a Boeing software product that performs data correlation and fusion. This product was designed to be portable between systems and it has been deployed on numerous Boeing platforms. It was straightforward to implement the DCFM based on this product and the development team was able to convert the Cohesion algorithm into a DCFM UoP in less than two months. It took a similar amount of time to construct the RVC.

Reuse of the Cohesion software as the DCFM required two main activities. The first was to align the Cohesion interface with the DCFM model. The second activity was to modify the software to satisfy the requirements of the FACE Technical Standard. Both of these activities were instructive regarding application of JCA and FACE standard.

Interface Alignment

The primary activity in the DCFM development involved aligning Cohesion's interface with the FACE Data Model. To do this, a wrapper was built to contain Cohesion and to attach it to the FACE TSS. This provided a thin layer to translate TSS messages into the format required by Cohesion and then to take the responses from Cohesion and convert them back to messages sent over the TSS. The wrapper also provided Cohesion with its configuration and started the Cohesion software.

Although the mechanics of constructing this interface layer was straightforward the activity uncovered several issues. Some of these resulted in the Army updating the DCFM data model while others required the development team to make assumptions about the data and data flows that were not specified in the model. These assumptions are documented in the lessons learned.

One set of issues related to the mechanics of the data model construction. In the DCFM data model complex types were constructed by combining fundamental types. This led to duplication of fields such as separate identifications for substructures of an entity message, and duplication of latitude and longitude in both entity position and region of uncertainty. Some of these required the Sikorsky/Boeing Team to make assumptions regarding which were the authoritative sources. While this was a simple matter to address it highlighted challenges in building complex models.

Another set of issues revolved around ambiguity in the model itself. Although the fields were generally well defined, the model did not indicate how these fields should be used. In some cases, the intended meaning of fields was unclear, such as in the aforementioned case of duplicated position. Another involved the track source field where the Sikorsky/Boeing Team and the Army had different and incompatible interpretations that required resolution. To avoid these cases in the future additional clarity is required on the meaning of each of the fields in the data model.

An additional observation is that behavioral aspects of the data require greater specification. This included items such as the track data delivery rates, number of tracks delivered per frame, and allowable staleness of sensor data. Since the DCFM and RVC were development activities,

____

Raider^{TM} is a trademark of the Sikorsky Aircraft Corporation.

Wind River® and VxWorks® are trademarks of Wind River Systems, Inc.

these issues were resolved by making reasonable assumptions. All of these issues indicate the importance of including a behavior model alongside the data model.

In summary, aligning the interface is not as simple as matching data model entities with corresponding software structures. When attempting to reuse legacy software, the semantic meanings of both data model entities and legacy software structures need to be fully understood. This understanding process can lead to post-delivery changes to the data model or required updates to the legacy software. Both of these implications need to be considered when attempting to reuse legacy software within the framework of a model-based open architecture such as the one defined in the FACE standard.

Cohesion Modification for FACE Requirements

The Cohesion software was designed for portability and has minimal Operating Environment (OE) dependencies. This facilitated alignment to the FACE Safety Profile. However, Cohesion used several C++ operations disallowed by the safety profile including use of streams and several disallowed POSIX calls. Another concern was regarding the use of exception handlers. The VA identified exception handlers as not allowed. However, there is some ambiguity regarding whether or not exception handling is allowed in the safety profile and we elected to not remove exception handling from Cohesion. This issue has been submitted to the FACE Standard Subcommittee/Conformance for resolution.

Verification Activities

An iterative and incremental process was used for interaction with the Army Verification Authority. The first increment, called Preliminary Verification Evidence, contained the DCFM source code, results from running DCFM through the Conformance Tool Suite, and additional supporting documentation, such as a Software Requirements Specification (SRS) and a Software Architecture Description Document (SADD).

The second increment, called Initial Verification Evidence, was provided to address feedback from the Army VA. This increment addressed expectations of verification evidence. It also contained results from all inspection tests not provided with the Preliminary Verification Evidence, and addressed noted deficiencies.

The third increment, called Final Verification Evidence addressed most concerns from VA assessments of the Initial Verification Evidence. Some were interpretive in nature and resulted in clarifications provided in revised conformance statements and verification matrices. For these items feedback was provided to the VA so that process improvements could be made to avoid such confusion in the future. Beyond differences in interpretation, operations disallowed by the safety profile, as discussed above, were addressed through updates to the DCFM source code. The approach to exception handling was not completely resolved due to lack of clarity on the use of exceptions in the Safety Profile. In coordination with the VA, the Sikorsky/Boeing Team utilized the compiler specific functions file supported by the conformance suite to work around errors reported by the conformance tool while this topic gets resolved.

Software Reuse

While JCA Demo focused on modularity, portability, and interchangeability of software components, working through the process provided insight into the other aspects of software reuse which are paramount for JCA. There are several aspects of software reuse to consider:

- Reuse of existing capability by aligning to a FACE conformant interface.
- Reuse of FACE conformant UoPs across systems.
- Substitution of one FACE conformant UoP with another that meets the same functional requirements.

Reuse of Existing Software

The DCFM interface was specified as a FACE Data Model and a series of component interaction diagrams. While the data model was sufficient to perform basic correlations for a proof of concept demonstration, a production application would require a more complex and detailed data model to meet performance requirements. The FACE Technical Standard definitely facilitated this process and significantly reduced the cost generally associated with reuse.

As discussed previously, preexisting software not built to the FACE standard will require modification for conformance. The expectation is that these would be one time changes that would produce a reusable FACE conformant UoP. The changes could impact software pedigree and qualification, impacting the cost of reuse. The FACE Business Guide (Ref. 12) can be used to determine the business case.

Portability and Reuse across Systems

For additional portability and reuse evaluations, the DCFM and RVC were hosted on three additional OEs, as depicted in Figure 2. Each OE used a different FACE transport service to interact with different lower-level middleware and operating systems. The transport service calls were identical, as specified by the FACE standard, but recompilation on each OE was necessary to link in each of the different transport service implementations. Although the DCFM proved to be portable, changes were required in the transport service itself. The software successfully ran on all three OEs, which gave the team confidence that it would also run successfully on the unknown OEs at the ASIF/MIS lab.

Substitution of One UoP for Another

Sikorsky/Boeing did not attempt to substitute one DCFM UoP for another but approached the DCFM and RVC with the assumption that either could be replaced by a component from another vendor built to the same performance requirements, data model, and FACE specification.

Anticipated challenges are associated with limitations to the model-based approach to interface specification. While the model provides definition for the data to be transferred between UoPs, the FACE Data Model does not currently support modeling interactions between components. This is an architectural problem primarily impacting UoPs that need to maintain state. For example, the model failed to reveal how the data could be affected or used by other UoPs. This can lead to incompatible UoPs which all conform to the same set of functional requirements.

This last example is an artifact of the requirements as much as the data model. Nonetheless, it reveals the complexity of specifying a model that may be interpreted differently by all potential vendors as well as the need to provide behavioral models.

REUSABLE VERIFICATION COMPONENT

Portability of FACE components implies use on multiple avionics systems likely with unique FACE environments. Across these environments, there will likely be variations in processors, operating systems, and network communications, as well as variation in deployment of components to environments. Component software will likely have to undergo a compilation and build process, and be configured according to the developer's instruction. Experience indicates that this rarely proceeds without error. Therefore, it is desirable for the component developer to provide a means of verifying the item in the installed environment to reduce risk prior to full integration. In addition to the DCFM, the JCA Demo project required a Reusable Verification Component (RVC) for this purpose.

There are a number of considerations when selecting an approach for implementation of an RVC. First, the target resident portion of the RVC should not rely on FACE infrastructure features beyond that required by the delivered component as this may impose additional burden on the integrator. For instance, the integrator might have a limited implementation of I/O services addressing only a few bus types or may not have the logging service available. The capabilities of the system that was to be the target of the DCFM were not known.

Another consideration is what tools are available to support using FACE Data Models and generation of code. For the JCA Demo program, use of the FACE tools was a requirement. These tools allow for editing data models, construction of a system model, generation of the data types dictated by the data model, and generation of a simple ARINC-653 based transport service for communication between UoPs in accordance with the system model.

The method chosen by the Sikorsky/Boeing Team was to implement the RVC as its own Unit of Portability (UoP) that would communicate through TSS to the DCFM (see Figure 3). Figure 4 illustrates the strategy and shows the data model elements as well as the software code automatically generated by the Ecosystem tools. Like the DCFM, the resulting RVC is portable to any FACE OE. This was demonstrated on the three OEs that were previously described.

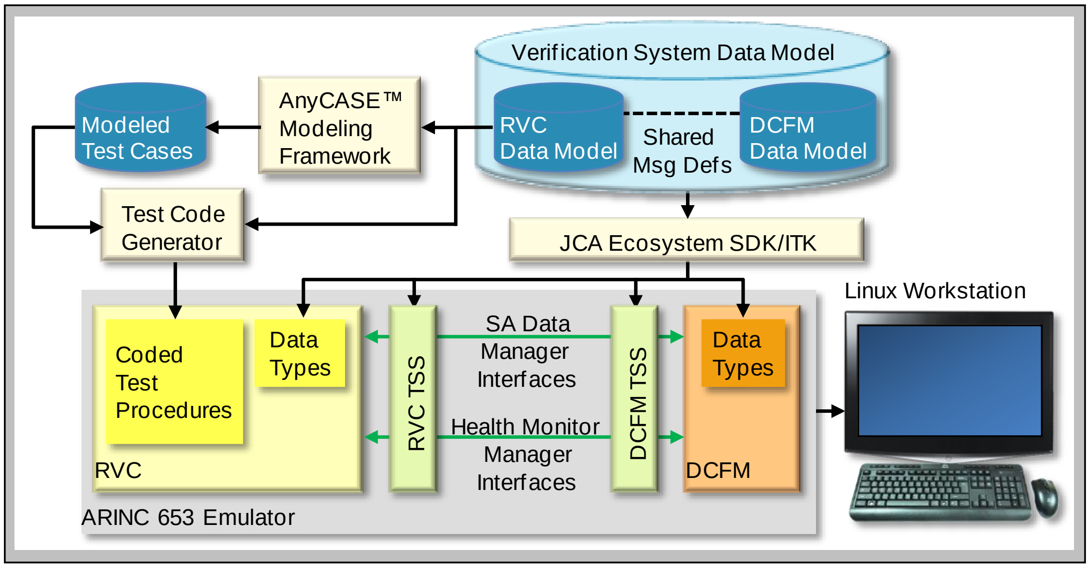

Figure 3. Reusable Verification Component (RVC) Diagram

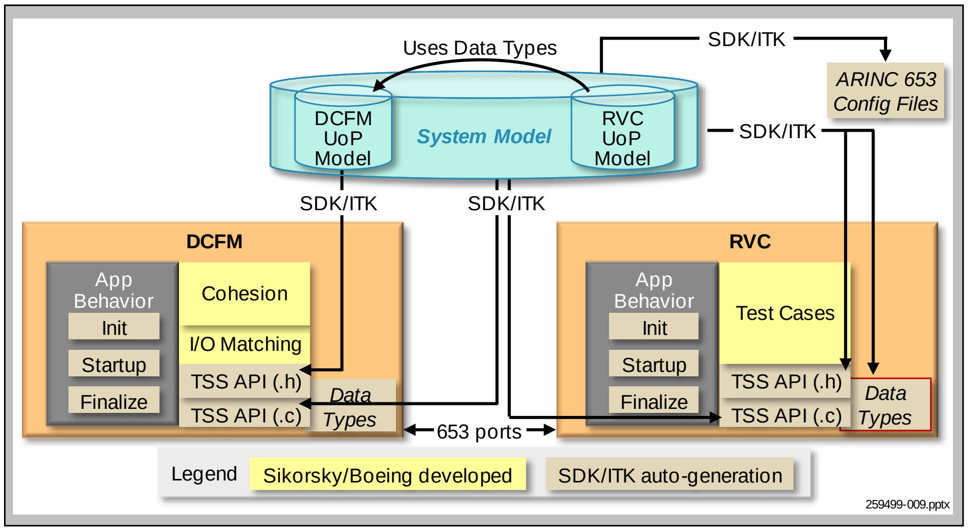

Figure 4. RVC Development and Usage

Technical Approach for RVC

The approach used for the RVC leverages concepts embodied in the philosophy underlying FACE data modeling to formulate a design pattern that can be employed across a wide range of test applications. When building a FACE Data Model, the UoP developer first identifies the set of entities that are the subjects of the function implemented by the UoP for which communications are planned. Properties are identified and added to the entities so that all data that will be received or published by the UoP is defined. Once the entities have been defined, the UoP developer creates a set of views, or messages, and projects data from the entities into the views thereby defining their content. For example, the DCFM UoP data model defines an entity for a source track and gives it properties of position and position quality. Similarly, entities are created for correlated tracks and the air platform itself. The model projects from these entities into views (messages) for source tracks, correlated tracks, and platform position.

The RVC uses the entities in the DCFM data model to create a set of objects which it will use to hold the operational data context of the running DCFM. RVC test cases operate on the set of entity instances. Data to be input to the DCFM is first written into the entity instances. This data is then extracted into the proper views and sent to the DCFM. Likewise, as data is received from the DCFM, it is unpacked from the views into the entities. This approach decouples the test cases from the packaging of data into message views and results, and yields a product that is more resilient to change. The internal architecture of the RVC is depicted in Figure 5.

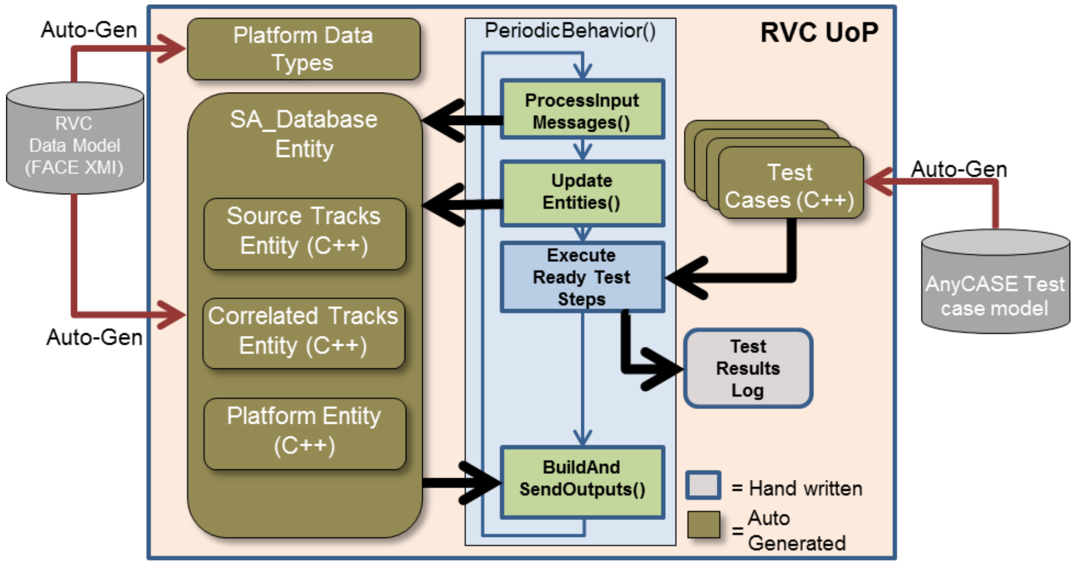

Figure 5. Internal Architecture of the Reusable Verification Component (RVC)

Testing

Testing a correlation function relies on observing the behavior of tactical objects and their relationships across time. Movement of the tactical elements and of the platform could be accomplished by periodically updating the associated entities from the test cases, but that would result in long tests which would take more effort to generate and maintain. In the case of the RVC, a simple model of movement was implemented in which items were moved along a trajectory over time. To support this approach the RVC data model was enhanced to capture the movement parameters which were then manipulated by test cases directly. By using the data model to describe these parameters the FACE IDL Compiler automatically created the data types for the supplemental data as well.

The essential operation of the RVC is generic and can be implemented by a set of base classes that have no dependency on the function being tested and can therefore be used on a wide range of applications (see Figure 6). As illustrated by the figure, application specific behavior need only be added to a single class via overriding a set of member functions. The set of classes forms the basis of a reusable test framework from which application specific reusable test software can be developed. In the case of the RVC for the DCFM, the only piece of software that needs to be written to implement the application specific processing is the RVC Test Class subroutine. The customizations include providing routines to perform initialization, processing of input messages, updating entity models, and to send output messages. Aside from initialization, all of these functions are called periodically.

Modeled Test Cases

To use the framework described above, a set of test case data structures with the instructions for the tests to be performed needs to be populated. Building each of the test cases by hand and counting on the compiler to find coding errors would typically be a labor intensive process. On the JCA demo program, a modeling tool framework called AnyCASE^{TM} was modified to support capture of test cases and automatic generation of the test case data structures. AnyCASE provides user friendly mechanisms for defining test cases and the steps required to stimulate the inputs to and verify the outputs from the DCFM. To speed up implementation of support for the FACE standard in AnyCASE, an existing application was used to automatically translate the Data Model Meta-model into source code for the modeling tool.

The tool supports importing and browsing of FACE Data Models in order to provide drag and drop referencing of data model entity characteristics within test steps. All modeled test cases were checked for consistency as they were constructed. While not required for code generation, AnyCASE was also enhanced to support tracing of test cases to requirements to ensure that all requirements were addressed by at least one test case.

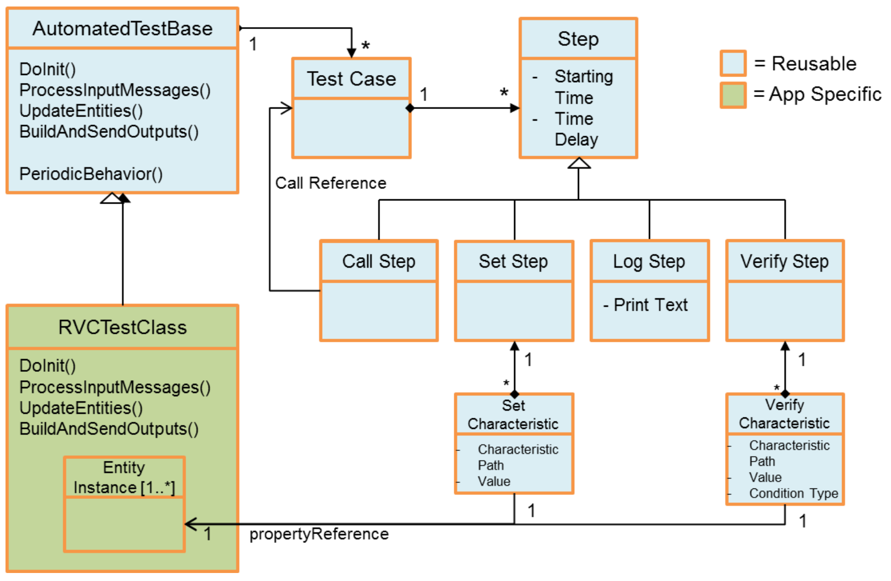

Figure 6. Test Case Model for RVC Construction

AnyCASE^{TM} is a trademark of the Sikorsky Aircraft Corporation.

Using a model-based approach proved to be an effective means to construct the RVC code. A second benefit was that the test cases were captured in a platform and language neutral format leaving open the possibility of generating automated test software for alternative environments. For example, to implement the RVC in Python on a host machine with a compatible transport service implementation, the same test cases could be used with only the code generator having to change. In this way, modeling of test cases supports another approach to reuse and portability.

MIS INTEGRATION EFFORT

The MIS team established two Operating Environments (OEs) for the purposes of JCA Demo. OE #1 consisted of a PowerPC 7447 dual-processor with 1GB of RAM and 64GB of storage operating Wind River VxWorks 653 2.3. OE #2 consisted of an Intel Xeon dual-core processor with 1GB of RAM and 4GB of storage operating LynuxWorks™ LynxOS®-178 2.3. As part of JCA Demo, information on the OEs was withheld from Sikorsky/Boeing. As previously discussed, the Sikorsky/Boeing Team had to make several assumptions due to insufficient detail included in the technical specification. Those assumptions were vetted with the MIS team to identify issues that would disrupt integration. For one issue, it was necessary for the MIS to make and change and release a new version of the data model.

ARINC-653 emulator

Upon delivery of the Sikorsky/Boeing DCFM and RVC, the MIS team first sought to reproduce the RVC test results using the simulated ARINC-653 environment provided with the FACE tools. The integration team encountered compilation difficulties with the provided software components related to details on the compiler versions, build management settings (CMake), and versions of standard C++ libraries. After resolving the compilation issues the RVC was successfully executed on the DCFM in the simulated environment.

Integration

The Sikorsky/Boeing DCFM was integrated on OE #1. During integration the DCFM source code required the following modifications. Some compiler issues similar to those found on the ARINC-653 emulator required changes to the build process. FACE tools used to assist development of the DCFM produced function calls specific to the ARINC-653 emulator that are disallowed by the FACE Safety profile. The integration team replaced the TSS provided by the FACE tools with an alternate TSS, where the API used certain function parameters in a different manner. These portability issues were corrected with minimal software effort.

The Sikorsky/Boeing team also encountered similar issues as described in the *Portability and Reuse across Systems* section. Formal system integration testing on OE #1 and DCFM integration on OE #2 were not completed by the writing of this paper due to an unrelated technical issue. Further findings will be provided during the presentation.

Overall the integration issues were minor in nature. By applying the FACE standard the integration effort was improved by providing independence from the platform and transport mechanism. Corrections to the FACE tool, availability of a FACE conformant operating system, and clear description of transport mechanism fields would further improve the integration process.

CONCLUSIONS

JCA Demo achieved its goal to exercise a model-driven acquisition approach utilizing the FACE Standard and tools. The effort demonstrated software portability by developing and integrating a software application and its reusable test component on multiple operating environments. JCA Demonstration produced several conclusions as follows:

- The FACE Standard provided the independence from the underlying platform and transport paradigm as shown by the transition of the DCFM and RVC UoPs to multiple OEs with minimal changes.

- The model-driven acquisition approach was partially successfully in reducing the impact to software rework by facilitating consistent interface definition and supporting generation of application and test software.

- Use of the FACE Data Model enabled the successful description of the composition of entities and their properties. The data model did not sufficiently describe the relationships between properties or the behaviors associated with the data.

- Legacy software designed to be agnostic of the OE is suitable for reuse as long as semantic aspects of the data elements are well defined.

- An extension to AnyCASE was created from the FACE Data Model Meta-model that enabled importing and browsing FACE Data Models. This approach could be applied to other tools to facilitate adoption of the FACE Standard.

- Implementing the reusable test component as a UoP was an effective approach for automated testing of application UoPs during development and test to

LynuxWorks^{TM} andLynxOS® are trademarks of LynuxWorks, Inc.

ensure proper operation on target platform after delivery.

- Early engagement with a FACE Verification Authority was beneficial by establishing expectations and interpretation of conformance requirements during development and verification planning.

- The FACE Tools enabled rapid evaluation of the FACE Standard.

- Use of the FACE standard did not introduce latencies that would impact timing requirements associated with safety-critical applications. Further studies are necessary to determine to what extent latencies could impact control loop timing requirements.

The quality of the lessons learned highlighted the importance of maturing the FACE standard and JCA concept through a representative acquisition process.

ACKNOWLEDGMENTS

The authors thank Alex Boydston for contributions to this paper. The authors express their gratitude to the Aviation Development Directorate (ADD) Joint Multi-role (JMR) project office and Sikorsky/Boeing JMR/FVL Team for jointly funding this effort. From the Sikorsky/Boeing Team, significant contributions to the technical work are attributable to Don Akers, Fernando Dones, John Glowa, Melissa Hofmann, Mark Joy, Jay Kallaher, Dave Liptak, Pat Michelangelo, Dave Phillips, Andrew Ruggeri, and John Sechrist. From the US Army, significant contributions to the technical work are attributable to Ashley Denny, Bill Tanner, Bob Duncan, Chris Kellow, and Steven Price.

REFERENCES

[1]Boehm, B.W. *Software Engineering Economics.* s.l. : 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]ISIS FACE Project, Candidate FACE Tools by ISIS, https://face.isis.vanderbilt.edu/redmine.

[6]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.

[7]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.

[8]Bereson, A. L., and Lobbia, R. N., "Efficient Track-to-Track Assignment Using Cluster Analysis", 9th International Conference on Information Fusion (FUSION 2006), Florence, Italy, July 10-13, 2006.

[9]Koontz, R., and Johnson, D., "Apache Mission Processor Software Architecture: Future Airborne Capability Environment (FACE™) Considerations", American Helicopter Society 70th Annual Forum Proceedings, Montreal, Canada, May 20-22, 2014.

[10]Helipress, "Military helicopters: the Sikorsky S-97 Raider takes another step toward the maiden flight", http://www.helipress.it/schede-361-military_helicopters_the_sikorsky_s_97_raider_takes_another_step_toward_the_maiden_flight_video, July 7, 2014.

[11]DuBois, T., Kinahan, W., and Dones, F., "Joint Common Architecture (JCA) Recommendations", American Helicopter Society 70th Annual Forum Proceedings, Montreal, Canada, May 20-22, 2014.

[12]NAVAIR Public Release 11-1200 via The Open Group, "FACE™ Business Guide, Version 1.1", http://www2.opengroup.org/ogsys/catalog/G115.