Harnessing the Richness of the FACE™ Technical Standard

ARCM Radio Control Platform-Specific Device Service – a Real-World Case Study using TES-SAVi MMOSA Processes

The Open Group FACE™ and SOSA™ U.S. Air Force Technical Interchange Meeting (TIM) Paper by:

Sean P. Mulholland, TES-I, TES-SAVi
William G. Tanner, TES-SAVi

September 2022

AIR FORCE LCMC

U.S. AIR FORCE

Table of Contents

Executive Summary

The FACE™ Consortium was established to develop a software Reference Architecture (RA) enabling the reuse of applications for safe and secure software intensive systems. However, the FACE RA is not enough on its own to develop reusable safe and secure systems across organizations. We believe one must leverage advanced Modular Open Systems Approach (MOSA), open technical standards, and advanced technical processes in a digital engineering environment.

This paper presents TES-SAVi's Model-based Modular Open Systems Approach (MMOSA), AWESUM® model-based tool suite and a case study for employing MMOSA to develop the U.S. Army's Aviation Radio Control Manager (ARCM) components to meet DO-178C DAL C and FACE Edition 3.1 conformance. The overall result is the realization of the promise of the FACE open standard goal for enabling rapid reusable UoC development and simplified integration of UoCs for building composable systems.

TES-SAVi Model-based Modular Open System Approach (MMOSA) Process

MMOSA Overview

MMOSA is a lifecycle process for cyber-physical systems development utilizing digital engineering concepts for implementing a MOSA with Agile and DevSecOps techniques in a manner such that the resulting system is qualifiable.

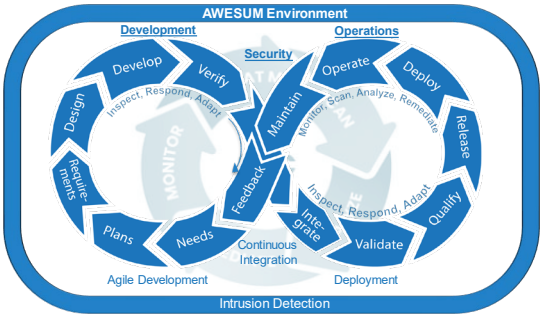

Figure 1: MMOSA Lifecycle Process

The figure above depicts the full systems development lifecycle continuum. The MMOSA Lifecycle Process was created to formalize a development process that meets the requirements of airworthy systems development. The MMOSA Lifecycle Process leverages Agile best practices for customer-focused development through working software iterations that evolve to meet the customer's needs. DevSecOps techniques are incorporated to improve the systems development lifecycle through automated development, verification, and deployment.

The key tenets of the MMOSA Lifecycle Process are:

- Automated support for the full systems development lifecycle continuum
- DevSecOps and Continuous Development/Continuous Integration (CI/CD)
- Multi-discipline design and management for single source of truth (SSOT)
    - Model maintenance and support for information (data) provenance
    - Holistic systems development processes
    - Open Standards adherence
    - Support Reference, Objective, and System Architectures which are leveraged as foundation for system requirements and design
- Top-down/Bottom-up approach
    - Top-down approach identifying needs, requirements, architecture, and design

- Bottom-up approach identifying complete specificity for external interfaces and identifying reusable capability interfaces and reusable components

- Needs Validation throughout the Process
- Certification/Qualification support for plans/processes, artifacts, and resultant system
- Unified Project Model: *all* data accessible/usable by *all* roles

There are eight (8) core MMOSA Models:

- Requirements
- Semantics
- Architecture
- Capability Interfaces
- Device/Component Interfaces Control Description
- Document Artifact Model
- Test Case/Procedure/Results Model
- Traceability Model

This paper focuses on one portion of the MMOSA process; the bottom-up approach identifying complete specificity for reusable capability interfaces utilizing the FACE data architecture to fully specify the semantics of the interfaces. Three of the eight core MMOSA models we will focus on are the Semantic data model utilizing the FACE Data Architecture, the Capability Interface Model and the Device/Component Interface Control Description Model.

MMOSA Capability Interface Model

The MMOSA Capability Interface Model (CIM) implements the patent "Capability Driven Architecture (CDA)"[9] in that it provides a bottom-up interface design approach for developing reusable abstract interfaces that hide the details of an external device or component while providing full control of the device, and rapid integration of different devices or components into different systems and platforms.

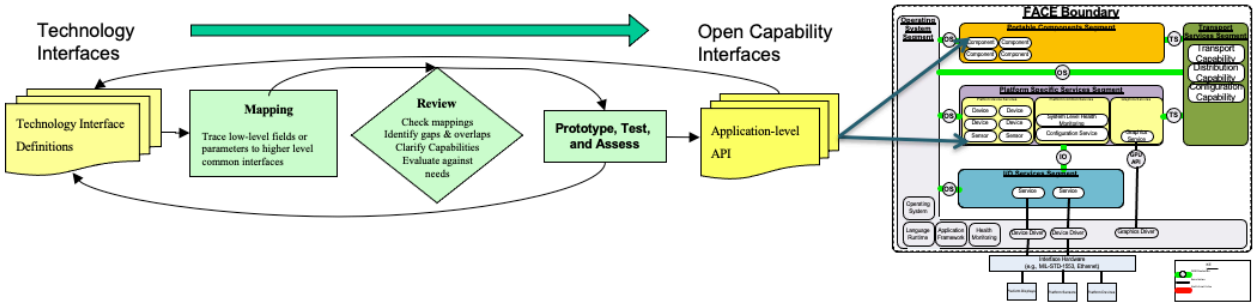

Figure 2 CDA Process for Defining FACE Conformant Capability Interfaces

The CDA process, depicted above, is a combination of top-down and bottom-up approaches where the input to the process is the system requirements and the low-level interface documents. These low-level documents are imported into the AWESUM tool suite. The documents reside within the SUM database. The process involves abstracting the interfaces into a non-proprietary top-down commonality-based design. The high-level system requirements are also entered into the toolset. The remaining process fills in the gaps between the system requirements and the low-level ICDs.

The functional abstraction analysis process is iterative in nature. It is used to define standard interfaces and categorize the underlying control code for the capability.

The primary idea behind the process is that by documenting the detailed interfaces, bubbling those interfaces up into their primary functions, and then bubbling up those functions into capabilities provides a process by which a complete capability interface can be defined.

The input into the CDA process is low-level Interface Control Documents (ICDs), Application Programming Interfaces, SNMP MIBs, or other like interface definitions. These inputs are further specified by applying the FACE Data Architecture Conceptual Data Model Entity/Association Perspective, and Observation Perspective to create a detailed model of the external interfaces.

This method of Functional Abstraction, integrated with the FACE Data Architecture allows for the creation of abstract interfaces to be created that are linked to external interfaces of the same data domain, such as communications. The combination of the Capability Interfaces linked to the external interfaces promotes the ability to rapidly integrate common and dissimilar capabilities and devices on dissimilar platforms.

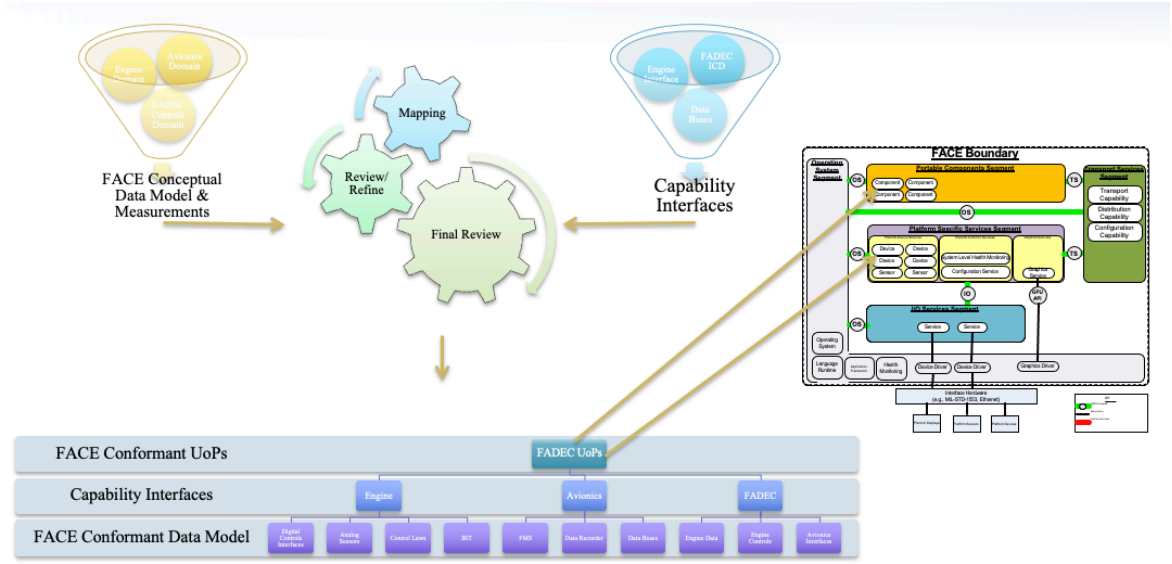

Figure 4: Capability Interface Development with Integrated FACE CDM & LDM

Once the Capability Interfaces are fully defined and refined through the CDA iterative process, the following can be auto generated in whole or in part from the Systems Unified Model:

- Interface requirements
- Interface Design

- FACE Logical and Platform Model Entities, Associations, and Queries
- FACE UoP (PCS & PSS) Models and Templates
- FACE TS code
- Capability Interface to/from External Interface code
- Test Cases, Test Procedures and Test Results from procedure execution
- Traceability Model
- Documentation Artifacts

The following sections describe how the Capability Interfaces Functional Specifications are defined and developed.

Capability Interface Function Specification

There are three types of Capability Interface functions based on their usage: getter, setter, and command/control. Each Capability Interface function also has a queue size, zero or more parameters (input, output, input/output) and return value parameters. In the MMOSA process, each parameter is specified with Name, Usage, Type, Measurement and Conceptual Semantics/Context:

Table 1: MMOSA Capability Parameter Attributes

Name	Usage	Type	Measurement	Semantic Anchor	Semantic Path
[name]	Input				
[name]	Output				
[name]	Input/Output				
[name]	Return Value				

Parameter Attributes

- The Type Parameter attribute is the data type of the parameter, which specifies the physical type, size, and sign of the data, *e.g.*, tBool, tUInt8, *etc*.
- The Measurement Parameter attribute is a reference to the FACE Logical Measurement semantics for the parameter. For example, a parameter may reference a Measurement from the FACE Shared Data Model:

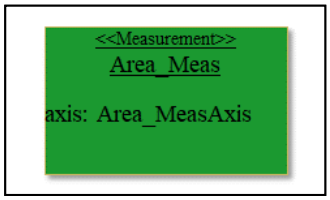

Figure 5: FACE Logical Measurement

- The Semantic Anchor Parameter attribute is a reference to the FACE Conceptual Entity or Association that specifies the starting context of the semantics, or meaning, of the parameter. This specification (and the Semantic Path described subsequently) are akin to the FACE 2.1 notion of Characteristic Projections. For example, a parameter describing a ground area within the context of a mission may have a Conceptual Data Model with the following Entities, the Mission Entity being the Semantic Anchor:

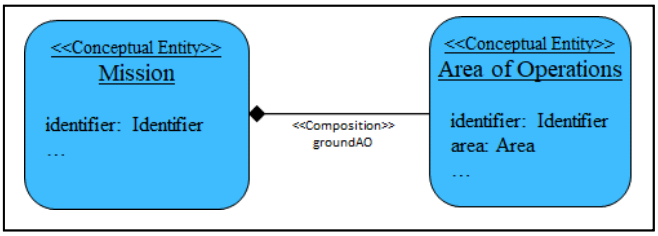

Figure 6: Example Conceptual Data Model Snippet

- The Semantic Path Parameter attribute is the textual path (in dot notation form) from the Semantic Anchor to a leaf Characteristic Composition, typed to an Observable, that provides the complete semantics and context of the parameter. For example, the semantic path attribute for "ground area" parameter within the context of the mission, and with a semantic anchor of the Mission Entity would be:

  .groundAO.area

FACE Data Model Generation

With such a concise specification of the Capability Interface Model, which includes referencing the Conceptual and Measurement Semantics provided by the FACE USM or DSDM, the AWESUM Tool Suite can generate the requisite FACE Conceptual, Logical, Platform, and UoP model elements[1] for a FACE Conformant UoC.

[1] This includes Entities, Associations, Characteristic Compositions and Participants, Realizations, Measurements and their cohort, Platform types, Queries, Templates, and UoP model elements – in short, the entire set of FACE data model elements for a FACE Conformant UoC.

Radio Control Use Case

FACE Diagram Context

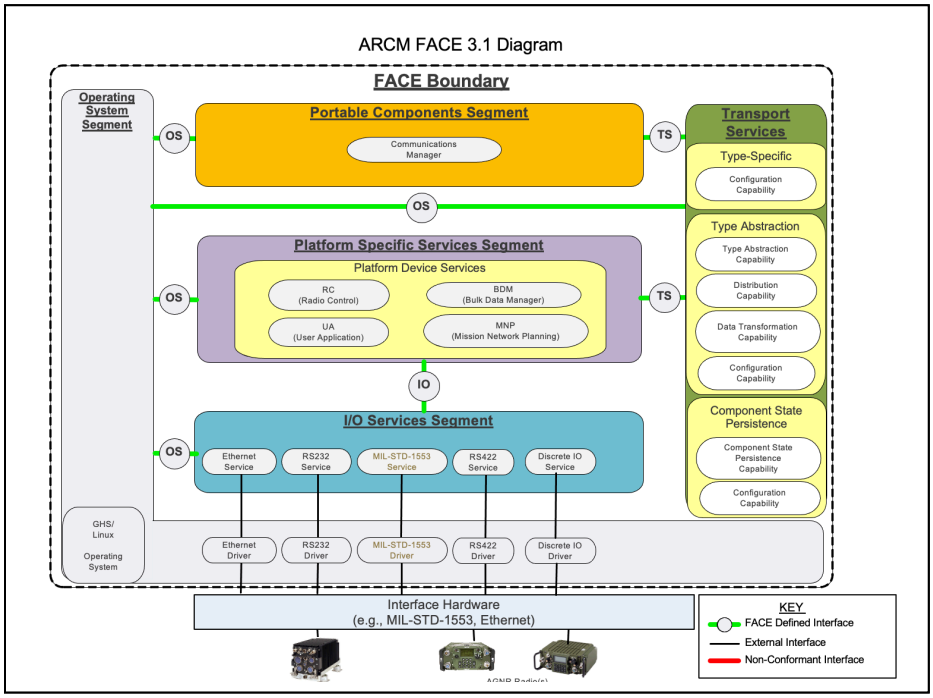

Figure 7: Radio Control FACE Diagram

Radio Control Capability

The Radio Control Capability accommodates many different types of radios, each of which have different and often disparate functionality, messaging sets and messaging paradigms. Particularly challenging are differences in radio messaging paradigms. For example, some radios provide responses regarding the status of commands sent to them (*i.e.,* the interface is a traditional command/response paradigm). However, other radios do not provide responses to commands, but instead require a separate request for command status inquiry after an elapsed time.

Capability Interface functions developed per the CDA process are generalized and normalized while at the same time aligning and supporting the differing aspects of multiple radios. Individual radio ICDs must also be supported, unchanged, in their entirety[2]. In support of such a wide variety in radio messaging paradigms,

[2] In addition to satisfying requirements for test coverage, customers commonly require that each interface be accessible regardless of generalized capability interfaces.

the Radio Control Platform Specific Device Service (PSDS) makes use of a significant portion of the MMOSA Capability Interface function set.

Radio Control Functions

Each Radio Control function falls into one of the "usage types" in the MMOSA Capability Interface function set:

- Getter Functions – Radio getter functions provide the ability to get radio data such as radio settings, configuration, status, and mode. For example, there are getter functions for radio volume, squelch, current data rate, transmit power, receive frequency, radio state, communication mode, *etc*. Radio getter functions usually adhere to the following naming and signature:

  ```
  get[data item name] (output:[data item name],output:[data item source])
  ```

  For example:

  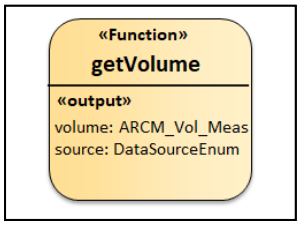

  Figure 8: Example Getter Function – getVolume

  Radio Getter Functions output the specified data item and the source of the data item value. The source is typically the actual radio, or the value stored by the Radio Capability.

- Setter Functions – Radio setter functions provide the ability to set radio data. Most of the setter functions are inverses of getter functions. Setter functions set radio data, such as: volume, squelch, data rate, and so on. Radio setter functions usually have the following naming and signature:

  ```
  set[data item name] (input:[data item name],return:[success indication])
  ```

  For example:

  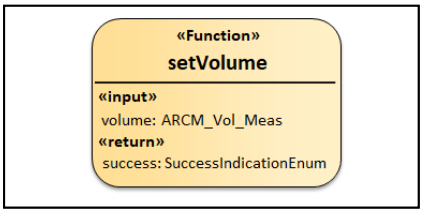

  Figure 9: Example Setter Function: setVolume

  Radio Setter Functions take as a single input parameter the desired data item value, in this case the desired radio volume. The return value, success, indicates whether the "set" operation was successful or unsuccessful.

- Command and Control Functions

- Action Functions – these are functions that command the radio to perform a specific action, such as power up/down, zeroize
- Selection Functions – these are functions that select a configuration among several configurations, for example selecting a preset from a set of presets, loading a specific radio configuration (a group of settings) from several radio configurations, or selecting a specific antenna to use

Action and Selection Functions usually have the following naming and signature which takes a single input parameter that more specifically commands/controls, with a return status parameter:

```
[command] (input: [command type], return: [command status])
```

For example:

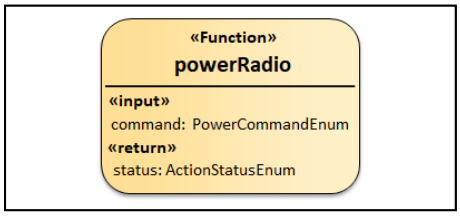

Figure 10: Example Action Function: powerRadio

In the example, the specified command is either POWER_ON or POWER_OFF, and the return value is (among others): ACTION_SUCCESSFUL, INVALID_PARAMETER, etc.

Another example that performs an antenna selection:

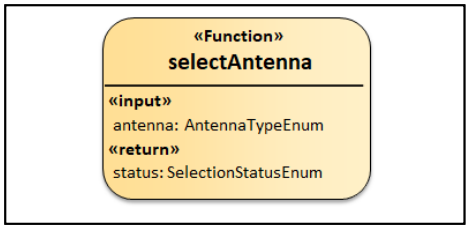

Figure 11: Example Selection Function: selectAntenna

In the example, the selection is either HORIZON or ZENITH, and the return value is (among others): SELECTION_SUCCESSFUL, INVALID_MODE, INVALID_PARAMETER, etc.

Note that in ARCM, each of these abstracted capability functions were developed with Device Interface Control Description (ICDs) input and abstracted via the CDA process. This resulted in a non-proprietary open modular interface descriptions for the Communications Domain.

While these examples don't show all variants of Radio Control Capability Interface functions (permutations of parameters, queue sizes, etc.), they do provide a set of function examples sufficient to demonstrate mappings to FACE 3.1 and that the breadth of the FACE 3.1 messaging paradigm features are needed to

successfully support the needs of the Radio Control program. These mappings are covered in the next section.

FACE Conceptual and Measurement Semantics

The following diagrams show the FACE Conceptual and Measurement Semantics in support of the examples above.

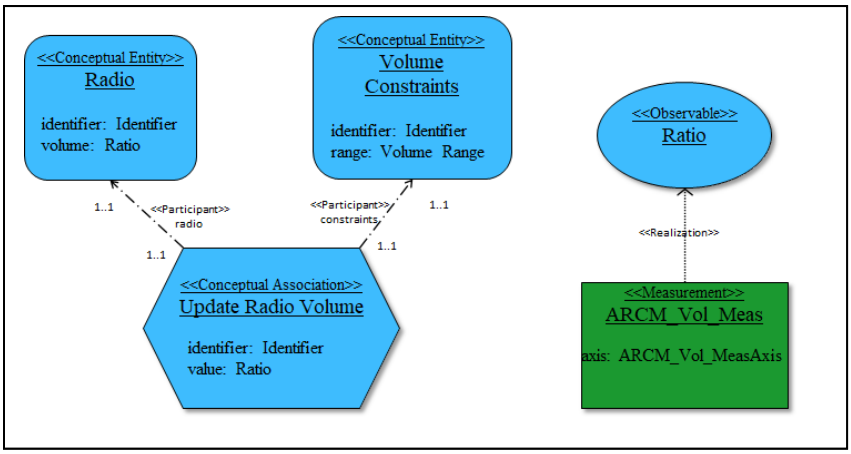

Figure 12: Conceptual and Measurement Semantics

FACE Connections and Message Types

Mapping Radio Control Capability Interface functions, as defined by the MMOSA process, to FACE Edition 3.1, takes into consideration the differing means by which the FACE 3.1 Data Architecture allows UoPs to communicate with other UoPs across the transport service segment (TSS)[3], *i.e.*, the different messaging paradigms.

MMOSA Capability Interface Function to FACE 3.1 Message Paradigm Mapping

The following table shows a partial mapping of MMOSA Capability Interface functions to FACE 3.1 messaging paradigm constructs.

Table 2: MMOSA Capability Interface Function to FACE Edition 3.1 Message Paradigm Mapping

MMOSA Capablity Interface Function Signature				FACE 3.1 Messaging Paradigm			
Queue Size	**Input Params**	**Output Params**	**Return Param**	**Paradigm**	**Connection**	**Direction**	**Message Type(s)**
0	Y	N	N	Subscribe	Single Instance	Inbound	messageType
0	N	Y	N	Publish	Single Instance	Outbound	messageType
>0	Y	N	N	Subscribe	Queuing	Inbound	messageType
>0	N	Y	N	Publish	Queuing	Outbound	messageType
N/A	Y	Y	N	Client	N/A	N/A	requestType (output), responseType(input)
				Server	N/A	N/A	requestType (input), responseType(output)
N/A	Y	Y	Y	Client	N/A	N/A	requestType (output), responseType(input,return)
				Server	N/A	N/A	requestType (input), responseType(output,return)

[3] Note that there are currently several PR/CR tickets documenting current deficiencies in the FACE Technical Standard. See https://ticketing.facesoftware.org tickets: 864, 914, and 944.

FACE Messaging Paradigms

The FACE Technical Standard Edition 3.1 specifies three general types of logical Connections (formerly called Message Ports in FACE Edition 2.1.x) to represent inter-UoP communication:

- Client/Server
- Publish/Subscribe
- Lifecycle Management

Now we will discuss the first two message paradigms: Client/Server and Publish/Subscribe, and how they relate to the MMOSA Capability Interface function types.

Client/Server Connection

FACE 3.1 Client/Server Connections support a Request/Response communication paradigm. The Connection specifies a role of either "Client" or "Server". Each Client/Server Connection references two Message Types, a "request" and "response" message type.

MMOSA Capability Interface functions that have both input and output parameters are represented as Client/Server Connections in generated FACE 3.1 models. For Radio Control, Client/Server Connections support the setting of radio data and radio commands as most of these Capability Interface functions take one or more input parameters and return a parameter indicating the success of the operation. Recall the `setVolume` Capability Interface function from the Radio Control use case above:

```
setVolume (input: volume, return: success)
```

For the FACE 3.1 data model representation, the Radio Control PSDS UoP provides the ability for a representative Application UoP to set the volume of the radio. The Radio Control PSDS UoP is in a server role whereas the representative Application PCS is in a client role. From the perspective of the Radio Control PSDS UoP, it is in the role of a "server". The "request" and "response" Message Types for the Server Connection of the Radio Control PSDS UoP look like:

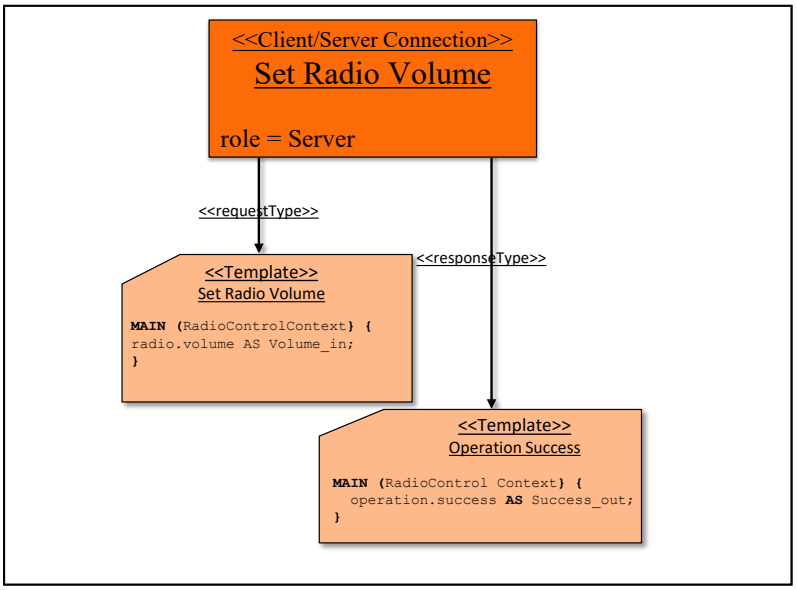

Figure 13: Radio Volume Connection (Server role)

In other words, the "server" Connection accepts (receives) a request type message to set radio volume, *e.g.*, "Set Radio Volume to 20". The "server" connection provides (sends) a response type message that specifies whether the request was successful or not, *e.g.*, "Operation Success TRUE/Operation Success FALSE".

Note that the naming convention for template members clearly conveys the direction the data is flowing across the Connection. When modeling Connection specifications (as with all specification models), it is a good idea to expose as much information as possible, even if this is perceived to be at the expense of reuse of model elements.

From the perspective of the Client UoP – the representative Application UoP in our example – the role is reversed. The "request" and "response" message types are essentially the same except for the directional parameter suffixes:

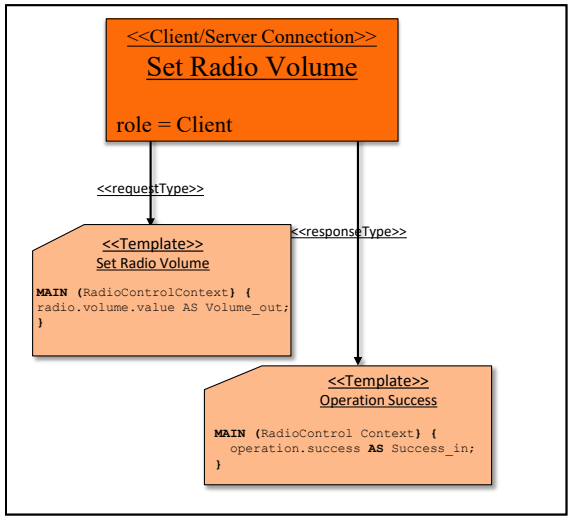

Figure 14: Radio Volume Connection (Client role)

The "client" Connection sends a request type message that specifies the desired radio volume, *e.g.,* "Set Radio Volume to 20". The "client" connection accepts (receives) a response type message that specifies that the request was successful*, e.g.*, "Operation Success TRUE" or not "Operation Success FALSE".

When modeling or reviewing Client/Server Connections, it helps to think about the request and response within the context of the role of the UoP Connection, either Client or Server, and the directionality of the data flowing across the Connection.

Publish/Subscribe

FACE 3.1 Publish/Subscribe Connections specify a single Message Type and are further characterized with a message exchange type (direction) of either Inbound Message or Outbound Message. Inbound and Outbound are from the perspective of the UoP and its Connection. A Publish/Subscribe Connection with an Inbound Message Exchange Type can be considered as subscribing to the Message Type whereas a Publish/Subscribe Connection with an Outbound Message Exchange Type can be considered as publishing the Message Type.

Publish/Subscribe Connections also specialize as Queueing Connections (those with a queue depth) and Single Instance Message Connections.

MMOSA Capability Interface functions that have *all* output, or *all* input parameters are represented as Publish or Subscribe Connections (respectively) in generated FACE 3.1 models. For Radio Control, Publish/Subscribe Connections support the getting of radio data as most of these Capability Interface functions have only output parameters. Recalling the `getVolume` Capability Interface function described above in the Radio Control use case:

```
getVolume (output: volume, output: source)
```

For example, for the FACE 3.1 data model representation, the Radio Control PSDS UoP publishes radio volume, which a representative Application UoP can subscribe to. The Radio Control PSDS UoP is the publisher of this data whereas the representative Application PCS is a subscriber to this data.

The radio volume being published by the Radio Control PSDS UoP looks like:

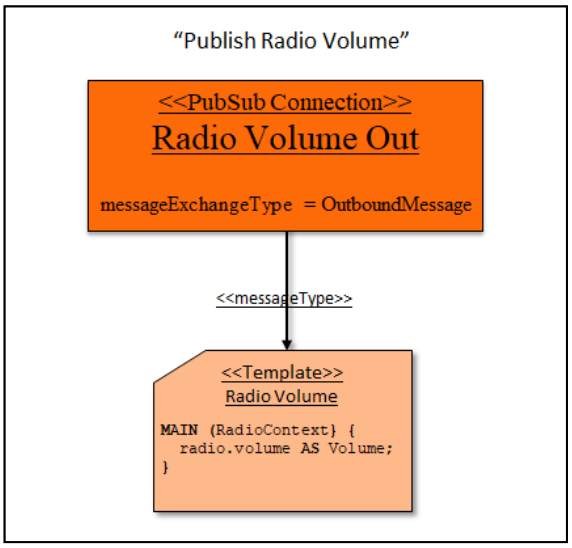

Figure 15: Publish Radio Volume

Whereas the radio volume being subscribed to (by the representative Application UoP) looks like:

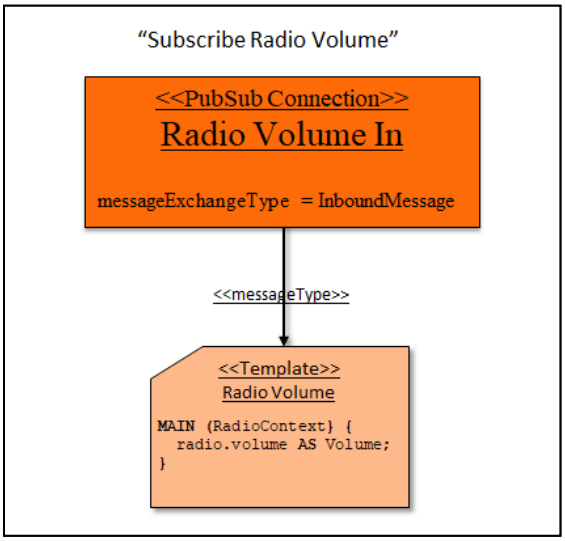

Figure 16: Subscribe Radio Volume

Conclusion

The FACE Reference Architecture (FACE RA) provides a rich set of functionality and interfaces enabling different teams to build PCS and PSS applications that can be integrated into one cohesive system. However, this comes at a cost of complexity of implementation. FACE components can be difficult to develop. This is mostly due to the required change in approach to FACE component development and the lack of ecosystem tools available to support FACE data architecture and interface development.

To manage and mitigate this complexity and change in approach, we believe one must utilize a formal process that leverages advanced tooling implementing Modular Open Systems Approach (MOSA), supports open technical standards, and a digital environment to support the complex application and data architecture development. It also requires embracing the full extent of the FACE Technical Standard.

This paper presented an ongoing real-world case study of the Model-based Modular Open Systems Approach (MMOSA) and AWESUM model-based tool suite to develop the U.S. Army's Aviation Radio Control Manager (ARCM) components to meet DO-178C DAL C and FACE Edition 3.1 conformance. ARCM implements over 700 messages controlling four devices.

Capability Model Elements	#
Capabilities	38
Action Messages	144
Publish Messages	577
Setter Messages	256
Total Messages	977
Fields	2,110
Types	1,971
Enumerations	476
Structs	3,879

Device Model Elements	#
Devices	4
Messages	320
Message Fields	4231

FACE Model Elements	#
Conceptual Entities	265
Conceptual Associations	715
Logical Entities	69
Logical Associations	660
Logical Measurements	784
Logical Measurements not in SDM	633
Logical Enumerations	221
Logical Measurement Systems	114
Platform Entities	69
Platform Associations	660
Platform Queries	707
Platform Types	505
Platform Enumerations	365
UoP Templates	1,274
UoPs	2

Figure 17 Communication Model Metrics

The overall result is the realization of the promise of the FACE Open Standard goal for enabling rapid reusable UoC development and simplified integration of UoCs for building composable systems.

TES-i, with the AWESUM tool suite implementing the MMOSA process, is realizing the promise of MOSA, AGILE, DevOps, and Digital Engineering and is showing how we are "*Going Faster with Open Standards*".

References

1. Mellor, Stephen; Balcer Marc: Executable UML, A Foundation for Model-Driven Architecture. Addison-Wesley.
2. FACE™ Technical Standard, Edition 3.1 (C207), published by The Open Group, July 2020; refer to: www.opengroup.org/library/c207
3. "FACE Share Data Model Governance Plan, Edition 3.0", X1817US. https://www.opengroup.org/face/
4. "*AV-2 for FACE 3.0*", Open Group FACE Plato site
5. "Air Force's Continuous Integration and Continuous Development with the FACE™ Technical Standard, FACE Army TIM 2018", Huntsville AL. https://tes-savi.com/publications
6. "*Modular Open Systems Approach (MOSA) Panel on Standards*", Zimmerman, Phil, Defense Standardization Program Workshop, July 12, 2018
7. "10 U.S. Code § 2446a - Requirement for modular open system approach in major defense acquisition programs; definitions", (4) The term "major system interface" part (b).
8. "Buyers, Suppliers, and Integrators, Oh My! FACE Army TIM 2020", Huntsville AL. https://tes-savi.com/publications
9. "Capability Driven Architecture an Approach to Rapid Platform Integration", Tucson AZ, Jan 2012. https://tes-savi.com/publications
10. "Composable Data Models (DSDMs)", Open Group FACE Plato site https://www.opengroup.us/face/diog/guidance/protected/documents.php?action=show&dcat=&gdid=22797
11. "*Impact in Action*", Open Group FACE Plato site

About the Author(s)

Sean P. Mulholland is a co-founder of Tucson Embedded Systems, Inc. Sean has a Bachelor of Science in Computer Science and Systems Design from the University of Texas at San Antonio. Sean currently serves as TES CEO and President. Sean has 32 years of experience in software intensive system development, design, integration and testing, especially as it relates to mission critical and safety critical systems. Sean has designed and built several product lines that produced significant advancements in the areas of Geographic Information Systems, Military Ground Systems, Unmanned Ground Vehicles, Unmanned Aerial Vehicles, and Manned aircraft systems. Sean is a contributing author to the FACE™ Technical Reference Architecture and has been active serving as a key resource in FACE Data Architecture development and serves as co-lead in the Domain Interoperability Working Group (DIOG) Guidance Group. Sean's current work is focusing on the development of a process and supporting tool suite for optimizing the system development of safe and secure systems for military and commercial markets.

William G. Tanner is a contributing author to the FACE™ technical reference architecture and has been active in the FACE Domain Interoperability Working Group (DIOG). Bill is currently the lead data modeler for several TES, TES-SAVi, and Army projects across varying subject matters. Bill has a Bachelor of Science in Computer Science Engineering from Northern Arizona University and over 30 years of embedded software, embedded software application development, and modeling experience, half of which are in software engineering project and product management, the other half directly related to systems and software engineering and modeling using UML, xtUML, and the FACE Data Architecture. Prior to joining TES in 2006, Bill worked for IBM as a software engineer in the disk storage system division, and Project Technology and Mentor Graphics as a data modeler, OOA/OOD instructor, and project manager for the division's UML modeling, verification, and code generation tools.

About The Open Group FACE™ Consortium

The Open Group Future Airborne Capability Environment™ (FACE) Consortium, was formed as a government and industry partnership to define an open avionics environment for all military airborne platform types. Today, it is an aviation-focused professional group made up of industry suppliers, customers, academia, and users. The FACE Consortium provides a vendor-neutral forum for industry and government to work together to develop and consolidate the open standards, best practices, guidance documents, and business strategy necessary for acquisition of affordable software systems that promote innovation and rapid integration of portable capabilities across global defense programs.

Further information on the FACE Consortium is available at www.opengroup.org/face.

About The Open Group SOSA™ Consortium

The Open Group SOSA™ Consortium enables government and industry to collaboratively develop open standards and best practices to enable, enhance, and accelerate the deployment of affordable, capable, interoperable sensor systems. The SOSA Consortium is creating open system reference architectures applicable to military and commercial sensor systems and a business model that balances stakeholder interests. The architectures employ modular design and use widely supported, consensus-based, nonproprietary standards for key interfaces.

Further information on the SOSA Consortium is available at www.opengroup.org/sosa.

About The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through technology standards. With more than 870 member organizations, we have a diverse membership that spans all sectors of the technology community – customers, systems and solutions suppliers, tool vendors, integrators and consultants, as well as academics and researchers.

The mission of The Open Group is to drive the creation of Boundaryless Information Flow™ achieved by:

- Working with customers to capture, understand, and address current and emerging requirements, establish policies, and share best practices
- Working with suppliers, consortia, and standards bodies to develop consensus and facilitate interoperability, to evolve and integrate specifications and open-source technologies
- Offering a comprehensive set of services to enhance the operational efficiency of consortia
- Developing and operating the industry's premier certification service and encouraging procurement of certified products

Further information on The Open Group is available at www.opengroup.org.