Lessons Learned for Developing and Integrating MOSA DO-178 FACE Control Components

The Open Group FACE™ and SOSA™ Consortia 2023 Army and Technical Interchange Meeting (TIM) and Expo, by:

Sean P. Mulholland, TES-i, TES-SAVi
William G. Tanner, TES-SAVi
Ken Erickson, TES-i
Todd Moore, TES-i
September 2023

Table of Contents

Executive Summary

TES-i and AWESUM are successfully "leading the charge toward Modular Open Systems Approach (MOSA)" by building Device Control utilizing a MOSA, DO-178C and the FACE™ approach. This paper presents our lessons learned over the past years applying the FACE Technical Standard, Edition 3.1 to build and integrate FACE components that are airworthy and developed using Model Based System Engineering (MBSE) models to auto generate documents, code, and tests.

It is important to note the AWESUM tool suite is built on the tenet that "data belongs to the user". All data formats leverage open standards and simple interfaces to implement complete transparency as per the Open Group vision of Boundaryless Information Flow.

Introduction

For over two decades, TES-i has supported our customers in developing reusable systems that adhere to the most stringent standards and qualifications for airworthiness, automotive and medical safety. Many lessons TES-i learned the "hard way" can provide guidance as the Department of Defense (DOD) moves towards reusability in an open and standard approach.

TES-i and its subsidiary, TES-SAVi, serve in a unique role in the open standards ecosystem. While we are a relatively small company (60+ employees), we have led in real-world development and integration through many varied roles:

- Reusable Software Component (RSC) developer
- Reusable Verification Component (RVC) developer
- Hardware developer
- Process developer
- Systems integrator
- FACE Conformance Verification Authority (VA)
- Open standards developer
- Data Model experts
- Subject Matter Experts (SMEs) in FACE™ Technical Standard
- SMEs and Trainers for FACE Data Architecture (DA)
- SMEs in HOST and CHIL

This paper presents our lessons learned over the past years in applying the FACE Technical Standard, Edition 3.1 to building and integrating FACE components and a case study of the Aviation Radio Control Manager (ARCM) component that will attain FACE Conformance, DO-178C DAL C airworthiness and has been developed utilizing very advanced Model Based System Engineering (MBSE) tools and processes.

Case Study

Device Control with MOSA

When people think of reusable software components, they seldom think that device control software which provides external device configuration, control and input/output would be a prime candidate for reuse. After all, the devices on our aircraft systems are often somewhat unique or at least customized per installation. However, the U.S. Army realized many years ago that they could gain large savings across the fleet of aircraft if they bulk purchased their mission equipment. Over the years, this purchasing has led to the reuse of mission equipment devices and created the huge potential for reuse of the software used to control these devices. The types of devices we have interfaced with include communications, aircraft survivability, navigation, turbine control and more.

Over the last 20 years we have developed a formalized approach to developing control software called the Capability Driven Architecture (CDA). CDA provides the ability to build device control interfaces that adhere to open system standards largely through automation. Originally, the automation was not built to be FACE Conformant as it preceded the FACE Technical Standard, but has since been adapted to produce fully FACE Conformant applications built to DO-178C guidance.

Below we will dive into an example of the ARCM set of FACE Components that provide a capability-based interface that is targeted to be FACE v3.1 conformant this year and meet DO-178C SL-C airworthy certification shortly after.

ARCM Background

ARCM is a set of reusable platform-portable software components that facilitate the integration of (legacy and next-generation) radio systems onto Army Aviation platforms. ARCM is being developed utilizing a Modular Open Systems Approach (MOSA) as well as DO-178C DAL/SL-C. ARCM is the U.S. Army's successful collaboration between Aerial Communications and Mission Command (ACMC), Georgia Tech Research Institute, TES-i, and is coordinated with the MOSA-TO which is now part of the APEO Engineering and Architecture.

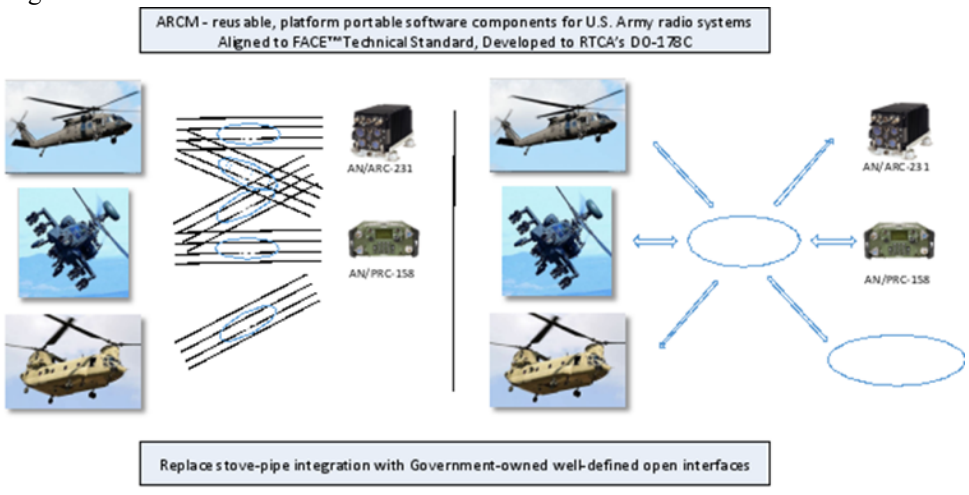


Figure 1: Solution Space for ARCM - Common Radio Control

ARCM Software Architecture

ARCM software is based on a layered software architecture aligned with the FACE Reference Architecture. ARCM Computer Software Configuration Items (CSCIs) reside in the FACE Portable Component Segment (PCS) and Platform-Specific Services Segment (PSSS). ARCM incorporates a Real Time Operating System (RTOS) Platform Abstraction Layer (PAL) Application Programming Interface (API) that provides a standard interface between the ARCM CSCIs and the FACE Operating System Segment (OSS), allowing for source code portability of ARCM across multiple environments with no impact to the CSCIs.

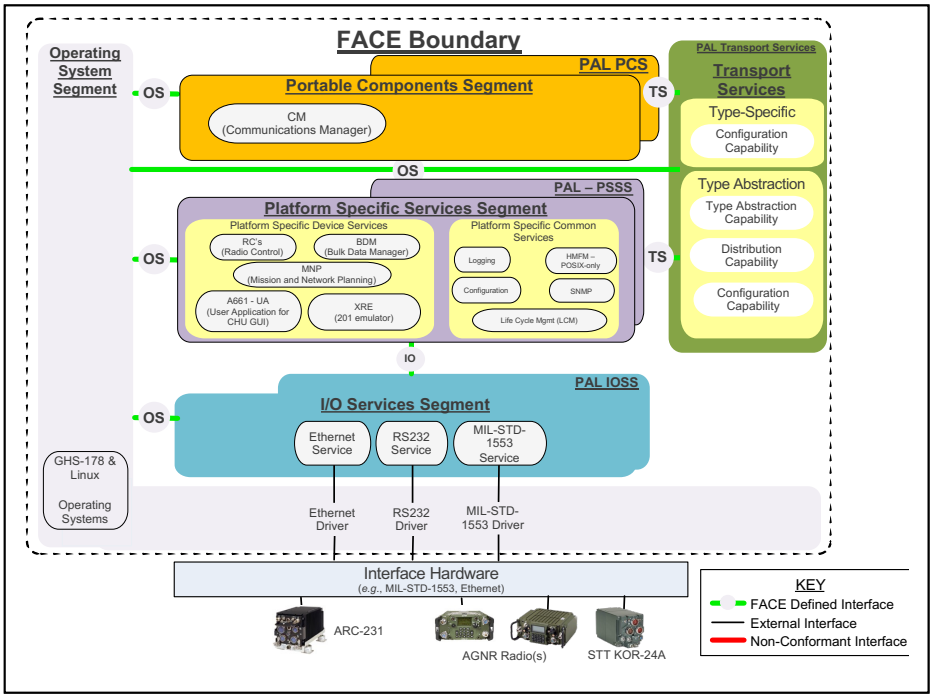


Figure 2: Radio Control FACE Diagram

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 separate requests 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 interface control

document (ICDs) must also be supported, unchanged, in their entirety[1]. In support of such a wide variety in radio messaging paradigms, the Radio Control Platform Specific Device Service (PSDS) makes use of a significant portion of the Model-based Modular Open Systems (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:

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

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

  Radio Publish Functions publish/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 control data. Most of the setter functions are inverses of publish 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])
  ```

  Radio Setter Functions take as a single input parameter the desired data item value. 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, and set data which sends data to the radio or synchronize which syncs with the radio
  - **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:

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

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

In ARCM, each of these abstracted capability functions were developed with device ICDs input and abstracted via the CDA process. Applying the CDA process results in non-proprietary open modular interface descriptions. In our use case, ARCM was developed with an abstracted interface of the Communications Domain. The ARCM abstract FACE interfaces support the federated radio devices that are interfaced through binary based ICDs and the new network radios that are interfaced through Simple Network Management Protocol (SNMP) Management Information Base (MIBs).

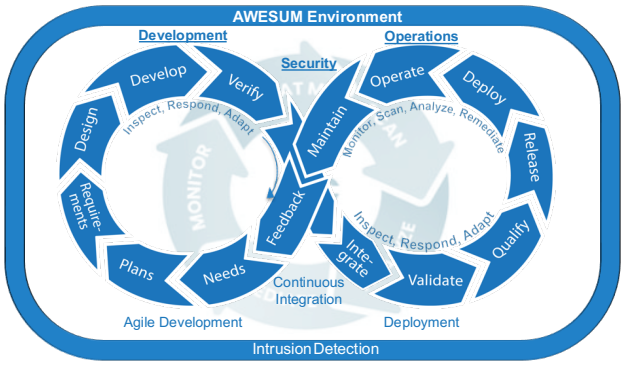
MMOSA Process

ARCM has been developed by following a digital engineering MOSA based process called MMOSA. MMOSA is a lifecycle process for cyber-physical systems development utilizing digital engineering concepts for implementing a MOSA with Agile and DevSecOps (*short for development, security and operations*) techniques in a manner such that the resulting system is qualifiable.

The MMOSA Lifecycle Process figure 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 and core models utilized in MMOSA:

Figure 3 MMOSA Lifecycle Process



MMOSA Process Key Tenets

- 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 RefArch, ObjArch, SysArch 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

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

Figure 4 MMOSA Process Tenets and Models

The following section focuses the MMOSA bottom-up approach identifying complete specificity for reusable capability interfaces utilizing the FACE Data Architecture to fully specify the semantics of the interfaces.

MMOSA Capability Interface Model

The MMOSA Capability Interface Model (CIM) implements the patented "Capability Driven Architecture (CDA)" [9] to provide 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.

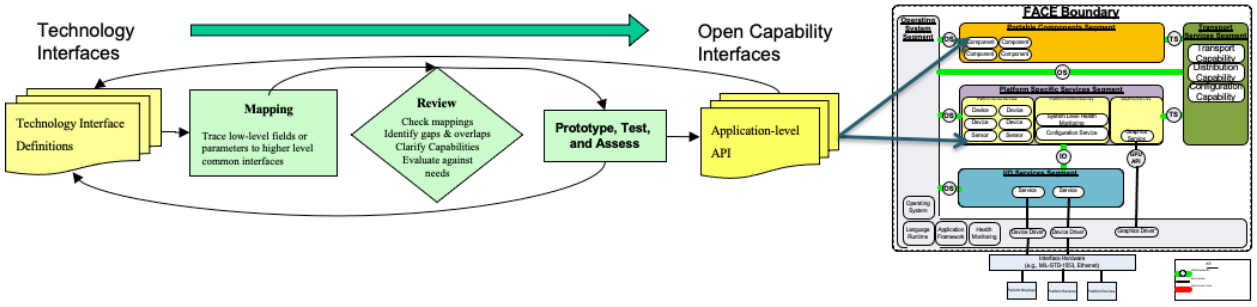


Figure 5 CDA Process for Defining FACE Conformant Capability Interfaces

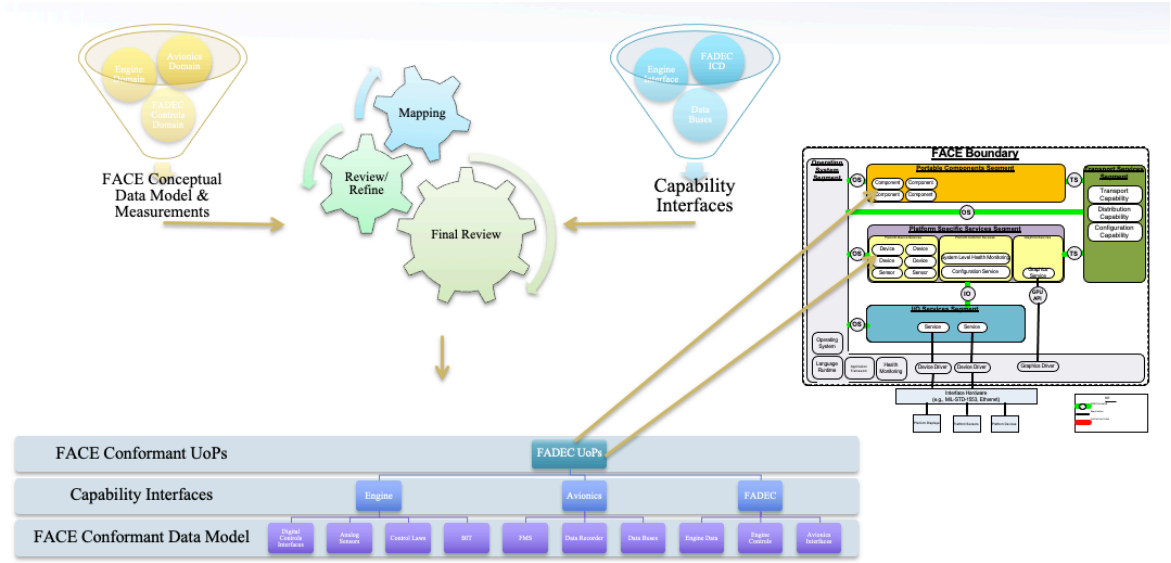
The CDA process, depicted above, is a combination of top-down and bottom-up approaches where the input to the process is the system requirements and the low-level interface documents. These low-level documents are imported into the AWESUM tool suite residing within the System Unified Model (SUM) database with each paragraph individually managed. 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 and 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 is defined while retaining provenance of the interface elements and traceability supporting DO-178C. 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 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 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 9: Capability Interface Development with Integrated FACE CDM and 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 SUM:

- Interface requirements
- Interface Design
- FACE Logical and Platform Model Entities, Associations, and Queries
- FACE Unit of Portability (UoP) Portable Components Segment (PCS) and Platform-Specific Services Segment (PSSS) Models and Templates
- FACE Transport Services Segment (TSS) code
- Capability Interface to/from External Interface code

- Test Cases, Test Procedures and Test Results from procedure execution
- Traceability Model
- Documentation Artifacts

FACE Data Model Generation

With such a concise specification of the CIM, which includes referencing the Conceptual and Measurement Semantics provided by the FACE UoP Supplied Model (USM) or Domain Specific Data Model (DSDM), the AWESUM Tool Suite generates the requisite FACE Conceptual, Logical, Platform, and UoP model elements[2] for a FACE Unit of Conformance (UoC).

The ARCM FACE Conceptual Data Model and Measurement Semantics are specified in such a way to minimize the duplication of modeling data. In fact, the entire FACE Logical Data Model (LDM) and Platform Data Model (PDM) entity models are auto generated by the AWESUM tool suite.

The UoP Models along with the FACE Templates (IDL) and Queries are auto generated from the Capability Model. The effect is that Message Models are developed for reusable capability-based models that have sufficient specificity to auto generate most of the FACE Data Architecture model artifacts thus greatly reducing the work effort by a factor of 10.

The chart below shows the number of elements that are generated for ARCM. Note that ARCM implements device control for 5 devices with 8595 ICD message fields providing 157 FACE Transport Service Messages across 2 primary UoPs.

Device Model Elements	#
Devices	5
Messages	355
Message Fields	8595

Capability Model Elements	#
Capabilities	71
Functions	157
Published Elements	976
Setters	430
Variables	4049
Total Functions	1563
Parameters	3079
Types	1971
Enumerations	476
Structs	3879

FACE Model Elements	#
Conceptual Entities	410
Conceptual Associations	392
Conceptual Observables	75
Logical Entities	43
Logical Associations	117
Logical Measurements	1,275
Logical Measurements not in SDM	1,124
Logical Enumerations	446
Logical Measurement Systems	114
Platform Entities	40
Platform Associations	117
Platform Views	113
Platform Types	390
Platform Enumerations	320
UoP Templates	12,743
UoPs	2

Figure 10 Communication Model Metrics

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

Lessons Learned

The following are lessons learned that our data architects, system engineers, trainers, testers and FACE subject matter experts have compiled over the years. The lessons are grouped by category to aid the reader, but they sometimes overlap. It is a good idea to read all of them as you never know which one(s) may resonate with your situation.

Data (Model) Architecture (DA)

1. The FACE DA is difficult for most system and software engineers to learn.
   The FACE DA was designed to provide the high level of specificity needed to describe FACE Transport Service messages. Because of this need, the FACE DA requires different skill sets than are traditional for both system and software engineers. For this reason, engineers need to develop new skills to be able to produce data models that adequately describe the semantics and the mathematical foundation of the message fields.

2. A small team of data modelers can often out-produce a larger team of modelers.
   The complexity of the FACE Data Architecture is well known as indicated by papers written on the subject. The FACE DA necessitates data modelers to understand the FACE metamodel, model rules of construction, semantic modeling, measurement modeling, UoP interface modeling, and code generation. Because of this complexity modelers need to be skilled at abstract thought and exhibit detail oriented attributes. We have found that very few people have these basic attributes to be top notch data modelers. Therefore, we have seen that the cultivation and training of modelers with a well defined process to be effective in producing highly productive data modelers. The key is to start with persons that are capable abstract thinkers, are highly detailed oriented, and desire to learn a new way of developing systems. By providing these persons with high quality training, process, and tools that directly support the FACE DA we have found a very small team of one to three persons can produce significantly higher quality data models in far less time than teams that are two to three times that size.

3. DSDMs are difficult to build and more difficult to use.
   The concept of domain models is a straightforward concept, but was not the original design of the FACE DA. The FACE DA was originally designed to produce one common set of Conceptual Entity-Association model. The idea was that through refinement and addition of model elements, the Conceptual data model for the Aviation community would emerge and the need for data model merging would not be required at the conceptual level. This approach was deemed to not be executable by the FACE Consortium and was abandoned. In its place, a concept for UoP Supplied Models (USMs) was championed. However, that approach did not have a method for data model sharing. Thus, the concept of Domain Specific Data Models (DSDMs) was added to the FACE Technical Standard.

   Now we are seeing engineers struggle to understand the DSDMs being provided to them for their usage in building USMs. The DSDMs are often incomplete in that they contain a small percentage of the data elements and concepts required to build real world systems. They can also be built against a different version of the FACE Technical Standard. In addition, often the DSDMs have had errors and do not pass conformance. Our recommendation to these engineers is that they should try to understand and use the DSDM(s), but where they are not complete or compatible, the USM developers should develop their model extensions in alignment with the DSDM and provide feedback to the DSDM

owner. In this way, the DSDMs can be enhanced to support real world systems development.

FACE Technical Standard

4. FACE Technical Standard edition changes (even minor version changes) cause incompatibilities that often turn out to be significant and incompatible.
   How? Changes in tool chains, requirements, generated source code, programming language version differences, interface changes, and conformance changes just to name a few of the issues.
   For instance, updating component development from FACE v2.1 to FACE v3.x was a drastic change that caught us by surprise. We expected differences, but the level of incompatibility in all facets of development not only surprised several of our customers but us as well. The biggest impact we had to overcome was the injectable interfaces and code size impact. (This is discussed in other lessons learned in more detail).

FACE Conformance

5. FACE Conformance takes planning and time.
   Start conformance efforts early in the development cycle. Don't wait until late in the development process only to find the need for major refactoring or development efforts to achieve FACE Alignment/Conformance.

6. Conformance verification define the finish line.
   The FACE Conformance process is significantly different than what most engineers have experienced. Many engineers assume that passing the Conformance Test Suite is all that is required. However, the process is more involved and involves demonstrating adherence to all the requirements for the FACE segment. We recommend that FACE developers focus on achieving FACE Conformance by following the FACE Approach even if the component need to be rearchitected. The reason is that the FACE Technical standard was developed to ensure developers don't "side-step" requirements. Rest assured that potential loopholes were identified early on and closed as part of the FACE design.
   Note, that sometimes engineers believe they must rearchitect software components that are not required to be changed. It is best to consult a FACE SME or FACE VA early in the effort to ensure conformance is needed for every component.

FACE Operating Environment

7. RTOS's are getting better at fully implementing the FACE Approach but there are still gaps. For example, complete FACE Technical Standard & airworthy support are not always implemented such as with TCP/IP stacks.

FACE Transport Service

8. Middleware in the form of FACE Transport Services is advancing their support for FACE TSS, but there are still issues.
   For example,

   a. TPMs are being redefined for version 3.2 to incorporate Data (De)Marshaling and

(De)Serialization

b. TSS optional requirements that UoCs rely on cause portability issues

9. Less frequently utilized interfaces still need improved definitions.
   For example Configuration Services are relatively poor in their specification.

10. Amount of code in callbacks can increase compile time by a factor of 10. In addition, many Integrated Development Environments (IDEs) don't support large and/or complex software dependencies.

11. Code size grows multiplicativity with the increase in the number of FACE TS messages.
    As an example, ARCM contains over 150 transport service messages. While there are techniques to reduce these, the fundamental approach of the FACE Technical standard requires significant SLOC for each message. Much of this code is only executed at instantiation time, but still factors in to the overall SLOC.

FACE Integration

12. Integration time is greatly reduced by following the FACE Technical Standard.
    We have found that software components developed to the FACE Technical Standard are much easier to integrate into other systems due to the well-defined interfaces and interface descriptions.

13. Integration Models can be useful but are still underspecified and must be tied to a specific TS implementation. However, most TS implementations do not implement complex data transformations and data filtering as specified in the Integration Model.

DO-178C and FACE

14. The FACE Injectable Interfaces increase the SLOC and should be planned for.
    We have found that automatic code generation and test generation can mitigate the increased efforts that are often seen for FACE large scale application development.

15. DO-178C requires tool qualification for analysis support tools.
    Due to the large code sizes for FACE Applications, it is imperative to plan on using tools to reduce overall engineer workload. For instance, formal code reviews are difficult when the size of code increases due to #11 above. Analysis support tools can greatly reduce workload so tool chains and tool qualifications need to be addressed in project planning.

Development Process

16. Implement the FACE Approach from the beginning.
    Don't try to circumvent the FACE Technical Standard – embrace the FACE Approach in building modular reusable components restricted to the FACE allowed interfaces. Remember, the standard was built to prohibit circumvention.

17. Implement the most restrictive profile your system requires first.
    We have seen several companies develop an implementation of their software component(s) to the General Purpose profile and plan to adjust to support airworthy implementations later. We have found this is difficult due to the significant constraints on the Safety and Security profiles and they are left

with a significant amount of rework.
Therefore, we recommend you build your components for the most restrictive profile you will need to support rather than planning a staged approach. The Safety Base profile is usually the best to target as the Security Profile is so restrictive that many applications cannot be built on that profile. The Safety Extended profile, like the General Purpose profile, allows too many interfaces that can make it difficult to move to Safety Base and the Security Profiles.

18. FACE edition 3.x is significantly different than FACE Edition 2.1.
    FACE 3.x is significantly different, requires a different tool chain, different libraries, different software implementation, possibly a different architecture, and worst case a different fundamental design. This is particularly true as the FACE 3.x added injectable interfaces which touch most every part of a FACE UoC implementation. Other major changes are in the Transport Services Segment and I/O Services Segment. We recommend you seek out training on the differences as well as Subject Matter Expert advice on your overall system and software component design.

19. The Problem Report/Change Request (PR/CR) process for tickets to get incorporated and published into CTS often takes 1-2 years.
    This time frame needs to be understood in that it does not align to most development efforts needs. We recommend working with your FACE trainers and FACE Verification Authority to reporting issues and find workarounds rather than relying on Approved Corrections to be implemented. It is not always the case that it takes that long, but often the issues are complex and there often are not quick nor easy solutions. This is primarily true because the FACE Technical Standard has been used for mostly small scale development efforts and issues for large software components are just now being identified. ARCM is one of the largest sets of UoC FACE development efforts and has been a leader in identifying problems with the standard.

20. Get training and listen to the trainers.
    The FACE Approach for design and development for ruse and portability is a significant challenge for most engineers and developers. We have found that without expert training it usually takes about 2 years for an engineer to gain a solid understanding and acceptance of the FACE Technical Standard. Training often helps reduce this time frame by speeding understanding of the approach, teaching the technical reasons behind the approach, clarifying technical issues and tooling and reducing the time it takes for engineers to accept the change in processes and tools the FACE Approach requires.

Tool Chains Matter

21. The FACE Approach, Technical Standard, and Data Architecture require a different set of tools and libraries.
    Trying to “adapt” existing tools and software libraries rarely produces an acceptable, much less optimal, outcome. The FACE Technical Standard is a complex standard that requires tools that “understand” the standard and help the engineers produce conformant software.
    Often tools produce invalid data models and/or nonconformant or invalid code. This is exacerbated by the different Technical Standard editions.

22. Few FACE tools can support the different technical standard editions.
    In our experience, complex systems often require use of software components built on to different standards, versions, and software libraries. It is important that multiple editions of the technical

standard are considered when choosing a tool chain.

23. Because the FACE Technical Standard is relatively new many of the tools produce incorrect models, tests, and code.
    Choose your tool chain carefully and exercise the development process fully.

24. Tools need to support the CI/CD process to ensure development is proceeding with FACE Ensuring Verifying FACE conformance throughout the development process is important in reducing the risks involved in FACE component development.

25. Development time is greatly affected by "slow tools".
    In working with customers we have seen some tools take a day (or even a week) to perform such simple functions as data model export, data model verification, code generation, and code compilation. These long turn-around times greatly affect development and put projects at risk. Often the first thing we help customers with is to establish tool chains that can perform these processes in minutes versus hours or days.

FACE Outcome is a Success:

26. We have proven that we can auto generate ~2M source lines of code (SLOC) that is highly modular, uniform and easy to test and review.

    a. With this capability to auto generate SLOC, we need to stop measuring systems in terms of SLOC. SLOC it isn't very meaningful anymore as much of the FACE code, while verbose, has little significance to code execution time and memory utilization.

    b. Small teams can develop complex components that can be readily integrated on disparate hardware and operating systems and integrated with many applications with minimal effort and schedule.

Conclusion

The FACE Technical Standard provides a rich set of functional interface definitions enabling different teams to build PCS and PSS applications that can easily be ported to new hardware and operating systems as well as simplify integration to create a cohesive system. However, this comes at a cost of complexity of implementation of the applications, the Transport Service middleware, IO services, and support libraries. These complexities cause FACE components to be difficult to develop due to the required change in approach to FACE component development and the lack of availability of mature ecosystem tools and training to support the FACE Data Architecture and interface development.

To manage and mitigate the complexity of FACE component development and integration, we believe one must utilize a formal process that leverages advanced tooling that implements MOSA, supports open technical standards, and a digital environment to support the complex application and data architecture development. We also strongly believe one must embrace the full extent of the FACE Technical Standard rather than "bolting on" FACE support into existing development efforts.

While numerous FACE improvements can be addressed through FACE tooling and processes, some development challenges stem directly from the standard and cannot be "fixed" without modification to the FACE Technical Standard and FACE infrastructure. None of these changes or additions are "earth-shattering"; however, we must persist in enhancing and supporting the growing ecosystem of developers, integrators, and tool vendors, making the FACE Approach a reality. The most striking revelation might be that besides these evolutionary modifications, there are no major additions needed for the FACE Technical Standard to support real deployments of FACE components.

In summary, this paper presented the lessons learned from supporting many customers and direct activities with 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. The overall result is the realization of the promise of the open FACE Technical Standard goal for enabling rapid reusable UoC development and simplified integration of UoCs for building composable systems.

References

1. Mellor, Stephen; Balcer Marc: Executable UML, A Foundation for Model-Driven Architecture. Addison-Wesley.
2. C207 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 TES-SAVi and Tucson Embedded Systems, Inc. dba. TES-i-i. 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-SAVi President. Sean has 34 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 Standard and Data Architecture Guidance Reference Implementation Guide Volume 3, as well as being active in the FACE Domain Interoperability Working Group (DIOG). Bill is currently the lead data modeler for several TES-i, 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.

Ken Erickson is a Software Engineer 5 and FACE Subject Matter Expert within Tucson Embedded Systems (TES-i) for over 25 years. Ken has a B.S. in Computer Science and Bachelor of Computer Engineering from the University of Minnesota, Duluth. Ken has 30 years of experience in real time and embedded software and systems requirements, design, development, integration and test, including both mission and safety critical systems. He is an active participant in the FACE TWG Transport Services Subcommittee, FACE TWG Security Subcommittee, FACE Integration Workshop Standing Committee, FACE TWG Ada Conformance Tiger Team, and Integration Workshop – Getting Started Guide Subcommittee as well as a member of the TES-SAVi FACE Verification Authority team. Verification work includes conformance testing of various vendor PSS/PCS/OSS' and TES created TSS, PCS', PSS', and Data Models for FACE Technical Standard Editions 2.0, 2.1, 3.0, and 3.1. Ken is also an active participant in the SOSA Test Tool Technical Working Group and was Principal Investigator on Phases 1 and 2 HOST SBIR which advanced to follow-on Phase 3 work.

Todd Moore is a Software Project Manager, DO-178C and FACE Subject Matter Expert within Tucson Embedded Systems (TES-i) for over 20 years. Todd has a B.S. in Computer Science from the University of New Mexico State University. Todd has 30 years of experience in real time and embedded software and systems development, integration and test, focused on safety critical systems for both commercial and military aircraft.

About The Open Group FACE™ Consortium

The Open Group Future Airborne Capability Environment™ Consortium (the 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. Our diverse membership of more than 900 organizations includes customers, systems and solutions suppliers, tool vendors, integrators, academics, and consultants across multiple industries.

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.