2023 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
MODULAR OPEN SYSTEMS ARCHITECTURE TECHNICAL SESSION
AUGUST 15-17, 2023 - NOVI, MICHIGAN

Using FACE™ Technical Standard Features to Address Interoperability Between Ground Vehicle Domain Open Standards

Mark Snyder[1]
Chris Allport[2]

[1]L3Harris, Palm Bay, FL
[2]Skayl, Westminster, MD

ABSTRACT

This paper offers a technical strategy to use Future Airborne Capability Environment™ (FACE Data Modeling and Transport Services Segment (TSS) mechanisms to address interoperability concerns between multiple open standards. It discusses features of the FACE Technical Standard that facilitate interoperability including data modeling constructs to address various common digital schema technologies, TSS capability approaches to allow flexible interoperability, and open standards that can be addressed with the approach.

Citation: M. Snyder, C. Allport "Using FACE™ Technical Standard Features to Address Interoperability Between Ground Vehicle Domain Open Standards," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 16-18, 2023.

1. INTRODUCTION

Multiple Open Standards have been developed over many years to help the ground vehicle domain realize the benefits of a Modular Open Systems Architecture (MOSA). Some relevant standards include The Future Airborne Capability Environment™ (FACE) [1] and Vehicular Integration for C4ISR/EW Interoperability (VICTORY), Robot Operating System (ROS), Sensor Open Systems Architecture™ (SOSA), and NATO Generic Vehicle Architecture (NGVA). Technical and business strategies must be adopted in order to make Open Standards work together, as discussed in the prior paper on FACE/VICTORY interoperability by Elliot, et al [2].

This white paper discusses specific technical approaches which can yield an effective strategy for interoperability between multiple Open standards, focusing on features built into the FACE technical standards designed to enable such interoperability. In addition to addressing standards such as VICTORY, we discuss how to address a much broader set of standards that include those based on other datatype-driven interface definition methodologies.

2. Open Systems Interconnect Model

The Open Systems Interconnect (OSI) Model, shown in figure 1 below, is a logical framework for understanding components typically used in distributed systems. While a full understanding of the OSI model is beyond the scope of this paper, it is beneficial to understand how each standard typically relates to the model. This provides a better understanding of the technologies necessary to achieve effective interoperability.

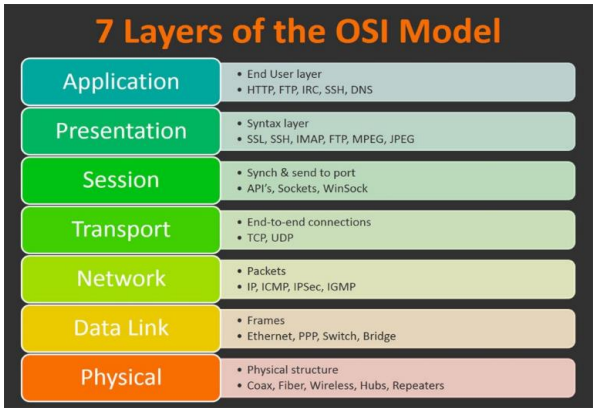

Figure 1: The OSI Model

The FACE Reference Architecture offers flexibility in assembling technologies at the Session, Transport, and Network layers to achieve performant integrations. VICTORY offers more flexibility at the application and presentation layers but is more prescriptive at the middle layers. Standards implemented primarily via Data Distribution System (DDS) and IDL, such as the NGVA standard, are prescriptive at the middle layers, and tend to rely on code generation for the presentation and application layer interfaces.

3. Data Modeling and Architecture

Data modeling for MOSA interoperability is primarily concerned with defining the system components' inputs and outputs. In general, the goals of data modeling include:

- Bridging the gaps between system architecture definitions and software implementation

- Providing an unambiguous definition of data structure that can be used for various aspects of the OSI model

- Providing an unambiguous definition of data semantics, or meaning, including the characteristics of how data is measured and reported

Each standard has differing methods of defining structured data types that may be considered a 'data model'. VICTORY and other XML-driven standards, for example, are defined using XML schema definitions (XSDs) for data structures that define messages. XSDs are a widely used method of defining structured data that is intended to be represented using XML. In VICTORY, the service definitions are described in the standard and codified by the web-service description language (WSDL) descriptions as components, which are roughly analogous to entities. NGVA, ROS, and others are built on Data Distribution System (DDS) and use Object Management Group (OMG) Interface Definition Language (IDL). IDL also includes the concept of modules, and service interfaces that are typically defined using an API that includes inputs and outputs that define services.

The FACE Consortium and the Open Group have defined the Open Universal Domain Description Language (UDDL) Standard [3] and UDDL modeling language and FACE UoP languages that can generate structured types in various forms. In developing the UDDL and data modeling approaches, the objective was to better enable the goals of data modeling through a more robust machine interpretable entity model. One approach to describe the maturity of a data

model is the Interface Documentation Maturity Levels (IDML), originally proposed in 2019 by Hand, et al., [4] is shown in Figure 2. The diagram provides a clear summary of the levels of interface documentation maturity. However, it is worth emphasizing the three distinct levels of maturity. Early maturity represents encompasses traditional paper-based documentation (or possibly no documentation at all). Advanced maturity captures all the current capabilities and recommended best practices of data modeling. This leaves the "mid" level of maturity with the most variance. On the early side (IDML 3), there exists schemas. Schemas are syntax-rich but devoid of semantics. Those that do capture the meaning of data do so in text that still requires a human to interpret. IDML 4 is a significant improvement over the former since it adds a tremendous amount of information about the data. Although this form lacks the full expression of what the data represents, it carefully captures the mathematical basis of the data (including, but not limited to, the units of measure and frame of reference) and the abstract concepts (called observable by UDDL) of each field.

Novice data modelers will often complain about "duplicating work" when building a FACE data model. This is typically a side-effect of building an IDML 4 "message model." In this style of data model, the data model elements mirror the applications' interfaces. This approach duplicates information in the model and does not require the powerful decoupling Query/Template mechanism explained below.

When modelers finally achieve IDML 5, they have a strong basis for a reusable entity model. This level exhibits decoupling between interfaces (messages) and the data model itself. This is a critical level of data modeling and the first level of maturity that really starts to realize the benefits of data modeling.

Simply put, if messaging standards were sufficient, all work would have been complete as soon as NATO Standardisation [sic] Agreement 4586 was published. How many standards have emerged since then? Regardless, countless engineering hours are directed at integrating these distinct standards. Therein lies an interesting observation – they *can* be integrated. Why?

Although the protocol may differ, although the units may differ, and although the messages may differ, all these message sets talk about the same things. Rather, they all talk about the same *domain*, they just do it in a slightly different way.

This ties back to the discussion about the OSI Model. Since the data models capture most (if not all) information about the domains, it is possible to calculate the integration between disparate message sets and delegate messaging, protocol, etc., to the appropriate level of architecture.

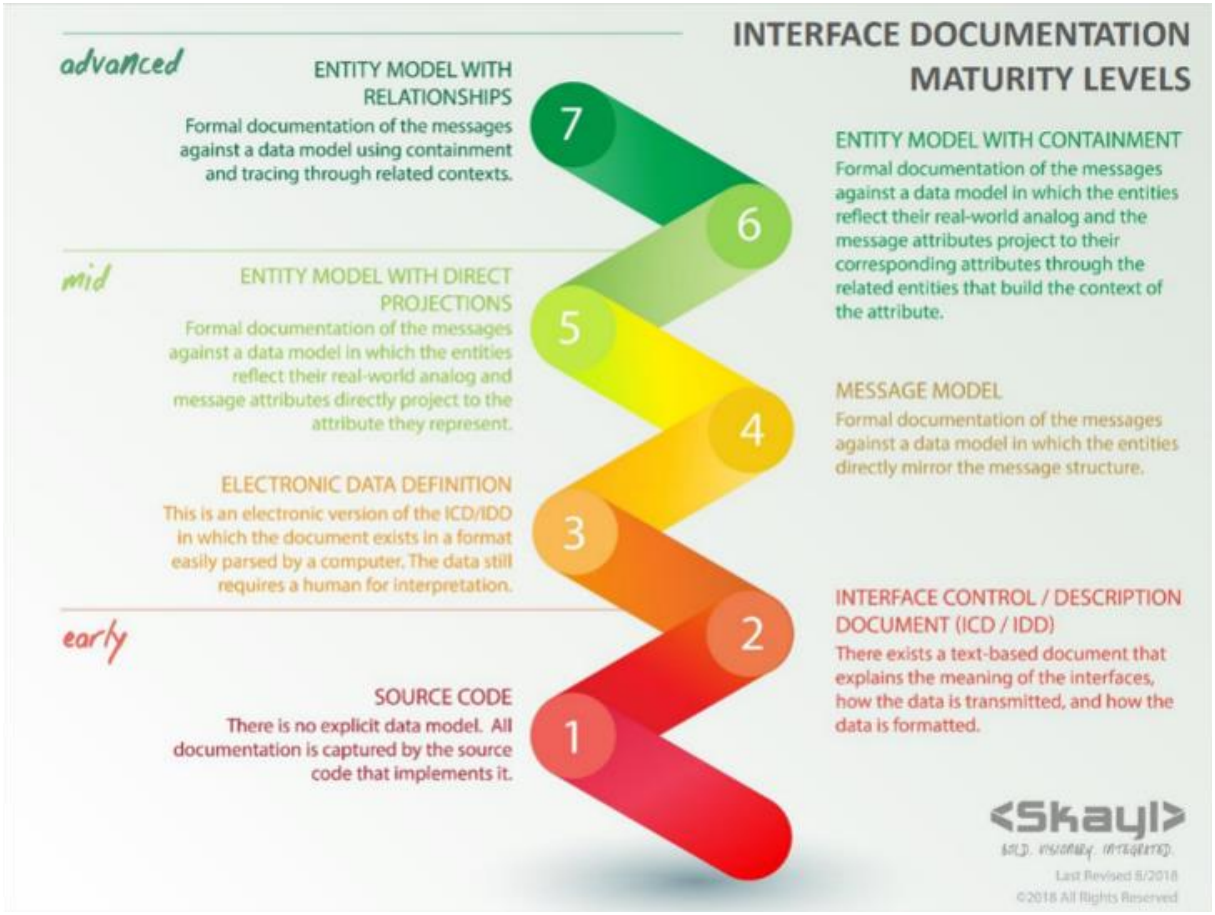

Figure 2: Interface Documentation Maturity Levels.

4. Data Modeling Relationship to Software

The relationship of data modeling to software is the subject of many challenges, as described by Snyder in a 2021 FACE TIM paper [5]. In general, a primary challenge is to maintain the relationship of data architecture modeling to software implementations that use it. For example, FACE portable applications at both the Portable Component Segment (PCS) and Platform Specific Services Segment (PSSS) can be written to standardized Type Specific (TS) interfaces that derive directly from the UDDL data model.

Since the UDDL language can be used to describe any interface in a flexible manner, a FACE UDDL model becomes the primary means to describe interfaces throughout a system. To align with the idea that data models capture "domains of data," the FACE Consortium created a concept of a Domain Specific Data Model (DSDM) to describe a more standardized set of interfaces used throughout a domain of interest (e.g., Ground Combat Vehicles). Many DSDM definition efforts are designed to be leveraged by multiple programs, offering a significant opportunity for reuse and commonality [6].

A secondary challenge of a data architecture (DA) is to describe the entity model in a way that maintains its relationship to data views, or messages, without its structure and

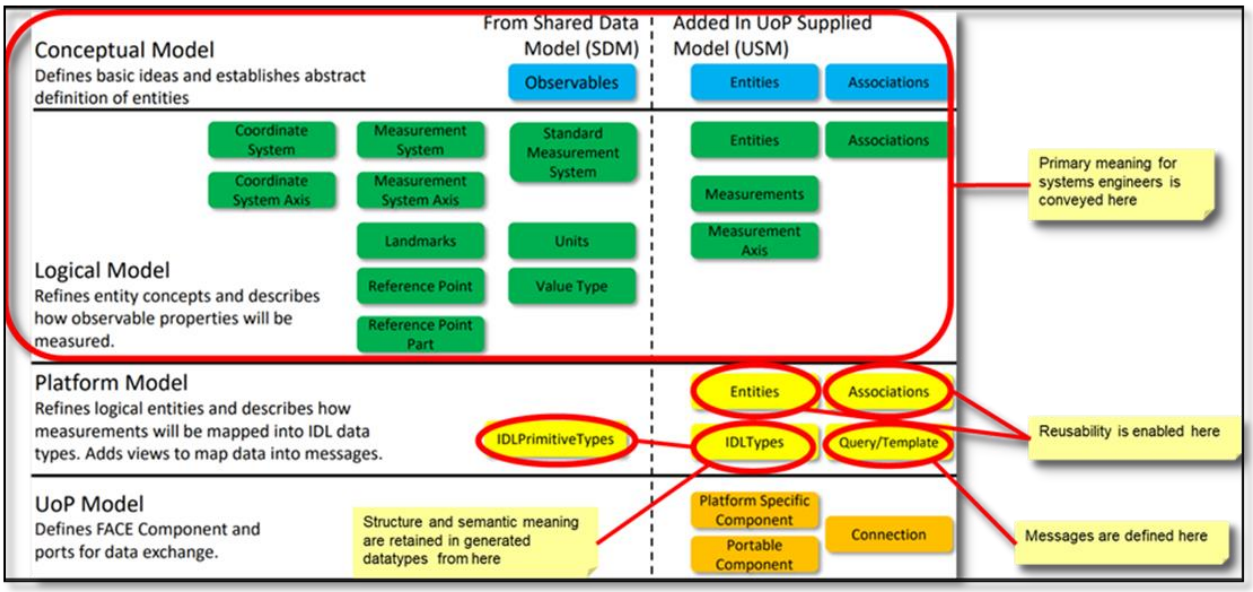

Figure 3: UDDL Data Architecture Traceability to Entity Model.

modularity being driven by messages (IDML 5 and above). In this way, the data model can capture both semantic meaning and necessary constructs to enable data transformation and interoperability. Figure 3 shows how the UDDL maintains traceability between the entity and platform views of the data architecture.

4.1. FACE Features Enable MOSA

The UDDL data architecture includes meta model elements that are intended to enable IDML 7 (Entity Model with Relationships) and decouple the entity/association model from interface/message representations. The primary mechanisms that support this are Queries and Templates. Queries are a mechanism that links a set of real-world elements (or entities) and their relationships (or associations) to describe the context for data exchanged in a system. It is this construct that allows a UDDL model to contain attributes of a described system to modular entities that are logically separate, and to express the interfaces as arbitrary groups of information. For example, the query below in Figure 4 shows how the separate attributes of a sensor and a gimbal are brought together in a single query. The documentation of the semantic (i.e., the meaning of the data), if expressed in the query itself. While not immediately obvious, it is possible to construct an "English sentence" from this query structure. In this case, the query is selecting information from an EOIR Camera and the gimbal on which it is installed. The template, on the other hand, is used to adapt the structure of the query data to what is needed by the application.

Tool vendors and other engineering organizations have developed tools to make the construction of FACE queries and templates fast, allowing the data model developer to consider the system context and not the semantics of the query or template languages.

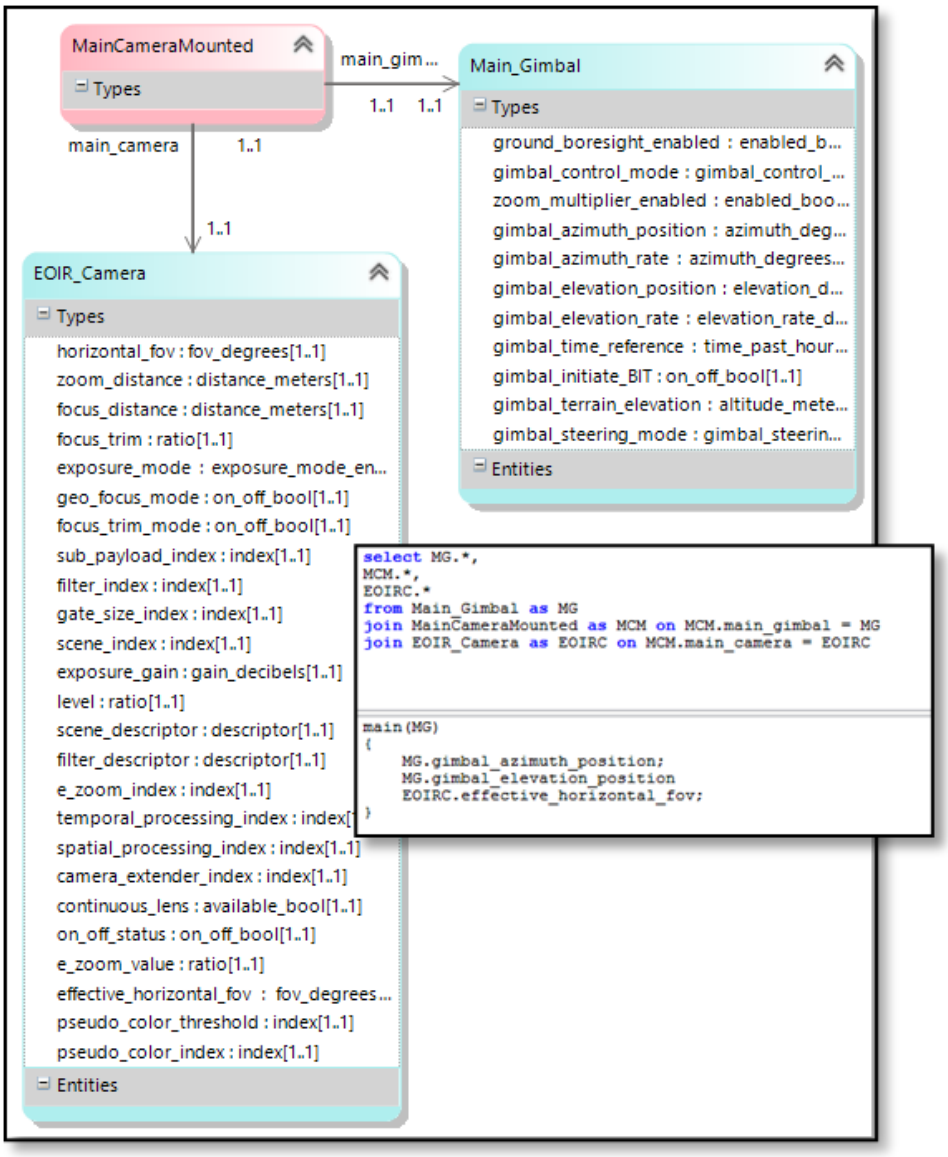

Figure 4: Example Query/Template.

The template language is a key part of the of the FACE Data Architecture to enable linking entity models to a variety of data layouts and applying them to multiple open standards. FACE templates include features such as sequences (lists of data), unions (choices of data), optional fields, renaming fields, and nested structures. These features allow templates to be defined that allow data definitions to closely match the features found in other standards (i.e., XSD or IDL). In essence, the template can match an intended format that can be related to another standard while the query links the template to the underlying entity definition. This is a key ability that allows non-FACE Data Architecture definitions to be described using FACE means, potentially via a reverse engineering process. Because the UDDL standard includes entities/associations and

underlying measurement systems, it may then be used to build adapters to other standards. These adapters can bridge the standards gap at both the upper and lower levels of the OSI model. In addition, entities and associations can be refactored to promote modularity without changing the templates.

For example, the model in Figure 5 shows an NGVA data definition (defined as an IDL) that is imported through a guided reverse engineering process to add semantic meaning, such as FACE logical

match the reverse-engineered schema, while the queries tie the entities to an underlying entity model. A similar process is possible to tie FACE models to imported VICTORY or other XSD schemas.

5. Relating the FACE Data Model to Lower Levels of the OSI Model

Once the FACE data model is defined with templates that closely relate to the needed data formats, automated processes become possible to build model-based data adapters

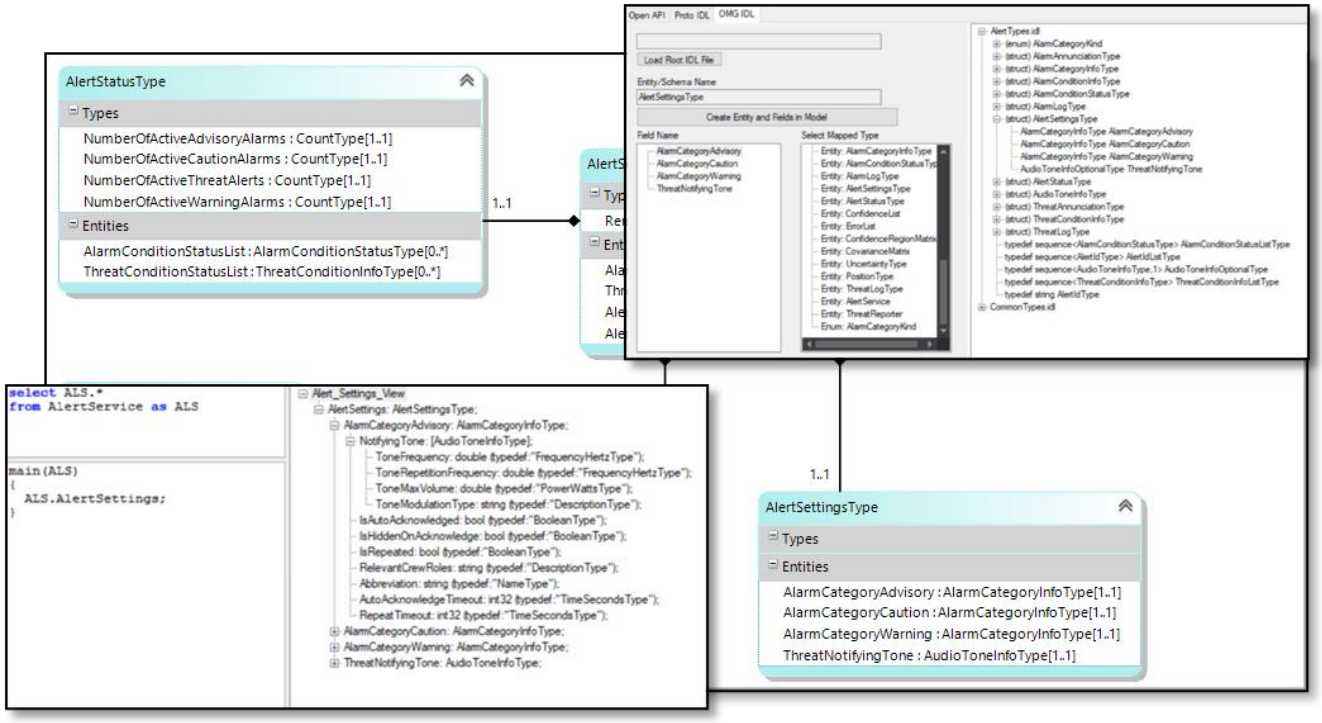

Figure 5: Example IDL from a FACE Data Model.

measurements and assignment of measured attributes to entities or associations in the model. Below is an OMG IDL definition of the data types, as specified in a standard like NGVA. In the center and right are FACE model entities and query/template that generate a FACE template version of the IDL that matches the data types and structures from the IDL. The generated template view matches the IDL in structure and the names

to allow other standards to communicate via the FACE TSS. For instance, to do this with VICTORY, one can first note that the VICTORY specification does not levy an API standard at the OSI application layer. An application can be VICTORY compliant as long as it is shown to implement VICTORY service patterns and communicate via the VICTORY data bus. Both conditions can be satisfied by building a FACE TSS adapter

that communicates via VICTORY at the back end and a conformant FACE TS API on the front end. The FACE TS API is actually very simple in concept. It presents 4 primary interfaces to software that is communicating via the FACE TSS:

- *Send_Message* to send a type-specific message (for instance the Alerts Settings message described above)

- *Receive_Message* to receive a type-specific message, usually through some form of a polling process under caller control

- *Register_Callback* to register a function that is called whenever a message arrives

- *TS_Extended* extends the callback mechanism to allow a type-specific sender and receiver to be paired. This is used to support RPC service patterns where a service sends a typed response

In a FACE environment, the system integrator chooses technologies and software modules to implement a given set of TSS objectives. To enable a FACE UoP to 'be' a VICTORY UoP would involve:

- Developing a TSS component that uses the VICTORY data bus message format on the wire and creates FACE TS APIs at the Presentation/Application layer. This could be done through a combination of auto generation or generic model driven software, depending on the capabilities of the chosen TSS.

- Prescribing a UoP pattern that implemented appropriate VICTORY service patterns. This might, for instance, 'wrap' existing VICTORY implementation logic into the FACE integrator code, so that UoPs would send or receive VICTORY Data Bus messages prepared using FACE TS data types that were seamlessly transformed to the appropriate wire format. VICTORY service discovery and negotiation would reside in the integrator code, transparent to the UoPs

This strategy would have multiple benefits. Once different endpoints were written to FACE TS API standards, the VICTORY Data Bus implementation usage could be reduced and eventually eliminated, allowing system integrators to use the more powerful and abstract FACE TSS to integrate their system. This would allow these new VICTORY compliant applications to interoperate with MOSA interfaces from other standards, while supporting existing VICTORY applications for as long as needed.

6. Example VICTORY Adapter

To test this concept, we implemented a VICTORY adapter to translate VICTORY Data Bus Position and Orientation messages from standard VICTORY XML-on-the-wire to a FACE Transport Services (TSS) implementation (Figure 6). The basic steps for this effort were:

- We constructed a simple FACE UDDL model for a ground vehicle with a position sensor. This model was not specific to the VICTORY design, but was generic in nature.

- We used the FACE templating mechanisms to construct a FACE template that matched the elements of the VICTORY Data Bus messages. The fields and structure of this template were designed to

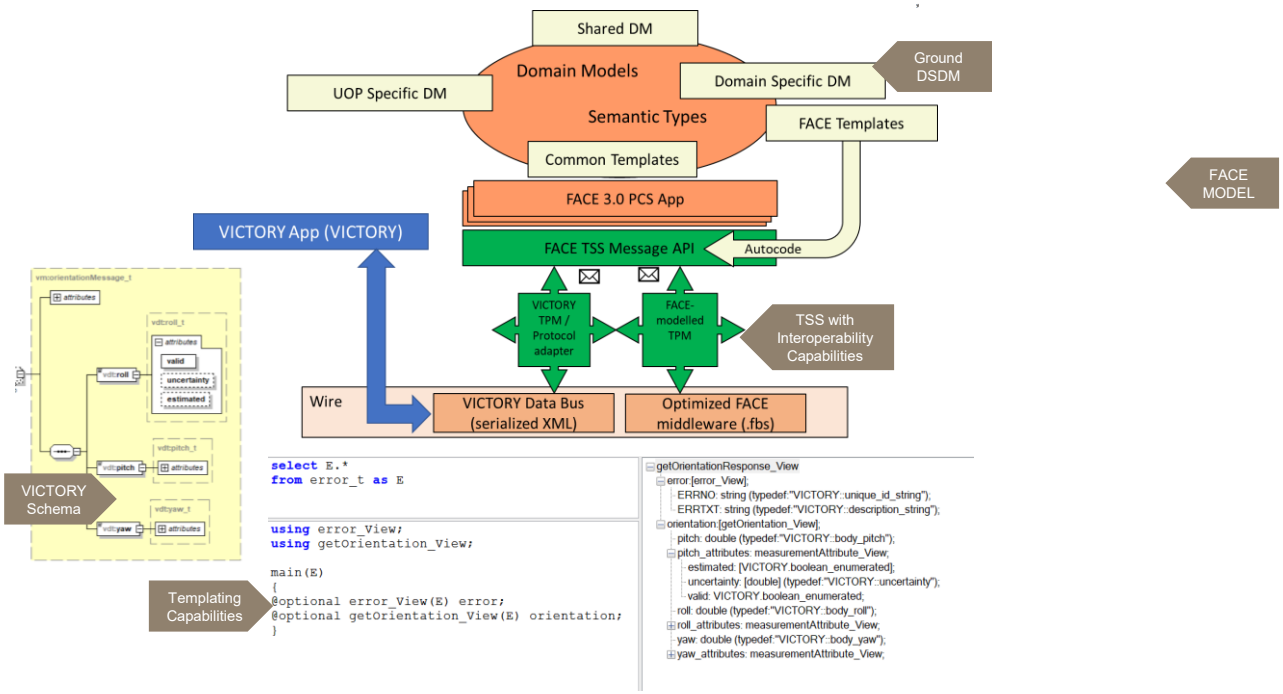

Figure 6: FACE-VICTORY Adapter Demonstration Architecture

match the XML message structure, to facilitate the planned adapter strategy.

• We employed a FACE TSS implementation that included *Type Reflective* capabilities, that supported the use of digital schemas at runtime to allow message structure contents to be understood by an adapter component employed in the FACE TSS.

• We built the adapter component with the ability to parse the XML off the wire, and to use the digital schema to create message buffers using in the TSS format and send them across the TSS middleware.

To test the approach, a virtual environment was used that simulated a vehicle sending VICTORY position messaged using XML over UDP sockets – a simplified VICTORY Data Bus implementation (Figure 7). The adapter was listened to these messages and bridge them onto the FACE TSS. A FACE UoP applications using Type Specific API calls was written to receive and display the position on an operator HMI. The demonstration operated as expected and showed the usefulness of the FACE model driven approach and flexibilty of the FACE and UDDL standards to adapt to the interoperability needs of this system.

While this adapter was simple because the message layouts were selected to be easy to automate, this is not always the case. Fortunately, there is an active community of developers providing tools and solutions that make this adaptation tractable.

The challenge of interoperability can be met, for instance, by employing model-driven Domain Specific Languages (DSLs) that allow the digital model of the data and relationships between system elements that

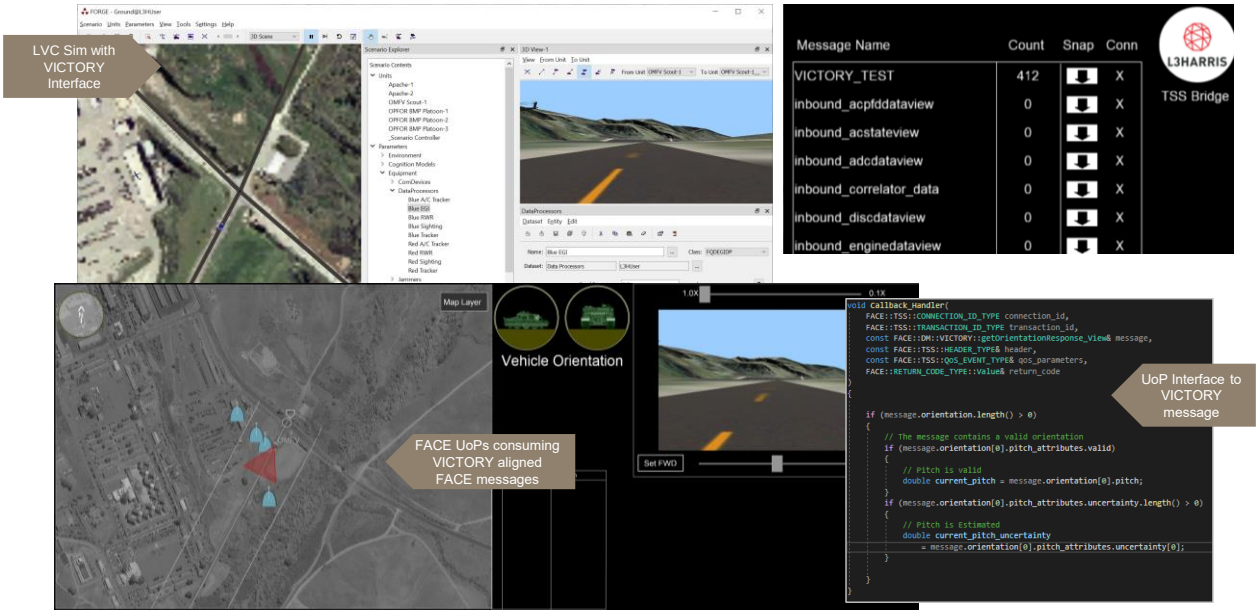

Figure 7: FACE-VICTORY Adapter providing Live Position on a FACE Digital Map

require or provide the data to be modelled. These models can then be used to facilitate code generation and other smart adapter strategies. Several commercial tools, such as Skayl's Phenom, exist to meet these challenges, and are widely used to solve domain interoperability problems in many MOSA contexts across DoD.

7. Conclusion and Recommended Approach

To effectively implement a strategy to make FACE, VICTORY, NGVA, and other standards interoperate, the following steps can be followed:

• Employ FACE UDDL and targeted UDDL construction tools to reverse engineer existing data specifications (XSL, IDL) into FACE models that match their structure.

• Add any meta model constructs that work with the FACE UDDL to enable automated and generic adapters to be constructed. These constructs allow the interoperability to be model driven, and these can be in a SysML tool or using dedicated DSLs, preferably commercially supported ones.

• Once suitable templates are constructed to adapt to standards, refactor the entity models to a common entity model structure that reflects the domain.

• Build or adopt a TSS infrastructure that is configurable and flexible using type-aware technologies. Make smart generic data adapters that handle interoperability with key standards, and employ tools designed for these purposes as necessary.

8. REFERENCES

[1] FACE™ Technical Standard, Edition 3.1 (C207), published by The Open Group, July 2020; refer to: www.opengroup.org/library/c207

[2] Elliott, L., Jenkins, S., Moore, M., and Yee, Howell, "Potential for VICTORY and FACE Alignment – Initial Exploration of Data Interoperability and Standards Compliance", proceedings of the Vehicle Electronics and Architecture (VEA) and Ground Systems Cyber Engineering (GSCE) Technical Session, 2019 NDIA Ground Vehicle Systems Engineering and Technology Symposium, August 2019

[3] Open Universal Domain Description Language (C198), published by The Open Group, July 2019; refer to: www.opengroup.org/library/c198

[4] Hand, S., et.al., "Interface Documentation Maturity Levels: An Introduction", proceedings of the Army FACE TIM, October 2018

[5] Snyder, M., "FACE Data Modeling for Software Developers", proceedings of the Army FACE TIM, September 2021

[6] Davis, J, et.al., "A Strategy for Leveraging Domain Specific Data Models", proceedings of the Army FACE TIM, October 2018