2024 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
MOSA TECHNICAL SESSION
AUGUST 13-15, 2024 - NOVI, MICHIGAN

Enabling Multi-Vendor Model Based Application Development Using the FACE™ Technical Standard

Mark Snyder[1], Mark McBroom[2] Kirsten McCane[3]

[1]L3Harris, Palm Bay, FL
[2]The MathWorks, Novi, MI
[3]The MathWorks, Washington, DC

ABSTRACT

This paper discusses how a software development approach based on the Future Airborne Capability Environment (FACE) standard and COTS model-based development tools can enable modular and open software applications to be rapidly developed and deployed in a manner strongly aligned with the Army's Ground Combat Infrastructure Architecture (GCIA) objectives. We describe the use of multiple model-based tools for data architecture, software generation, and system architecture, and describe how these tools have evolved to better support open standards. It then describes the methodology used to integrate Simulink with multiple FACE Transport Services Segment (TSS) implementations. The paper discusses the tools and techniques used, the software components involved, and the testing and validation process.

Citation: Snyder, M, McBroom, M., McCane, K "Enabling Multi-Vendor Model Based Application Development Using the FACE™ Technical Standard," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2024.

1. INTRODUCTION

The FACE™ Technical Standard [1] was created to promote the development of portable software applications. It defines a set of architectural segments that can be used to define a common operating environment that supports software portability and reuse across different embedded systems. The standard includes the Operating Systems Services (OSS) segment, which defines standardized subsets of open standard operating system APIs, and includes the Platform Specific Services Segment (PSSS) and IO Specific Services Segment (IOSS) that together define standardized means to access hardware resources. The Transport Services Segment (TSS) provides a standardized method for applications to access data interfaces in software programming languages. The Universal Domain Description Language (UDDL) standard [2] standard provides the

means to define data exchanges that can be tied to software UoPs (Units of Portability) within the system that use the TSS to communicate (Figure 1). As the FACE technical standard continues to mature, its adoption can be assisted by tools that enable users to implement the standard in their engineering workflows more easily.

1.1. FACE Tool Ecosystems

Many commercial tools exist that support the FACE standard and MOSA development using FACE. For instance, to develop model-based UoPs, MathWorks tools are a good choice. MathWorks joined the FACE Consortium and is maturing its support for

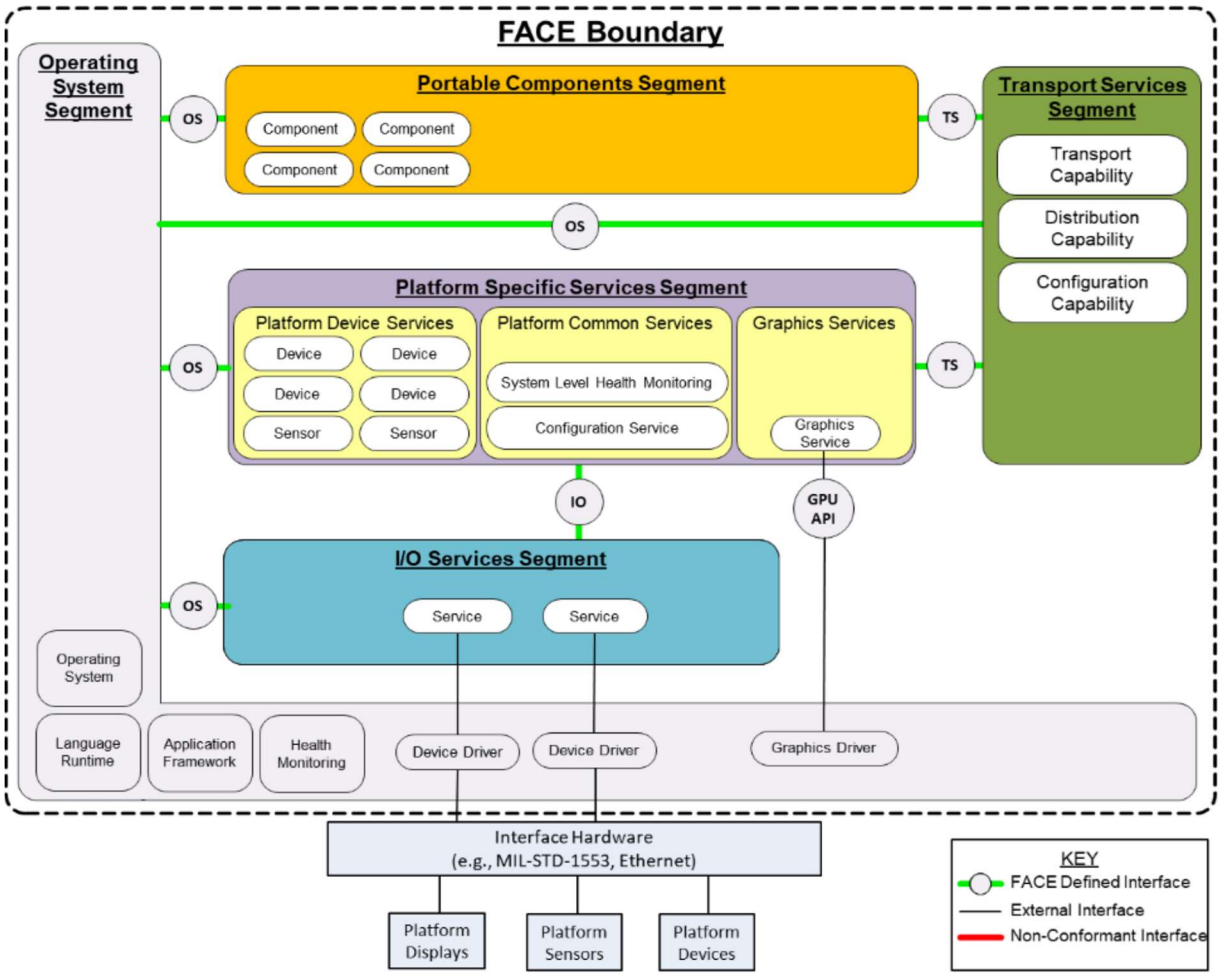

Figure 1: FACE Canonical Reference Architecture.

FACE is a widely adopted, consensus-based standard that has evolved over several years through the Open Group Consortium. Government, industry, software, and tool vendors all attend and contribute to the standard and offer products that aid adoption.

integrating FACE-aligned software components with other avionics systems starting with Simulink. Simulink is a tool used in a variety of industries, including aerospace, to develop and test complex systems. It is a model-based software tool developed by MathWorks that is commonly used for modeling, simulating, and analyzing dynamic systems. For example, Simulink can

be used to model and simulate the behavior and interactions between a set of FACE - aligned software components representing platform functions such as avionics systems, fire control systems, vehicle electronics systems, and other functions which use or control these. Given its industry use and capabilities, MathWorks and its family of products are in a great position to help users more easily develop and implement FACE - aligned software applications.

Other tools are used to support FACE application development that are also presented in this paper. Operator HMI development tools, middleware and TSS tools, and simulation tools are also discussed in this paper.

2. MOSA Digital Thread

Modular Open Systems Architecture in practice is concerned with producing capabilities that are modular, loosely coupled, have well-defined interfaces, minimize integration complexity, and promote reuse. MOSA starts with a strong architectural foundation that is grounded in digital engineering principles. Within the architecture, capabilities become components with well-defined interfaces, behaviors and known dependencies. Deployment is standardized onto common frameworks, minimizing investment in infrastructure by allowing reuse. The connectivity from architecture through components is well defined by interfaces and Open Standards such as FACE, and further defined for a domain of interest by objective architectures such as GCIA.

The FACE Technical Standard promotes a modular and model-based design approach that fits within the MOSA digital thread that is designed to minimize complexity, encourage reuse, and improve system integration. The FACE and UDDL standards define a model-based interface definition

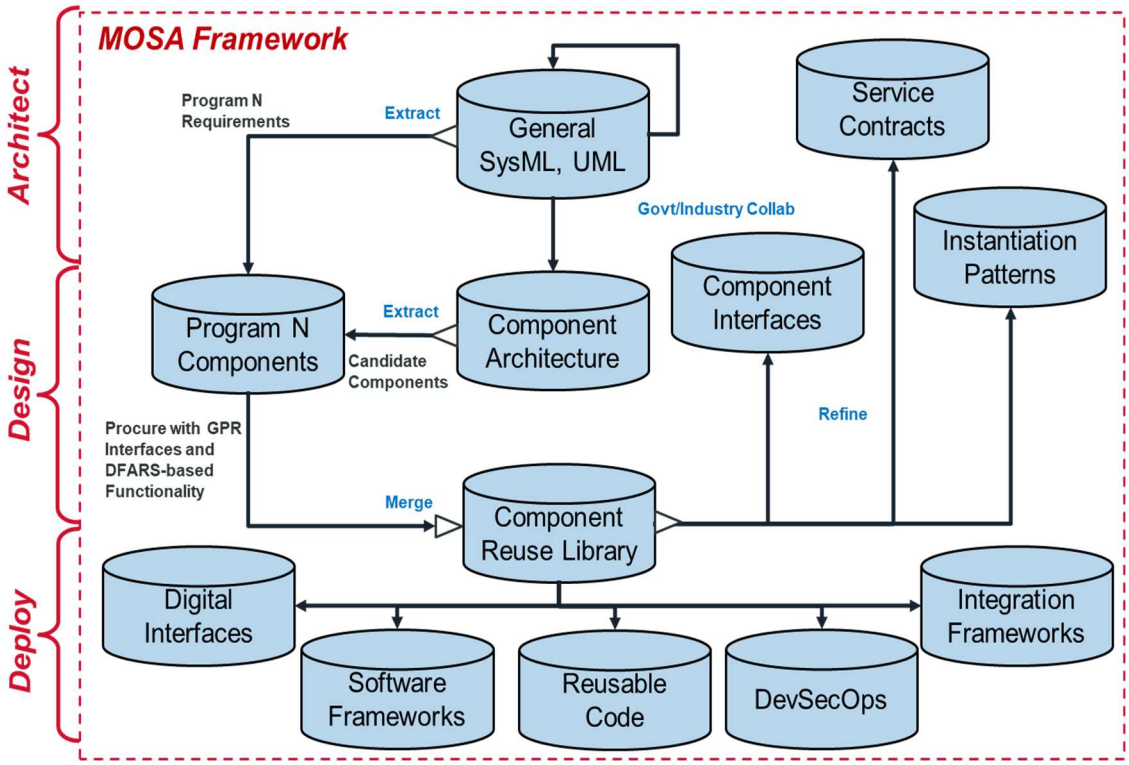

Figure 2: MOSA Development Lifecycle.

approach with standardized means of translating interfaces to software. By breaking complex systems into smaller modules and utilizing a model-based design approach and these standard interfaces, FACE Conformant applications can operate more effectively together, promoting interoperability and modularity, and improving system performance.

Architectural patterns such as Service-oriented architecture (SOA) are enabled by the FACE approach, which promotes loose coupling between software components by encapsulating functionality and exposing it through standardized interfaces. The FACE TSS can provide capabilities such as publish/subscribe and Quality of Service (QoS). In this way, system data needs such as data persistence, data reliability, data security, and advanced interoperability can be implemented, and architectural patterns like SOA can be realized. This aligned well with the Data Architecture and Data Sharing Infrastructure concepts found in the GCIA. The FACE TSS can also be easily integrated with existing open standards such as VICTORY [3].

2.1. Model Based Software and FACE

The FACE and MOSA approach aligns well with the model-based design process that is supported by tools such as Simulink. Using these tools, the system design provides traceability, visualization of algorithm/ logic flow, and early identification of issues in design or code. Within this environment, Simulink provides a variety of integration techniques, enabling engineers to connect and exchange data between components effectively. Additionally, it provides a high level of abstraction to effectively manage the complexity of the software.

Simulink also provides a range of modeling abstractions that help enhance your algorithm model to be suitable for mapping to run-time scheduling and communications that may be provided by integration frameworks and SOAs. Table 1 shows constructs within Simulink that promote modularity.

3. MOSA Software Deployment and Integration

Deployment in a MOSA environment often involves the use of a software framework approach. In embedded systems, model content is often transformed into embedded components such as code or run-time configuration. For instance, code generation may be used to populate software interfaces or middleware data serialization implementations, or to render algorithmic descriptions to embedded code. Code generation from models, facilitated by tools like Simulink and Embedded Coder from MathWorks, can play a significant role in implementing the FACE™ Technical Standard. It enables engineers to translate

Table 1. Simulink Modularity Constructs

Area	Benefit
Model Frameworks and styles	Ability to partition a design for real-time scheduling
Data Busses	Capture interfaces in terms of their data structure
Simulink Functional blocks	Model software services and provide points for integration of code
State modeling	Model application state and provide test frameworks
Messages and interface blocks	Ability to tie to middleware or TSS

design models into executable code adhering to standards. TSS toolchains also offer code generation for the FACE standard TSS interfaces and TSS capabilities that provide data transport.

3.1. Software Tooling and Modularity

Once a software application has been designed in a model-based, modular manner, Model based tooling environments such as Simulink can be also structured to provide end users with flexibility on how they may want to configure and deploy their software applications. For developers creating FACE-aligned software components, an architecture and design consideration is how they may break up or aggregate their application and combine or distribute UoPs as necessary to meet system needs. The Simulink environment allows for components within a model to be generated as individual executables or to be combined with other components to generate one executable. In distributed embedded environments, where an optimal configuration of which components are deployed as UoPs can vary, this can be an essential feature for a development environment.

3.2. Model Based Software and Safety

Model Based tools and a MOSA approach can promote software safety and higher levels of Verification and Validation, which can help to reduce the risk of accidents or incidents caused by software errors. By modularizing the system and adhering to standards such as FACE, system capabilities can identify their usage of Operating System Segment (OSS) profiles that can identify intended deployment scenarios. The FACE OSS Safety Base profile provides a means for software to limit its OS footprint to ARINC 653, while the FACE Safety Extended profile offers more multi-threaded features an application may use. The FACE Conformance Test Suite (CTS) is leveraged to assess the software's conformance to that profile. the FACE™ Technical Standard provides standardized programming language guidance for languages such as Ada, C, and C++ that can enable easier verification of safety objectives. As with other model-based software advantages listed, software environments such as MathWorks can help meet safety objectives for software. Model-based design, verification/validation (V&V), certification artifact generation, compliance reporting, and embedded code generation with safety features such as redundancy and fault tolerance are some features that can help.

4. MOSA Model Based Example

To illustrate the advantages of a MOSA model-based software approach involving multiple vendors using open standards, a demonstration application was constructed. In this demonstration, a component of a ground vehicle (an EOIR sight system) was developed and integrated by a cross-industry team consisting of 5 different vendors. Each vendor made use of the FACE technical standard and model-based approaches.

4.1. System Architecture

The system architecture for the demonstration was oriented around the definition of several components with well-defined interfaces defined in a FACE/UDDL data architecture model. The components were all pre-existing, and had some level of interface definition using the FACE standard already defined. Some additional architecture and modeling work was conducted to form the system into an architecture that illustrated a MOSA system that illustrates aspects of the GCIA such as usage of FACE TSS, well-defined data modeled interfaces on portable components,

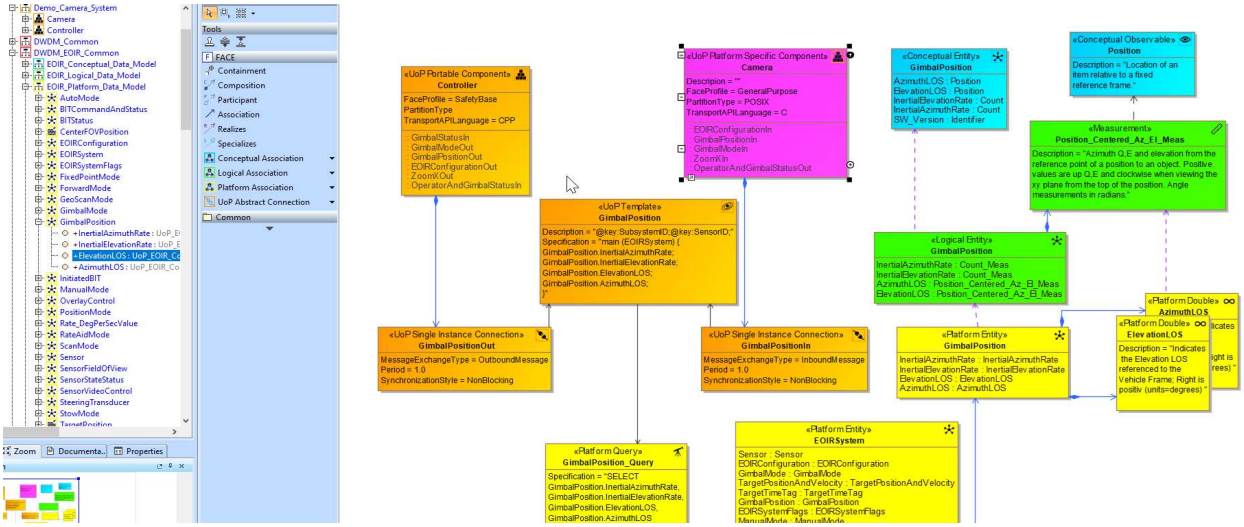

Figure 3: MOSA Demonstration Architecture FACE and SysML Model Views

a software portability and modularity framework, and data distribution patterns such as publish/subscribe.

An example model illustrating the data interfaces and Units of Portability (UoPs) along with an underlying UDDL domain model for the sensor system is shown in Figure 3. Data modeling for each interface was conformant to the FACE technical standard, and FACE Consortium tools were used to assess conformance and to generate interface definitions from the models. Each component of the demonstration was developed by a separate team using the standard, and integration was only performed at the end in order to assess the results.

4.2. EOIR Controller Component

To demonstrate support for the FACE standard MOSA approach, Simulink was

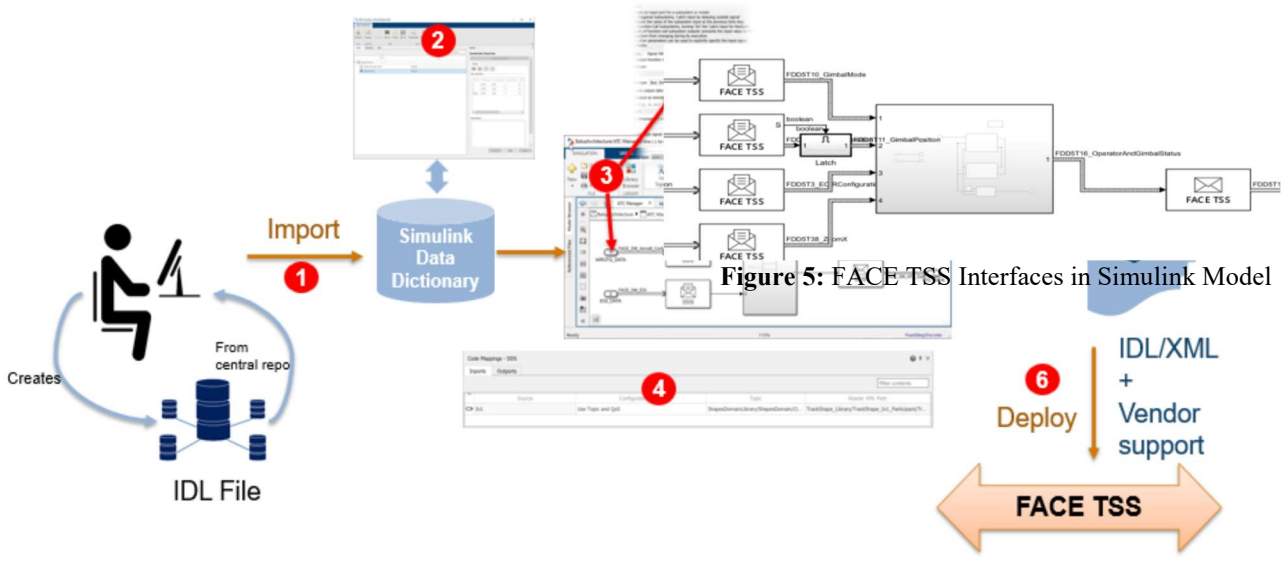

Figure 4: FACE driven workflow for EOIR Camera Controller Module

Figure 5: FACE TSS Interfaces in Simulink Model

selected to develop the camera controller UoP using the workflow in figure 4.

In steps 1 and 2, the model's data dictionary was defined. Using model-based transformations from the FACE model, IDL was generated, which Simulink can import. This IDL becomes viewable as a data dictionary in Simulink, which can be used as FACE TSS interfaces to modules in the model (Figure 5).

In steps 3 and 4,, the EOIR interface and internal components were created as Simulink model elements. The Simulink model contains both a closed loop control of camera position and a model of the 2-axis camera gimbal, allowing for closed loop simulation and testing in Simulink prior to code generation.

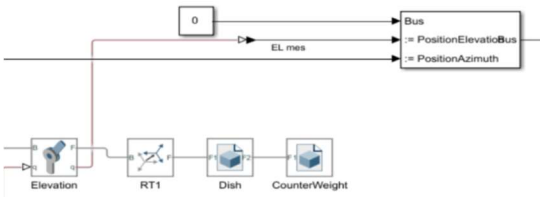

Figure 5: Camera Control Model Elements

In step 5 and 6, Simulink's FACE and embedded code generation facilities were used to generate the UoP implementation with FACE APIs, including the TSS Injectable Interface and Lifecycle Management (LCM) APIs. The resultant UoP is now ready for integration.

4.3. Crew HMI Component

The Crew HMI component allows an operator to control the EOIR system as part of a mission. The HMI itself consisted of several UoP components deployed to a crew HMI framework based on FACE graphics services and FACE TSS interfaces. Each HMI module was independently developed. One key module was the Situational Awareness UoP, based on the L3Harris FliteScene™ Digital Moving Map UoP. FliteScene is a FACE conformant map application that includes support for all common mission database types, as well as the ability to display tactical symbology, routes, sensor coverages, mission plans, terrain, and many more features.

Figure 6: Demo Operator HMI

The HMI graphical elements were created in a model-based tool environment (Protos UP) that allows graphical assets to be created and imported, and dynamic animations applied. Similar to Simulink, data interfaces to the elements and to state-machine logic for the HMI were designed to interface with data interfaces from the FACE model, and the code-behind logic was developed as FACE UoPs with TSS interfaces for all data. (Figure 7)

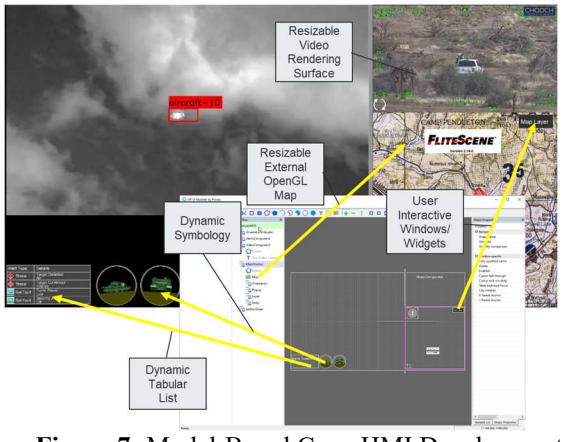

Figure 7: Model-Based Crew HMI Development

5. MOSA Integration

After the components were developed by the various teams, integration was performed. For the demonstration, integration was done using two separate FACE TSS implementations to show modularity. One TSS implementation was from Real-Time-Innovations (RTI) that used OMG Data Distribution Standard (DDS) as the underlying transport. The Data Distribution Service (DDS) is an OMG standard for real-time publish/subscribe middleware that is widely used in many domains including avionics, mission systems, automotive and industrial applications. DDS data types are defined in a standardized Interface Definition Language (IDL) that prescribes data types in a programming language-independent manner. These data types are typically sent or received from topics, which represent publish or subscribe connections through the DDS distributed data bus. Simulink and RTI DDS have a history of supporting ground vehicles, automotive applications, and associated weapons system, so this support was relatively well-developed.

For the second TSS, an L3Harris developed implementation based on open-source non-DDS message middleware was used. This implementation made use of socket-management libraries that create data distribution patterns between endpoints, and was in use on much of the existing simulation integration infrastructure described in the next section.

Both the EOIR and the HMI UoPs were used in the demonstration integrated with both TSS implementations at various times and for various purposes. The UoPs themselves are agnostic to which TSS is supplying the data distribution functions. In some cases, the TSS implementations were used together, with interoperability components used to bridge between middleware solutions.

For the Crew HMI, the software application was deployed to an embedded target system using FACE graphics services and a FACE OSS. The HMI software itself used Vulkan and OpenGL SC as interfaces to the GPU for rendering and video conversion. All software used for the crew HMI is fully aligned to the FACE standard, and several components of it have already passed FACE conformance.

5.1. MOSA Simulation and Validation

Conformance of the MOSA components and interfaces was done at multiple stages of development. Part of the demonstration objectives for this effort included utilizing the FACE CTS to assess the generated MathWorks UoPs and EOIR Data Model. One feature of the L3Harris Cameo FACE tool environment is continuous validation of the FACE model during editing. The FACE CTS Data Model Validation Tool (DMVT) and IDL generation tool were used in this stage to assess the data model, with no reported issues found.

To assess the performance of the generated UoP components, the team implemented a demonstration system employing the TSS implementations and observed the performance of the generated models within a simulated environment. To assess the correctness of the generated TSS and UoP software components, the team used TSS data capture tools to monitor TSS connections and log data messages. These were used to examine messages from the interface in various system states.

For system-level validation and mission effectiveness assessment, the demonstration components were deployed in a MOSA Mission Simulation Integration (MSI)

environment assembled by L3Harris and employing other third-party components that also align to FACE. MOSA Live/Virtual/Constructive simulation was done using an environment called FLAMES™ from Ternion™. FLAMES supports the use of composable elements in simulation models that closely reflect the way actual systems are created and fielded. Simulation actors are constructed from platforms, sensors, effectors, data processors, and AI logic mirroring human decision processes. All levels of the simulation can have custom models implemented at various levels of fidelity. In addition, any required interface can be used with custom models, FACE TSS in our case integrated into FLAMES as custom data processor models mirroring actors on the In-Vehicle-Network.

For the MSI, a highly modular LVC simulation model of a ground vehicle was used. FLAMES supports the concept of variable fidelity, and, for this demonstration, high-fidelity representations of elements of the ground vehicle and its mission, such as turret/fire control, EOIR payload control, and high-fidelity terrain were used. The EOIR payload controller UoP was integrated into FLAMES as an equipment controller model for the EOIR sensor. All communications into and out of the LVC simulation environment to other components used only FACE messaging over the TSS. Other Distributed Simulation Protocols such as DIS or HLA were not used. This approach ensured fidelity and proper interface validation for the UoPs that were being demonstrated and validated.

The MSI integrates high fidelity terrain, ground vehicle movement, and sensor simulation through the use of Unreal Engine as a component of the simulation environment. An embedded streaming video actor simulating the sight was used to create a low-latency networked video stream that was displayed by a video display UoP within the Crew HMI. This Modular MSI approach is shown in Figure 8.

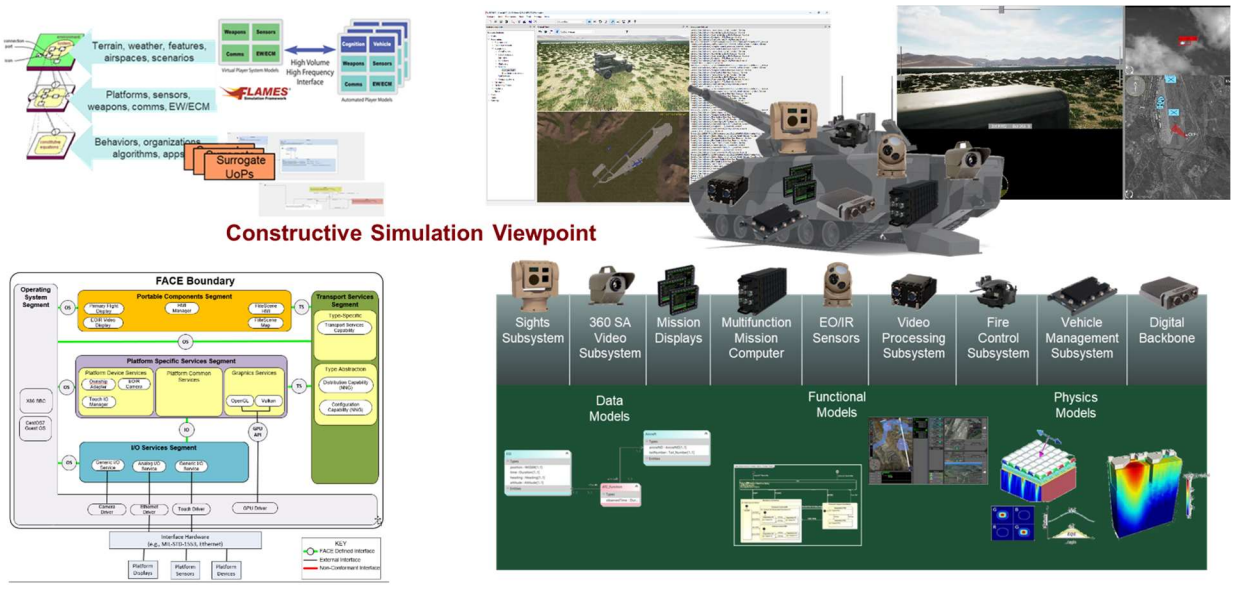

Figure 8: MOSA Mission Systems Integration for Ground Vehicles

6. Conclusion

This paper has described the use of the FACE technical standard and model-based engineering, from architecture through to component development, coding, integration and validation, in a multi-vendor demonstration highlighting the power and effectiveness of MOSA when applied to all levels of the lifecycle.

A system model and FACE data model of the system's key interfaces was constructed and used as part of an architecture based on the FACE standard and reflecting the goals of the GCIA. Components of the demonstration were independently developed, verified, and used model-based methods to ensure rapid development and adherence to standards. Integration, verification, and mission effectiveness was assessed, all within a MOSA simulation environment based on Open Standards. The overall solution illustrates several aspects of the MOSA digital thread, and served as a good environment to investigate concepts of the GCIA ground vehicle architecture.

7. REFERENCES

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

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

[3] Snyder, M, Allport, C, "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, 2023

[4] 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