Using VR to Validate and Visualize MBSE-Designed Interfaces

Sean Flanagan, Jake Bolton, Hunter Stinson

Integration Innovation Inc.

Huntsville, AL

Sean.Flanagan@i3-corps.com, Jake.Bolton@i3-corps.com, Thomas.Stinson@i3-corps.com

ABSTRACT

Several unique challenges arise in the new field of integration between Model-Based Systems Engineering (MBSE) models and game-based digital twin visualizations. First, no known standardized interfaces between the MBSE model and game engine visualization are defined, which can lead to custom or stove-piped solutions. Additionally, visualizing digital twin models in true to life scale is insufficient with typical desktop computers. The Bi-Directional Interface Requirements Operational System Test (BIFROST) prototype, funded by the US Army Program Executive Office (PEO) Aviation, seeks to address these challenges.

Progress of the BIFROST prototype is covered in the paper. The prototype aims to determine the feasibility and challenges of validating interfaces through visualizing changes to an MBSE model in a 3D game engine. Research was performed to visualize part of an uncrewed aircraft system ground control station in 3D using the Unity game engine. A Mission Control Architecture MBSE model, developed by PEO Aviation, is used to drive the digital twin of the ground control station through a set of virtual reality (VR) controls. Users can visualize, analyze, and test the human-machine interaction of the 3D models in VR prior to real-world system changes. This paper presents the recommended interfaces between the MBSE model and 3D engine, lessons learned, and future research areas.

ABOUT THE AUTHORS

Sean Flanagan is a software and systems engineer with experience in multiple defense systems domains. His focus is on Software Systems Architecture and MBSE. He is a Sr. Model Based Systems Engineer at i3 supporting PEO Aviation's Uncrewed Aircraft Systems Technical Management Division (PMUAS TMD). Sean earned a B.S. in Computer Science from Oklahoma State University and a M.S. in Software Engineering from Southern Methodist University.

Jake Bolton is a software engineer with experience in command-and-control software for uncrewed systems and XR-focused software. He is a UAS/VR Software Engineer at i3, supporting many efforts in expanding and starting different projects to support AR and VR and assisting with different simulation efforts. Jake earned a B.S. in Computer Science from Jacksonville State University in Alabama.

Hunter Stinson is a software engineer with experience in modeling and simulation, uncrewed systems, and Extended Reality (XR) devices. He is currently a Senior Technologist at i3 in Huntsville, AL where he helps drive technical solutions in the training and simulation domains. Hunter earned a B.S. in Computer Engineering from the University of Alabama in Huntsville and an M.S. in Software Engineering from the University of Alabama in Huntsville.

Using VR to Validate and Visualize MBSE-Designed Interfaces

Sean Flanagan, Jake Bolton, Hunter Stinson
Integration Innovation Inc.
Huntsville, AL
Sean.Flanagan@i3-corps.com, Jake.Bolton@i3-corps.com, Thomas.Stinson@i3-corps.com

INTRODUCTION

In 2018, the U.S. Department of Defense (DoD) released its Digital Engineering Strategy, which casts a vision of using digital engineering processes and tools to modernize systems, decrease development costs, and increase the speed of delivery to the fleet. The strategy outlines five goals: (1) formalize the development, integration, and use of models to inform program and enterprise decision making, (2) provide an enduring, authoritative source of truth, (3) incorporate technological innovation to improve the engineering practice, (4) establish a collaborative environment across stakeholders to perform activities, collaborate, and communicate across stakeholders, and (5) transform the culture and infrastructure to adopt and support digital engineering across the lifecycle (Office of the Deputy Assistant Secretary of Defense for Systems Engineering, 2018). Model-Based Systems Engineering (MBSE) is a popular approach to meeting the objectives outlined in the DoD strategy, with many program executive offices adopting MBSE for both legacy system sustainment and future platform and systems development. In fact, adopting MBSE for large DoD programs has led to a positive return on investment (Rogers & Mitchell, 2021). PEO Aviation, the aviation office for the U.S. Army, has embraced the digital engineering strategy and guidelines from the DoD. Additionally, PEO Aviation has fully embraced MBSE approaches for programs in its purview, including funding research initiatives to improve efficiencies and reduce defects (United States Army Program Executive Office Aviation, 2022). The Bi-Directional Interface Requirements Operational System Test (BIFROST) prototype originated from this initiative.

MBSE brings many benefits once adopted, but no technology is without challenges. One such challenge is validating interfaces, particularly human-machine interfaces (HMI), during the design phase. BIFROST was developed to connect an MBSE model simulation to external applications such as a virtual reality (VR) application. This connection allows a digital-twin representation of the MBSE model to be visualized and manipulated in 3D. HMI testing and validation are realized through VR's true-to-scale 3D visualization and interaction methods. This paper briefly covers MBSE benefits and challenges related to the prototype. The BIFROST approach, concluding results, lessons learned, and further research areas are also covered.

MBSE BENEFITS

MBSE is "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases (OMG Standards Development Organization, 2022)." Put another way, MBSE is a set of processes and tools that center around a model of the system that acts as the authoritative source of truth. Laskey et al. describe a model within a digital engineering environment as "… an approximation, representation, or idealization of selected aspects of the structure, behavior, operation, or other characteristics of one or more other systems. It also describes the assumptions on which the model is based, the communications between the model and the user, and the other outcomes of using the model. (Laskey, Farinacci, & Diaz, 2021)"

Traditional systems engineering follows a 'document-centric' full lifecycle development process, where documents (e.g., System Requirements Specification, System Design Document) act as sources of truth. As system complexity increases, more effort is required to maintain the documents, and the risk of documents not accurately representing the actual system increases.

MBSE follows a 'model-centric' full lifecycle development process, where all information about the system is represented and stored in a digital model or series of models. A unified model provides better transparency into the overall structure of the system. Modern systems, from aircraft to automobiles, are so complex that identifying and resolving inconsistencies through system checks are paramount. Costly repairs or modifications result if these checks

are not performed or performed late in the development cycle. Implementing MBSE, with its model-centric process, makes identifying inconsistencies easier (Akundi & Lopez, 2021).

Most MBSE efforts use a modeling tool such as Rhapsody or Cameo to make, update, and store the model. These tools are Systems Modeling Language (SysML) compatible and allow for the construction of diagrams from data elements to represent the system at all levels (e.g., enterprise, behavioral, functional, logical). Many MBSE tools can automate the generation of artifacts reducing the time required to maintain and update specific documents. For example, when the system architecture changes and specific diagrams in the model are updated, the tool can automatically generate a new system design document with updated diagrams. The stored model can be validated via built-in simulation capabilities or integration with an external application for software or hardware in the loop simulation capability. Formally validated code can be generated based on the model using code generation capabilities.

Additionally, government customers get a transparent view of the status and design of the system, with direct access to the model at any point in time. Requirements changes can be fed into the model for enhanced traceability and immediate feedback on how a requirement change affects the system. For example, take a situation where new hardware is injected into a system with different size, weight, and power characteristics than previous hardware. Those size and weight dimensions can be adjusted and validated against the model to see if the new hardware exceeds maximum system weight, size, and power limitations. Model reuse is another benefit to MBSE, where existing models can be imported or shared amongst varying projects for cost and time savings.

MBSE CHALLENGES

While MBSE brings many benefits, no tool, technology, or process is without challenges. The BIFROST proof of concept seeks to address four challenges within MBSE: (1) process and tool adoption, (2) extended duration in the design phase delays system validation, (3) interface validation early in the development lifecycle, and (4) effort involved in creating digital-twin models and synchronizing with the MBSE model.

(1) MBSE process and tool adoption across an enterprise often face cultural resistance. In many cases, technical challenges in MBSE are less of a problem than cultural challenges. MBSE changes typical systems engineering workflows and requires additional training and familiarization with new processes and tools. Change resistance is a common challenge in widespread MBSE adoption, and finding and training qualified staff with MBSE experience takes time and capital investment (McDermott, et al., 2020).

(2) A critical challenge faced in MBSE projects is the extended duration spent in the development phase, leading to a delay in the testing of system components. As MBSE relies heavily on modeling and simulation, it often takes significant time to build and refine the system models before verification testing can be conducted. This delay in testing poses technical risk, as the longer the system goes without such testing, the greater the uncertainty and potential issues when it eventually undergoes verification testing.

Testing and verification in MBSE projects encompass various methods, including software in the loop (SWIL), hardware in the loop (HWIL), human-machine interaction (HMI) studies, and flight tests for aerospace applications. These tests evaluate the system components' performance, functionality, and safety. However, due to the extended development phase in MBSE, integrating the system components may occur much later, making it difficult to conduct comprehensive physical tests earlier in the development cycle.

This delayed verification and validation of components can lead to several risks. Early physical testing makes it easier to identify and rectify potential issues, such as unforeseen interactions between components or performance deviations. The absence of proper validation may result in inadequate system behavior, compatibility problems, or even safety hazards. It also hampers the ability to accurately assess the overall system performance, making it difficult to make decisions regarding system design improvements or necessary changes.

(3) Interfaces between systems components are defined in the design phase but may not be thoroughly tested until later in the development lifecycle. Interfaces in this context refer to software-defined interfaces between components and hardware interfaces, including human-machine interfaces. Interface design is typically performed early in the development lifecycle to establish the necessary connections and interactions between system components. Human-

machine interfaces (HMI) are a crucial design consideration in any complex system. HMIs include but are not limited to traditional 2D user interfaces found on desktop computers or mobile devices, virtual or augmented reality devices, and custom-fabricated hardware systems (e.g., buttons, switches, touchscreens). Poor HMIs can lead to injuries, unused capabilities, and damaged equipment (Gruhn, 2011). For example, take an existing military platform that contains a weapons capability. Executing the weapons launch procedure correctly requires what operators describe as needing a 'third hand.' This type of HMI issue may not be realized until device fabrication late in the development cycle. A virtual testing and evaluation environment, if available during the design phase, would have allowed engineers to test and identify HMI interface issues like the one described.

(4) Another challenge in the MBSE domain is synchronizing digital twin 3D models with the MBSE model. One of the prevailing approaches within the MBSE community is to capture extensive digital representations of real-world systems and phenomena that can be used to run end-to-end simulations. These digital twin systems are the gold standard of digital engineering and allow engineers to perform unlimited analyses and trade studies to narrow the solution space before full system development begins. While this paradigm seeks to lessen the costs of development and test efforts, much of the burden gets shifted to the design phase. Detailed high-fidelity digital models are required, not only for the system being developed but also for the systems it will interact with. This includes human interface modeling that attempts to capture simulated user input. If such models do not exist or are not readily available for re-use, a budget must be set aside to create the digital environment in which the proposed system will interact. As the system evolves and requirements change, updating and synchronizing the various models to reflect these modifications accurately is essential (Chami & Bruel, 2018). The primary caveat to this method is that the ability to test the system of interest will only be as good as the fidelity of the modeling. Any estimations or errors in the digital twin representation get propagated into the test.

APPROACH

For our proof-of-concept demonstration, we created the (BIFROST) prototype that allows the bridging of two tools that previously did not have a means of communicating with each other. By doing so, we opened both to validation testing that was previously unavailable to either. This allowed us to use a VR environment to stimulate our modeled system within the modeling tool and provide immediate visual feedback when testing the interfaces between the two. Figure 1 shows the nominal communication paths between the tools.

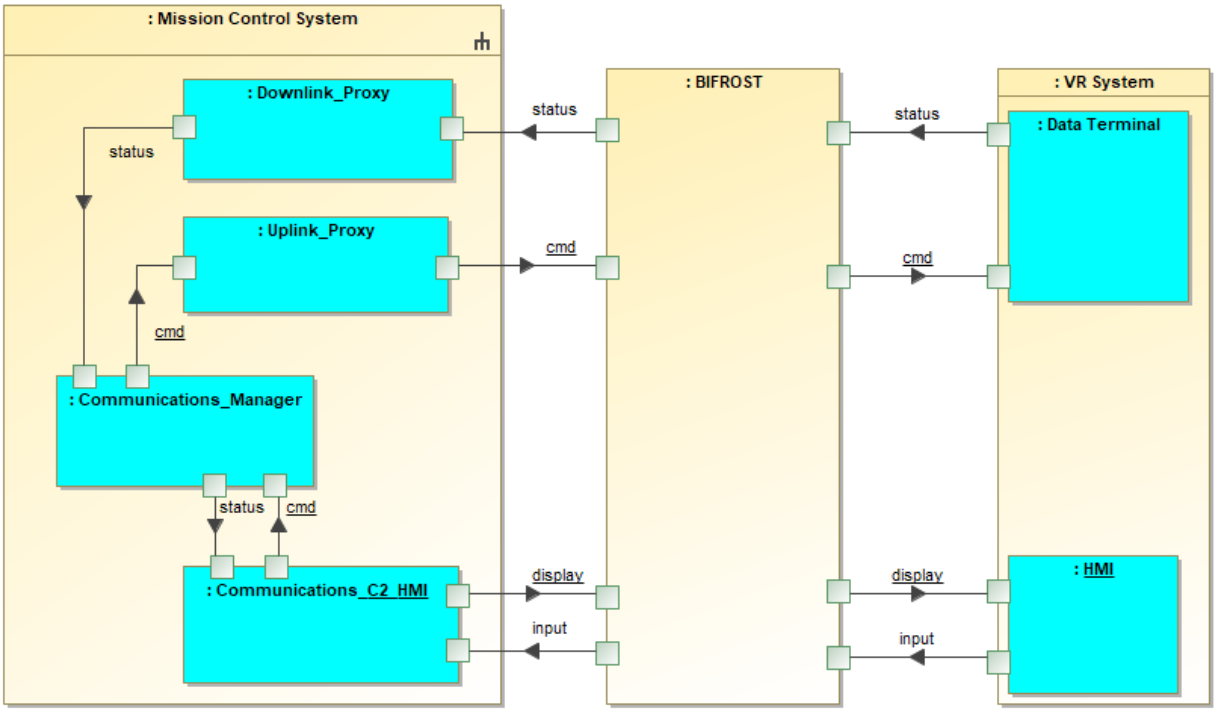

Figure 1: System Data Flow

The starting point for this effort was a Mission Control System Architecture model. This MBSE model was previously created as the design template for developing an Uncrewed Aircraft System (UAS) control station software. One of the unique details of this project was that the architecture would be handed over to be developed by a vendor other than the MBSE engineers creating the architecture. While it is normal for a systems engineering team to design a system for a software engineering team to implement, they almost always work on the same project in a quasi-parallel state, with a chief engineer or project manager overseeing the entire process. Feedback and iteration loops exist where the software development can help refine the design as part of their process.

During the implementation of the model, many design elements had to be corrected or modified to the point that the delivered software had diverged from the original MBSE model intended to be the "source of truth." Later programs that also sourced the model incorporated a system engineering phase where the model was updated in parallel with their development efforts.

Although based on extensive subject matter expertise and using existing well-documented messaging standards, no effective method existed for validating the system design. Black box analysis could be performed where inputs and outputs from the system were checked against the message standards. However, that evaluation could not check for continuous data flows where response messages, periodic status messages and acknowledgments need to be sent in the proper order and at appropriate times. Additionally, there was no way to perform white box testing to verify that the data flow within the system was being handled appropriately. Traditionally, these tests would be performed once the software was developed, but there was a desire to perform this type of testing in the design phase.

The model contained the modeled components, behaviors, and the interfaces and messages required for full system operation. For the scope of our study, we chose to take a subset of the architecture model and build a mechanism to test it directly from the design elements. The use case we chose was data terminal control. This part of a control station maintains a direct line of communication with the UAS. This functionality is straightforward and does not require modeling the more complex UAS behaviors. We hope to be able to extend our concept to the entire set of functionality to the point that we could interact with a VR implementation of a UAS or existing UAS simulators. Figure 2 depicts the system context block diagram with the selected components indicated in blue.

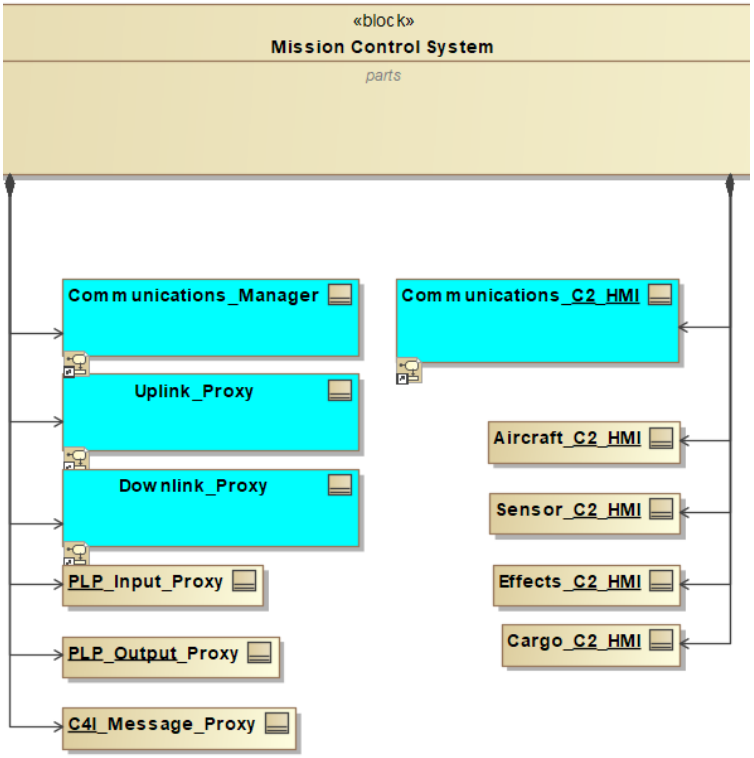

Figure 2: System Context Block Definition Diagram

After selecting the desired scope of our context, we created a separate set of interfaces to cover only the components of interest, which was a subset of the existing ones. By doing so, we could extend the existing model without modifying any existing behaviors or designs. If we extended this effort to encompass the entire system, we could easily replace our interfaces with the original ones. Figure 3 shows the defined interfaces.

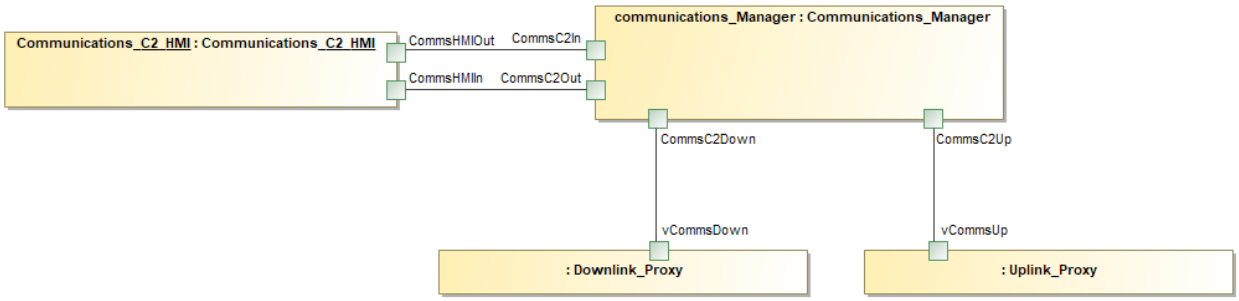

Figure 3: System Internal Block Diagram

The next modeling step was to create activity diagrams that would be executable using the Cameo Simulation Toolkit (CST) plugin created by CAITA Dassault Systems for Cameo Enterprise Architect. This plugin provides the ability to bring a model to life, providing the flow of data within the system, managing state transitions, and allowing a user to step through the execution of modeled system behaviors, much like you would with a software debugging tool. Figure 4 is a sample activity diagram depicting the system's receipt and processing of defined messages. There were nine messages for the data terminal control use case, five going out to the data terminal and four coming back to the controller.

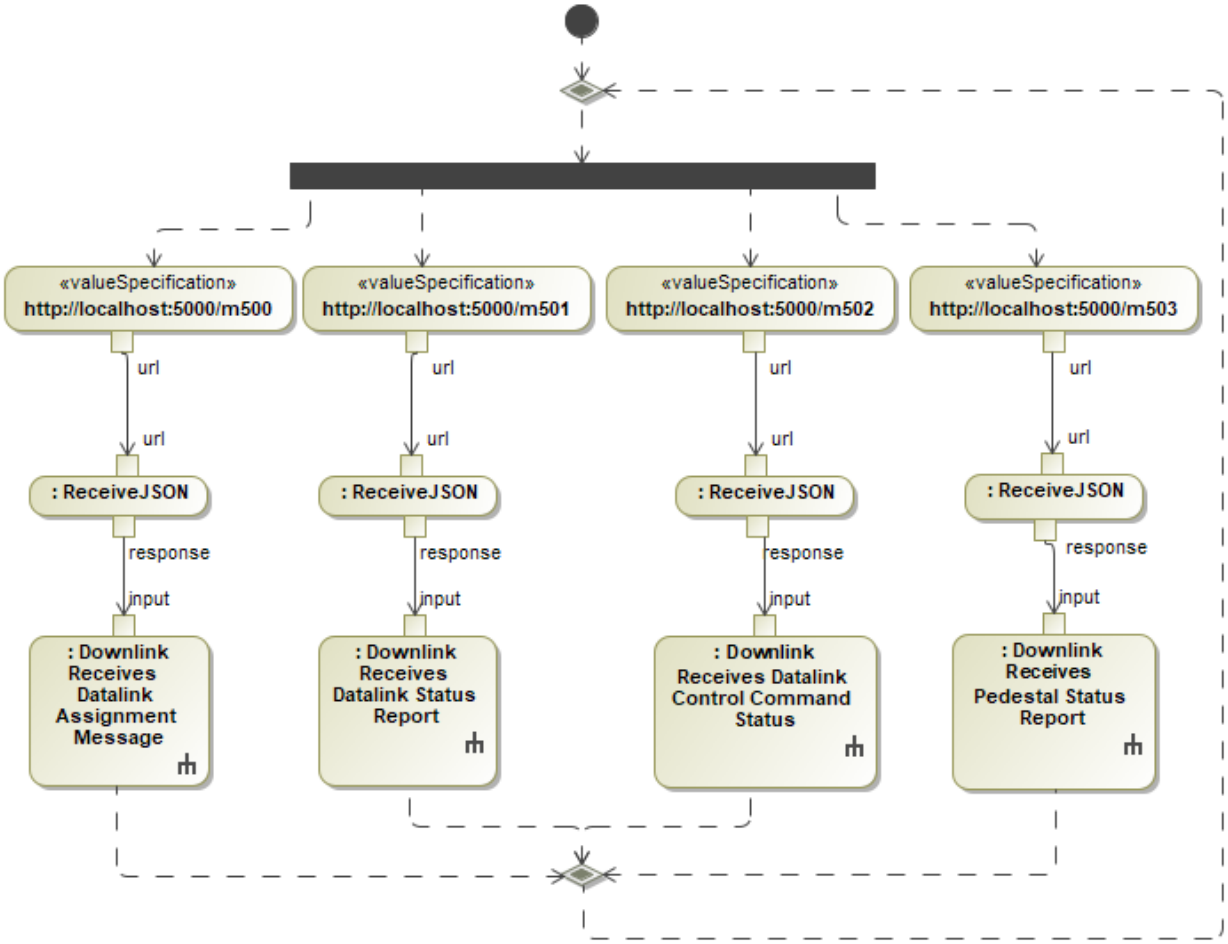

Figure 4: Communications Manager Activity Diagram

Within each activity diagram, opaque actions were used to insert executable code segments, in this case, JavaScript. Opaque actions are used to represent distinct implementation behavior. Our opaque actions created a connection to the BIFROST application and either transmitted or received each message as a JavaScript Object Notation (JSON) string via a Representational State Transfer (REST) interface. Opaque actions also serialized and deserialized the JSON message traffic.

Once the model had been extended to support model execution, it was then able to communicate with the BIFROST application. BIFROST acts as an HTTP server with a pre-defined REST API allowing systems to send and receive data packets. We used JSON strings to deliver the message contents for simplicity, but other message formats could be supported. The VR software established a connection on the other side of BIFROST, allowing the two systems to communicate by acting as a bridge.

The entire behavior of the model was stimulated by the receipt of messages from the "outside world," in this case the VR software. These messages were of two types. The first was in the form of user input via the VR HMI software that was turned into command messages to the data terminal. The second was response or status messages from the data terminal that became data displayed on the HMI. In this way we could verify the round-trip nature of the messages as they propagated through the system and immediately identify if any interfaces were defined improperly. This was especially important when performing coordinate transformations or ensuring that enumerations were properly sequenced.

VISUALIZATION ENVIRONMENT

The BIFROST concept comprises two networked computers and a Virtual Reality (VR) headset. The first computer was connected to the VR headset, a Vive Pro by HTC, and drove the visual experience. In conjunction with SteamVR libraries, the Unity game engine acted as the 3D model visualization backbone. Additionally, the computer hosted the servers that exchanged data between the Unity application and MBSE model. A REST-based API was used to exchange data for the model directly with its opaque functions and a gRPC-based API was used to transmit mouse input, keyboard input, and joystick input. The model's 2D computer screen's video data was sent to the Unity computer using a remote desktop protocol. The second computer hosted the running model with clients to take in and send data to and from the model, replicating the inputs given in the environment on any other application running on the model computer. Figure 5 visually describes the data flow between the visualization system.

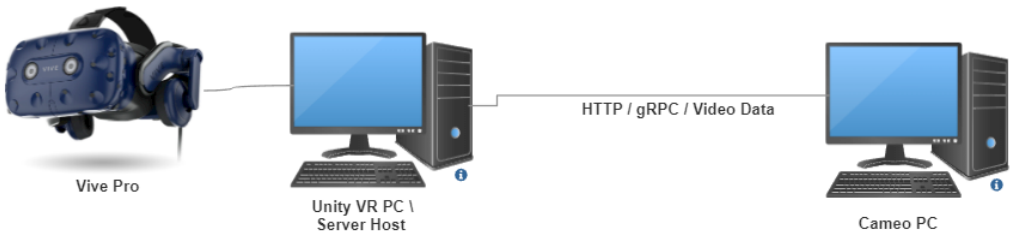

Figure 5: Data and Connection Flow

The environment contains different input methods that replicate the real-world user experience such as a mouse, keyboard, joysticks, dials, switches, warning, caution, advisory (WCA) status mechanisms, indicator lights, etc. Having these components positionally accurate to the real-world equivalents allows a user to go through the motions and learn where things are and where these designated locations need to be improved in the design. With implementation or emulation of screens to show the expected graphical user interfaces, the user can see the system's status and have situational awareness as if using the fully implemented system. This system enables various types of input to be transmitted over a network and replicated or fed into a simulation. These inputs include keyboard input, mouse movements, mouse clicks, joystick data, and more. The purpose of this capability is to allow the simulation of the model to be executed remotely on a separate machine. Once executed, the results can be sent back through the network via screen capture or data format, allowing them to be visually seen within the virtual environment. Figure 6 shows the virtual ground control station re-created in a Unity-based VR application.

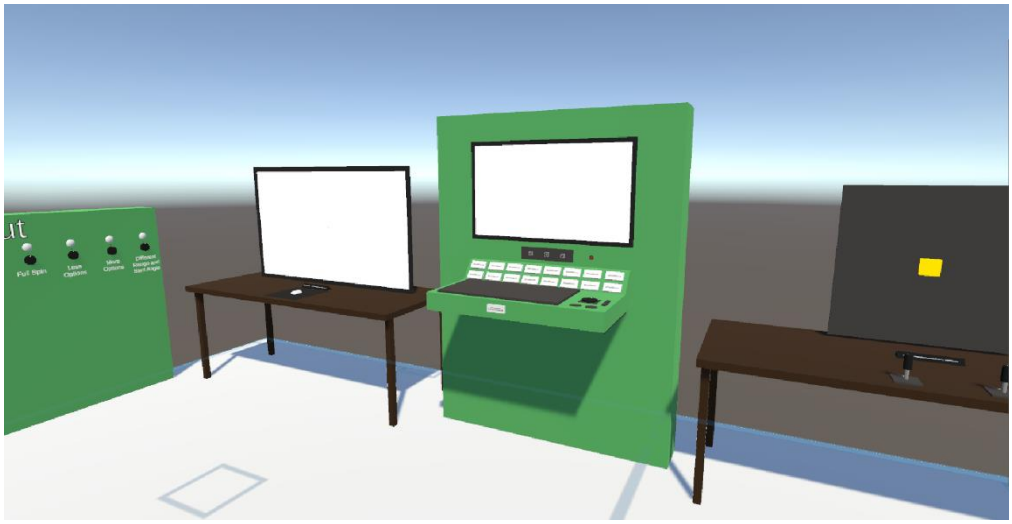

Figure 6: Basic VR Unity Testing Environment

By enabling users to go through the process, it becomes possible to identify and address issues in the flow of physical layouts, and to validate the logic and HMI aspects of the model state mapping to the HMI functions. This includes situations where relevant buttons and GUI elements are excessively distant from each other or placed within separate menus when they should be closer together, issues where warnings or alerts are insufficiently clear, or where key controls are inconveniently located. The aim is to optimize the physical layouts based on these observed difficulties while the end system still exists in its "model" state and has not yet advanced to physical hardware. This style of testing allows for physical changes to be cheap and quick to implement without the worry of retooling, waste of materials, and physical waste of old components. Making small changes to CAD or 3D models and following organized workflows makes it possible to quickly make complex changes in the environment. Changes can be iterative, and improvements are made over time, getting closer to a final usable solution without needing to cut or assemble anything physically. Methods for interacting with the model can be easily changed, swapping keyboard input for a physical switch or a mouse having some functionality swapped over to a joystick. The pseudo physical environment can be expanded upon if needed to allow multiple users to interact with the model and inputs, seeing how in the way another user could be for someone trying to accomplish their tasking.

Changes to the interface can also be iterated on as needed to allow the most useful functionality to be accessed and all the data needed to make decisions is available to the user. Testing a system that handles only one side of the input-output cycle would be impractical. Therefore, leveraging the full capabilities of the model simulation as the backbone, users can perform several actions. They can observe the system's status, make decisions, transmit corresponding actions/messages/inputs into the model, and witness the results represented through various means. These representations may include a graphical interface, a light, a sound, or the movement of a virtual object within the environment. This comprehensive approach ensures thorough testing of the system's scope.

The scope of testing expands significantly through the routine utilization of the virtual environment. The virtual validation of an interface instills greater confidence in the designs of the system. The virtual environment becomes instrumental in identifying issues that future integrators would have encountered much later in the process. Detecting issues early saves substantial time and resources that would otherwise be required for their resolution at a later stage. These tests and experiments can and will expose details that can be overlooked by a designer too deep into the process where they acquire blinders to issues as they are more focused on the next significant change to the model, which could end up with lost data and lack of communication.

The virtual environment allows new users unfamiliar with the system to explore the space safely. No background knowledge of the system is required, and interactions occur much earlier than in traditional engineering workflows. These new users can provide HMI designers with helpful information about what knowledge and assumptions were not translated from the original concept to the current implementation. Regarding the training of mental skills, the review obtained ample evidence for the viability of skill transfer from VR to real deployment (Wheeler, Engelbrecht, & Hoermann, 2021). As the system reaches a more mature and stable set of iterations, training could be achieved for the physical system as well as using the virtual environment to achieve a level of familiarity and knowledge that one could not acquire without being trained on the real-world counterpart in previous projects.

The utilization of MBSE within the virtual workflow has the potential to transform into the primary method of creating final systems. This approach eliminates the need for additional integration or implementation of the model, as the model itself can establish communication with devices or other software to interface with all the necessary inputs and outputs. Consequently, the model becomes a representative system that has undergone rigorous testing and extensive use.

CONCLUSION

The BIFROST proof of concept sought to address four challenges within MBSE: (1) process and tool adoption, (2) extended duration in the design phase delays system validation, (3) interface validation early in the development lifecycle, and (4) effort involved in creating digital-twin models and synchronizing with the MBSE model.

(1) While not directly impacting greater digital engineering adoption in the engineering community, we feel that by eliminating some of the common hurdles, using BIFROST can encourage other programs to make the transition. The

potential cost and schedule savings of early design validation testing make this approach attractive. The ability to agnostically connect with multiple digital engineering toolsets also helps ease the cost of adoption.

(2) (3) Through our proof-of-concept implementation of BIFROST, we successfully verified the modeled interfaces in two critical ways. First, we could trace the data flow within the system between its components. This allowed us to validate the usage of the data elements within the system and inspect it in a true white-box test. Second, we could also view the data externally from the VR representation of the system. This black-box style of testing ensured that the system correctly sent, received, and responded to messages appropriately. This gives the modelers and implementors confidence in the design that did not previously exist. It additionally allows the design team to improve and correct any interface or design issues much earlier in the process, saving development time and avoiding costly out-of-phase rework.

(4) BIFROST supported the early integration testing of MBSE-developed designs by exposing its interfaces to the outside world. The ability to interact with a VR digital twin of a ground control station to perform data terminal operations provided an effective means of validating system interfaces with immediate visual feedback. Conversely, the MBSE model of data terminal control logic allowed the VR implementation to validate its HMI layout and rapidly iterate through user experience (UX) studies. In both cases, the traditional paradigm is to develop the test driver or external system model in the same environment that the system of interest exists in, often by the same people. BIFROST allows us to expand our test environment to incorporate disparate platforms and development teams. This eliminates the need for custom test beds or the modeling of systems that are outside the primary scope of interest. It also allows developers to perform their work in the domains and tools they are most familiar with.

Lessons Learned

During the development of the executable model, great care was taken to ensure that none of the existing design was modified or impacted. The desired approach would be to model a system from the beginning, providing the appropriate hooks to enable model execution. Model execution and simulation is not a new concept. However, few Systems Engineering teams build it into their processes because of added overhead and complexity or unfamiliarity with the capability. We feel the benefits of validating a design through model execution outweigh the implementation costs.

The Cameo and Unity applications used for the research were processor-intensive; attempts to run them on the same system experienced performance degradation. One of the ancillary benefits of BIFROST was that the different pieces of the test system could be spread across computers on the same network, thus reducing the load on a single computer. Performance benchmarking should have been conducted as part of the development process, with some dedicated time to optimize the performance of message processing and polling. With dedicated time spent on optimizing performance, the team feels that all applications could have run on a single computer.

The communication structure between the visualization application and the MBSE model simulation was complex due to using two communication mechanisms (REST and gRPC). While this communication structure was acceptable for a prototype, a single communication medium should be used for simplicity. For example, Web Sockets would have provided full-duplex communication between the Unity application and the MBSE simulation. Additionally, using JSON strings to transmit and receive messages, while straightforward, created some integration issues due to loss of data precision and matching of enumeration literals. Future iterations should consider using protobufs or another format that is not string based.

We feel that our approach could scale to incorporate System of Systems levels of integration and testing through processing optimization and using a common, simplified communication protocol. While conceived as a point-to-point communication bridge, BIFROST could support linkages between multiple systems, whether it be an executable model, VR user interface, or simulation. For example, we envision the modeled Mission Control Architecture interacting with a UAS simulation while controlled via a VR instance of the HMI. With all three systems serving as digital surrogates for yet-to-be-built systems, interfaces could be fleshed out and validated before the implementation phase of development.

Future Research

The BIFROST research effort was the first step in furthering MBSE and visualization in VR. However, there are several areas for future research to expand to the knowledge domain. There are three areas of improvement: advancing the 3D model-based environment, identifying issues and improvements via simulation, and expanding linkages to external systems.

BIFROST envisions a future where model-based simulations provide system engineers and designers with an immersive and realistic experience. To achieve this, further research should focus on expanding the capabilities of the 3D environments within BIFROST. Specifically, including interactive elements such as buttons, levers, and joysticks would enable users to manipulate and control the simulated systems directly. This advancement would give engineers a more intuitive and realistic simulation experience, enabling them to analyze system behavior and identify potential design flaws more effectively.

Building upon BIFROST's existing capabilities, future research should aim to create a system that utilizes the model simulation itself as a tool to identify potential issues. By closely monitoring system behavior during simulations, BIFROST could automatically detect anomalies, performance bottlenecks, or areas where the system deviates from the desired behavior. This proactive issue identification approach would allow system engineers to quickly address potential problems and improve system performance early in the development process, thereby reducing time and costs associated with system rework.

To enhance its usability and enable seamless integration within the broader MBSE ecosystem, future research for BIFROST should focus on creating additional linkages to other MBSE tools, physical hardware, and AI/ML systems. These linkages would facilitate data exchange, collaboration, and interoperability between BIFROST and other critical components of the system engineering workflow. Integration with MBSE tools would enable engineers to leverage existing models, requirements, and analysis frameworks, fostering a cohesive and streamlined development process. Moreover, connectivity with physical hardware and AI/ML systems would enable BIFROST to interact with real-world devices and leverage cutting-edge artificial intelligence technologies for enhanced system testing, validation, and decision-making.

The future research areas outlined above present exciting opportunities for advancing the BIFROST project. By enhancing model-based 3D environments with interactive elements and customizable parameters, BIFROST can provide system engineers with a more immersive and flexible simulation experience. Furthermore, leveraging the simulation to identify issues and expand linkages with other MBSE tools, physical hardware, and AI/ML systems would further enhance BIFROST's usability, collaboration capabilities, and real-world applicability. With these advancements, BIFROST aims to empower system engineers and designers with powerful tools to accelerate system development and ensure the delivery of reliable, high-performance systems.

ACKNOWLEDGEMENTS

We thank PMUAS Technical Management Division for providing the resources to perform this study and explore new ways to bring Digital Engineering best practices to the development of the next generation of UAS.

We thank Dr. Patrick Buckley and Matthew Bower for valuable feedback on the paper’s problem domain and peer review.

REFERENCES

Akundi, A., & Lopez, V. (2021). A Review on Application of Model Based Systems Engineering to Manufacturing and Production Engineering Systems. *Complex Adaptive Systems Conference* (pp. 101-108). ScienceDirect.

Chami, M., & Bruel, J.-M. (2018). A Survey on MBSE Adoption Challenges. *INCOSE EMEA Sector Systems Engineering Conference.* Berlin.

Gruhn, P. (2011). Human Machine Interface (HMI) Design: The Good, The Bad, and The Ugly (and what makes them so). *66th Annual Instrumentation Symposium for the Process Industries*, (p. 6).

Laskey, K., Farinacci, M., & Diaz, O. (2021). *Digital Engineering Fundamentals: A Common Basis for Digital Engineering Discussions.* MITRE Corporation.

McDermott, T., Hutchison, N., Clifford, M., Van Aken, E., Salado, A., & Henderson, K. (2020). *Benchmarking the Benefits and Current Maturity of Model-Based Systems Engineering across the Enterprise.* Systems Engineering Research Center.

Office of the Deputy Assistant Secretary of Defense for Systems Engineering. (2018, June). *Department of Defense Digital Engineering Strategy.* Retrieved from https://ac.cto.mil/wp-content/uploads/2019/06/2018-Digital-Engineering-Strategy_Approved_PrintVersion.pdf

OMG Standards Development Organization. (2022, 12). *MBSE Wiki*. Retrieved from https://www.omgwiki.org/MBSE/doku.php?id=start

Rogers, E., & Mitchell, S. (2021). MBSE delivers significant return on investment in evolutionary development of complex SoS. *Systems Engineering*, 385-408.

United States Army Program Executive Office Aviation. (2022). *Forging the Future of Army Aviation*. Retrieved from PEO Aviation: https://api.army.mil/e2/c/downloads/2021/12/21/8f4c1ef1/peo-avn-fy22-strategic-plan-reduced-v11.pdf

Wheeler, S., Engelbrecht, H., & Hoermann, S. (2021). Human Factors Research in Immersive Virtual Reality Firefighter Training: A Systematic Review. *Frontiers in Virtual Reality*.