Modeling A UAV Surveillance Scenario- An Applied MBSE Approach

Viviana Lopez
Complex Engineering Systems Laboratory
Department of Manufacturing and Industrial Engineering, University of Texas Rio Grande Valley, Brownsville, Texas.

Aditya Akundi
Complex Engineering Systems Laboratory
Department of Informatics and Engineering Systems, University of Texas Rio Grande Valley, Brownsville, Texas.

2023 IEEE International Systems Conference (SysCon) | 978-1-6654-3994-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/SysCon53073.2023.10131074

***Abstract*— As the complexity of both products and systems increases across a wide range of industry sectors, there has been an influx in demand for methods of system organization and optimization. MBSE enhances the ability to obtain, analyze, communicate, and manage data on a comprehensive architecture of a system. In this study, a military combat surveillance scenario is modeled using SysML generating state machine diagrams and activity diagrams using the Magic Model Analyst execution framework plugin. This study seeks to prove the feasibility of an MBSE-enabled framework using SysML to create and simulate a surveillance system that monitors and reports on the health status and performance of an armored fighting vehicle (combat tank) through an Unmanned Ariel Vehicle (UAV). The Magic System of Systems Architect, which actively promotes system development architectural frameworks, was used to construct SysML-compliant models, allowing the creation of intricate model diagrams. The construction of the UAV surveillance scenario emphasized the capability of modifying a diagram feature and ensuring that the alteration is communicated to all linked model diagrams. This study builds on a previously published MBSE-enabled conceptual framework for creating digital twins. The purpose of this research is to test and validate the framework's procedures.**

Keywords—MBSE, SysML, MBSE framework, UAV, Surveillance

I. INTRODUCTION

As defined in the 2007 INCOSE Model-Based Systems Engineering (MBSE) Initiative, *Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification, and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases* [1]. MBSE improves the capacity to acquire, analyze, distribute, and manage data related to a system's in-depth architecture. Attested by several latest studies, MBSE techniques demonstrate enhanced interconnectivity among system stakeholders, allowing a system model to be understood from several viewpoints while assessing the implications of alternative solutions. MBSE adoption has successfully resulted in increased system dependability by providing a clear and comprehensive model of a system that can be examined for stability and reliability. Three pillars facilitate MBSE: a modeling approach, a modeling language, and a modeling tool. A modeling approach is a specified set of procedures that supports system interoperability that allows for consistent system model construction. A modeling language is a formal language that is used to represent, express, and communicate the structure, behavior, or other aspects of a system or process. A modeling language provides a way to describe a system or process using symbols, rules, and syntax that are defined in a formal specification. Modeling tools are software applications that enable the creation, analysis, and manipulation of models using a specific modeling language or notation. These tools are used to create and manage models of complex systems and processes in various domains, enabling users to visualize, simulate, and test different scenarios and solutions. They also help to improve collaboration and communication between team members by providing a shared language and framework for understanding complex systems. Modeling tools can vary in complexity, features, and usability. Magic System of Systems Architect is one MBSE tool that is used in this paper to actively support system development and analyze different system parameters. Magic System of Systems Architect, formally known as MagicDraw, is a tool based on a unique data repository that allows the design of complex multi-domain systems [2]. This software can provide organizations with a better understanding of how changes to a component or subsystem can affect the overall system. To aid users in evaluating potential design options, Magic System of Systems Architect makes use of MBSE methods and a variety of modeling languages. This encourages improved risk management and may result in fewer problems throughout many phases of the system's life cycle. This MBSE tool is just one of many with the features indicated. Technologies like IBM Rhapsody, Capella, Enterprise Architect, and Papyrus, among others, provide features comparable to the Magic System of Systems Architect. There are numerous modeling languages that can be used with MBSE tools; however, Systems Modeling Language (SysML) is used as the primary modeling language of this study. SysML is a comprehensive graphical modeling language that enables the visualization and communication of the main components of system architecture: structure, behavior, requirements, and parametric [3]. SysML was designated as the primary modeling language attributed to the fact that its semantics are more customizable and comprehensive but will be utilized to specify performance and quantitative measures. SysML models were executed by animating state machine diagrams and activity diagrams using Magic Model Analyst, an execution framework plugin for Magic System of Systems Architect [2].

In Section II, the authors explore a theoretical conceptual framework that has been previously established to facilitate the development of MBSE-driven digital twins [4].

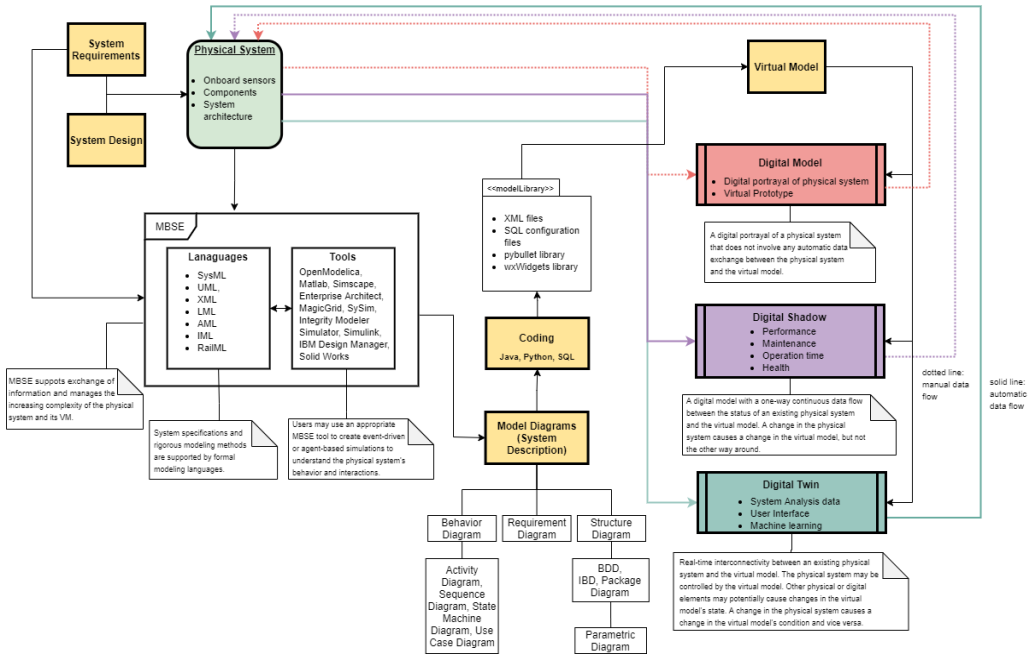

Figure 1. MBSE Enabled Process to Develop Varying Virtual Models from a Physical System [4]

Section III outlines a scenario that was created to assess the practicality of the MBSE-enabled framework, along with the relevant modeling tool and language. Section IV breaks down the surveillance scenario into four segments: problem domain, solution domain, UAV subsystem, and ground control unit (GCU). Additionally, a brief overview is provided on how the simulation of time-based transfer of UAV flight and image data was carried out.

II. MBSE Enabled Framework

In a prior study [4], an effort was made to establish a theoretical conceptual framework facilitated by MBSE for developing a digital twin. To design, develop, and implement a digital twin for a physical system, the framework specifies the development of system requirements before system design or employment (see Figure 1). According to the framework, several MBSE modeling languages can be used to represent system requirements. SysML is identified as the most utilized modeling language, which is used in this study. Modeling tools should then be used to build complex model diagrams that are interconnected and allow for an organized system design and implementation process. Magic System of Systems Architect was used in this study to develop models following SysML, enabling the development of complicated model diagrams. The next step is to create data connectivity through executable program files written in a suitable programming language. Depending on the desired virtual model type (Digital Model, Digital Shadow, or Digital Twin), information gathered from changes in either the physical system or virtual model is implemented manually or automatically. The final virtual model type is determined by the amount and combination of model diagrams utilized and the method by which data is transferred between the physical system and the virtual model [4]. After implementing the initial stages of the framework, the development of the UAV surveillance scenario using Magic System of Systems Architect demonstrated the interconnectedness of modifying a diagram's feature, which in turn updates any other associated model diagrams. This interconnectivity allows for more coordinated and efficient system design and execution. This paper's research evaluated the first two phases of the framework's validity, while the subsequent phases will be the focus of future work.

III. Surveillance Scenario

The focus of this research is to demonstrate the applicability of a previously developed MBSE-enabled framework. SysML and Magic System of Systems Architect were used to model a hypothetical UAV surveillance system that monitors and reports on an armored combat vehicle's health and performance.

The tutorial "Aircraft Radar Display SysML MagicGrid Sample with Simulation and Analysis" by Saulius Pavalkis was used to generate the SysML diagrams depicting the UAV surveillance scenario [11]. The tutorial provided a solid foundation and valuable insights into the application of SysML modeling techniques to an aircraft radar display system. This served as the basis for the development of diagrams depicting the intricate interdependencies and connections present in the UAV surveillance scenario.

The US Army maintains a fleet of ground combat vehicles designed to undertake combat operations against opposing troops. The Congressional Budget Office has estimated the cost of such vehicles until the year 2050. The total acquisition expenditures for the Army's ground combat vehicles are estimated to average about $5 billion per year until 2050 [5]. Traditionally, the Army's armored combat vehicle maintenance standards rely heavily on lengthy manual diagnostic processes [6]. Instead of using automated diagnostic paradigms, present practice only monitors if operational conditions are within the range of acceptability. There is a need for automated real-time monitoring of armored combat vehicles to evaluate ongoing vehicle health and better anticipate vehicle conditions to save both resources and lives.

Any tactical mission's objective is to defend and protect at all costs. However, maintaining and repairing tank units may be both expensive and dangerous if not managed carefully and timely. To minimize servicing time and implement additional safety measures, a UAV is employed to track and monitor a combat vehicle to detect potential changes in the tank's overall physical and structural health status and performance. The UAV will record/capture image data via an onboard camera

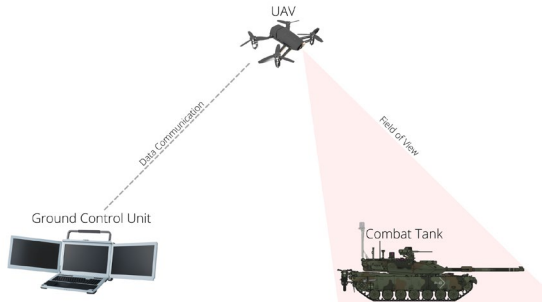

Figure 2. Surveillance Scenario

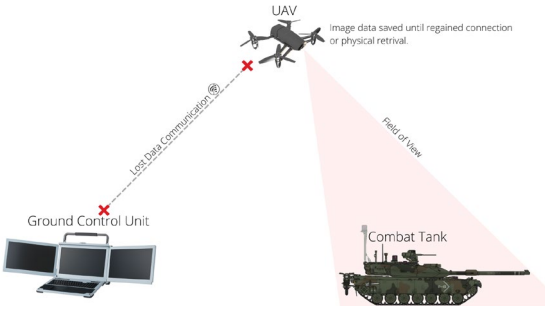

Figure 3. Lost Connection between UAV and GCU

and will maintain a maximum altitude of 200(m) and a minimum altitude of 60(m) from its target to ensure optimal surveillance parameters are maintained. The UAV will communicate to and from a ground control unit, as seen in Figure 2, where a flight operator can make informed decisions about the structural health and status of the target from the imaging data sent from the UAV. The UAV will also transmit data to the operator regarding its own health/battery status and performance. In the event of an abnormality in the tank's operations, the tank operator and the maintenance personnel will be notified and then equipped for unscheduled maintenance. In the event of a loss of communication between the UAV and the Ground Control Unit (GCU), the UAV will continue to track and store imaging data independently, as seen in Figure 3. When a connection is lost, the GCU will alert the operator. Once the link is re-established, all stored imaging and flight data is transmitted to the GCU, along with real-time data. If the UAV's link is lost, it will continue to monitor and capture data from its target until the battery is down to 25% capacity and then returns to its home base.

The modeling of the scenario will be utilized to demonstrate the feasibility of the MBSE-enabled framework (Figure 1) by developing and simulating the scenario using SysML. The scenario will be modeled and simulated utilizing pre-determined optimal UAV flying parameters and weather conditions; no physical experiments were performed. It is assumed that the operator's only engagement with the surveillance systems will be for assigning flight operations and analyzing incoming data.

IV. Developing Scenario Models

A. Problem Domain

The UAV Surveillance Mission was divided into four distinct categories: the problem domain, the solution domain, the UAV subsystem, and the GCU. The initial stage is to break down system information and categorize it according to what information influences each subsystem, the environment, or the mission. This is essential for simulating the transfer of imaging data, UAV health, and flight data among system elements. These elements are then deconstructed into separate requirements which allow information to be sent from the UAV to the GCU and viewed by a human operator. A Black and White Box were created. Black box provides external insights into a system. The purpose of a black box is to develop a thorough and consistent set of requirements to avoid future revisions caused by poor specifications. White box, on the other hand, is an internal perspective of the system in which the system architecture is gradually identified [7]. Critical performance needs might be recorded as value attributes of the system's black box or as flow properties of moving objects. The needed system reaction time may be described as a value property item of the system black box, that flows in or out of the system black box [8]. The Black Box consisted of Stakeholder requirements, Use Cases, System Context, and Performance Metrics (MoEs) (see Figure 4). Functional Analysis, Logical architecture, and system analysis constituted the White box. Table I. displays the requirements included in the Black Box for employing communications between the UAV and GCU.

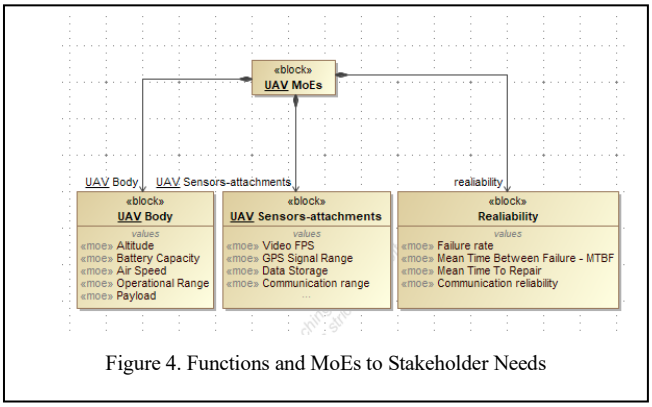

Figure 4. Functions and MoEs to Stakeholder Needs

TABLE I. Mission Communication Requirements

1.1	Imaging data shall display in less than 1s and refresh in less than 0.5s
1.2	GCU shall support the following operation modes: pre-flight, post-flight, UAV surveillance, and warning mode.
1.3	The in-flight mode system shall display the planned trajectory of the UAV on the GCU screen.
1.4	GCU screen shall provide visual and acoustic warning in case of UAV malfunction in less than 2s
1.5	GCU screen shall provide visual and acoustic warning in case of lost connection from UAV to GCU in less than 2s
1.6	GCU screen shall provide visual and acoustic warning in case of lost connection from GCU to combat tank in less than 2s

Figure 5 depicts a package diagram used to develop the use case for the GCU operator. The features for the functional

analysis of the GCU are: *display all data on a screen*, *receive imaging data*, *receive UAV flight data*, *perform operation mode*, and *provide warnings* to the operator and combat tank.

To develop connections between each subsystem so that communication can occur, a BDD was developed with corresponding ports that are referenced across multiple models as seen in Figure 6. Now that the connections have been developed, signals such as 'location data' or 'warning' can be sent between subsystems. For each port, activity diagrams (Figures 7 and 8) representing the operator's response to information received via the GCU were constructed [11].

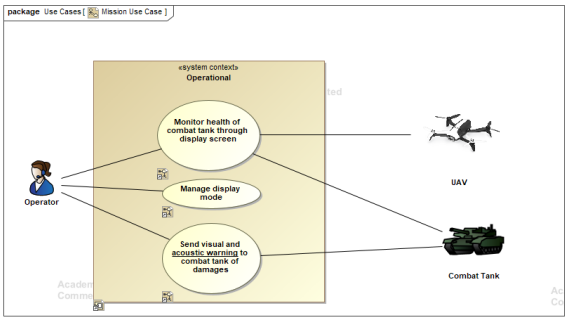

Figure 5. Operator Use Case

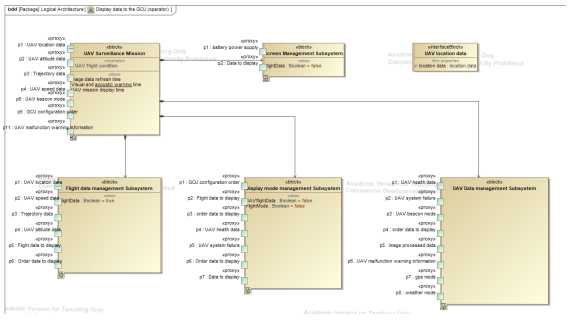

Figure 6. BDD For System Port Connections

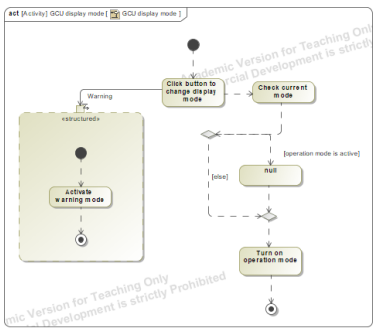

Figure 7. GCU Operation Mode Activity Diagram

These varied characteristics composed the problem domain, allowing for the incorporation of system requirements in the solution domain.

B. Solution Domain

Each action or activity must be meticulously documented with its corresponding requirements so that the implementation of each system adheres to the intended objective as seen in Figure 9, the model's primary purpose is to quantitatively characterize the information given to the

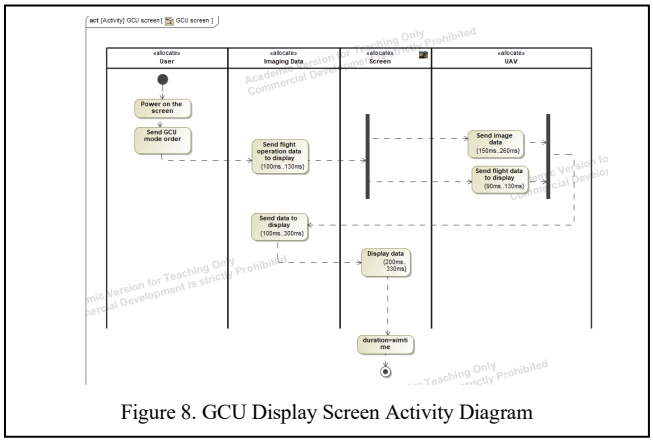

Figure 8. GCU Display Screen Activity Diagram

operator through the GCU. Depending on the system's abilities, various quantities of data representing various quantitative information, such as the time between delivered messages or imaging data, will be sent.

Modeling the system behavior within the scope of the surveillance scenario is the next phase. This stage employs both state and activity diagrams for enhanced customization and adaptability. The mission is subdivided into several states that correspond to distinct parts of the event. The system states and activities are modeled to reflect what is occurring with the UAV during the operation. An example of a system state is having sufficient battery life for the mission and providing a warning if it is insufficient. To describe the relationships between the GCU, the operator, and the UAV, an IBD representing each subsystem block was built (see Figure 10) [11]. Due to the scope of this research phase, information was restricted to the transmission and reception of data between subsystems. The following simulation data depicts the number of milliseconds required for the UAV to transmit image data to the GCU so that the operator may make judgments on the combat tank's health.

C. Simulation

As mentioned prior, the scope of this research was to simulate the communication between the operator, GCU, and UAV. A duration analysis was conducted to calculate the time (milliseconds) each message was sent and how long it was displayed on the GCU, see Figure 11 [11]. These results can also be referenced to the previously made GCU Display Screen Activity Diagram (Figure 8). The information can then be seen on the GCU screen. Figure 12 provides an example of the GCU interface and a type of imaging data that can be sent from the UAV [11].

D. Shared Workspace

Magic Systems of Systems Architect (MSOSA) supports various simulation features. MSOSA provides four different kinds of simulation engines: Activity engine, State Machine engine, Interaction engine, and Parametric engine [2]. In this phase of the case study, the Parametric engine and Activity engine were employed to simulate and model the UAV's flight sequence. In addition, MATLAB® and Simulink® were used to simulate and illustrate the scenario described in section III. Simulink® is a block diagram environment that supports MBSE by providing system-level design, simulation, code generation, and embedded system testing and verification where MATLAB scripts can be integrated into Simulink models [10].

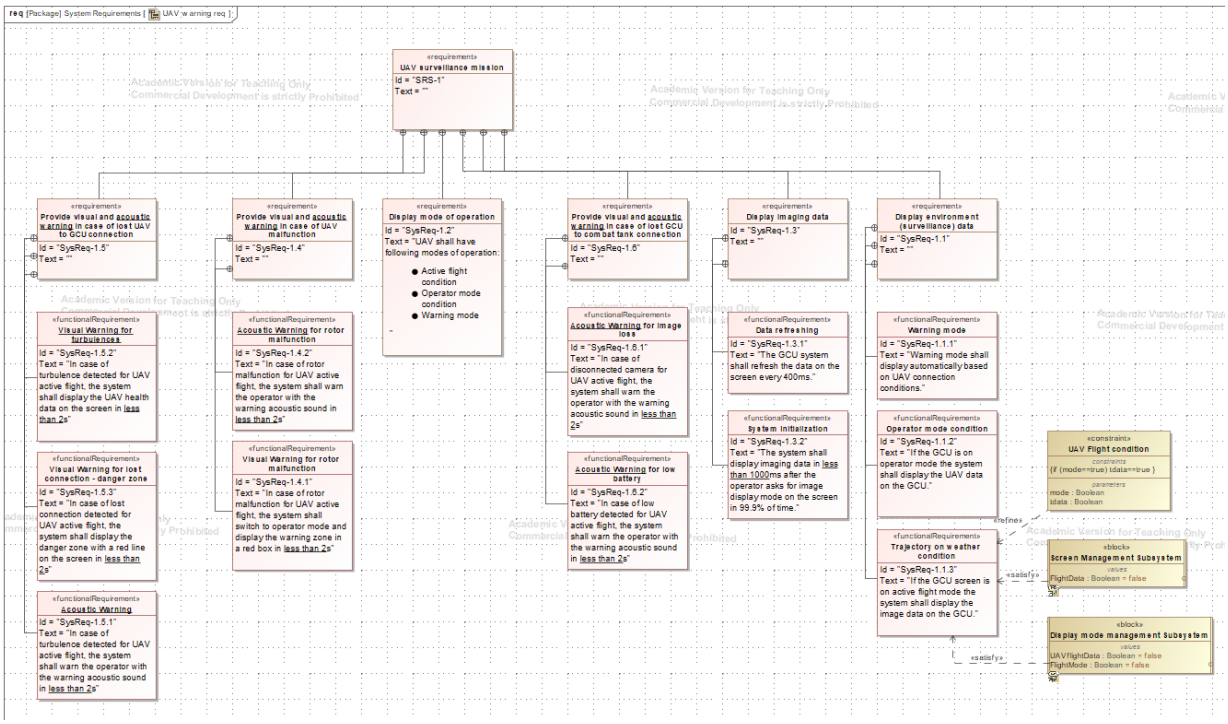

Figure 9. System Requirements

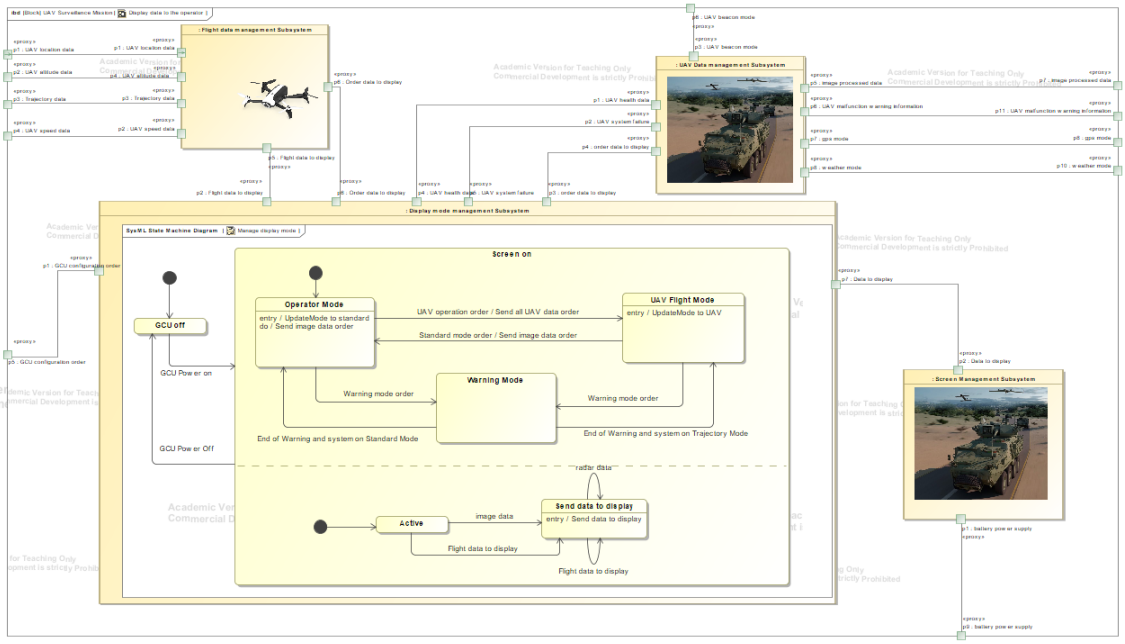

Figure 10. IBD Mission Communication

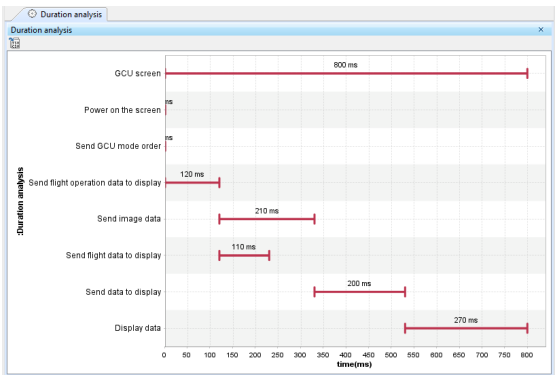

Figure 11. IBD Mission Communication

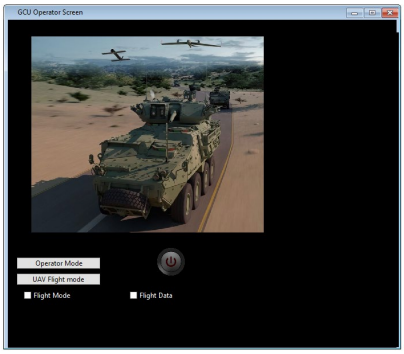

Figure 12. GCU Operator Screen with Imaging Data.

Using coordinate tracking, the drone follows the combat vehicle. While traveling to a predetermined destination, the tank transmits its GPS position to the UAV. The UAV functions by maintaining a fixed distance to these coordinate positions, where the operator will receive incoming flight and image data based on a time interval to verify whether the UAV is functioning properly. For the scope of this case study, the trajectory of the combat vehicle is predetermined. Chun-Wei Kong's 6-DOF (degrees of freedom) Quadcopter Simulation and Control MATLAB/Simulink project laid the groundwork for the simulations that were developed for this case study [9].

In MATLAB scripts, simulation-required UAV flight parameters were generated. MSOSA enables the integration of MATLAB and the establishment of a collaborative workspace. Even though these variables are editable and modifiable within MATLAB, visualizing and defining inputs and parameters expedites model development and assures consistency across many platforms/software. MSOSA recognizes expressions written in MATLAB syntax, which may be modified in MSOSA and imported into saved MATLAB files simultaneously.

A block definition diagram was created to specify and visualize certain parameters in the previously established MATLAB code files. As shown in Figure 13, four blocks were incorporated into this diagram: *test*, *A_SetDroneControl*, *C_XYZSignal*, and *E_animation*. The first block 'test' was constructed to verify that a shared workspace was established correctly.

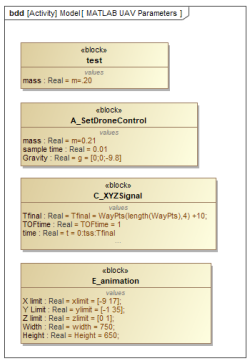

Figure 13: Block Definition Diagram, defining MATLAB code

As shown in Figure 14, the test block is separately chosen and simulated to verify the shared workspace. MSOSA will create a shared workspace with MATLAB after the simulation has begun, and the new mass should be represented in the corresponding file as seen in Figure 15. This shared workspace ensures interoperability by allowing a user to make a modification on one platform and have it simultaneously updated on another. Not only does this save time, but it also ensures that all parameters, values, and inputs stay constant throughout product and system development. Once each block has been independently simulated, resulting in updated values in the appropriate MATLAB code, an activity diagram was created to begin the required processes for executing all the MATLAB scripts to provide a simulation output.

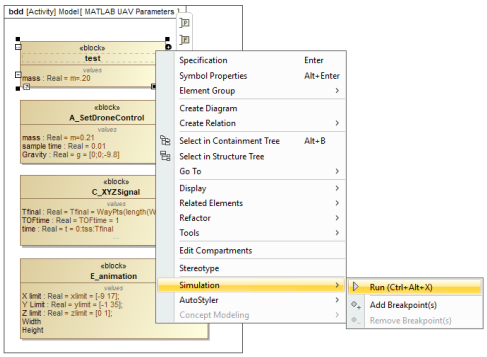

Figure 14: Simulating Test Block

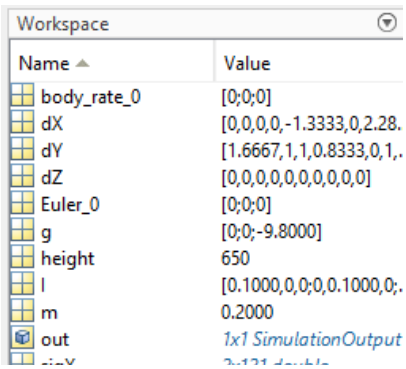

Figure 15: Updated Test Parameter

Although MSOSA offers some simulation capabilities, its range of simulation outputs is limited. To overcome this, MATLAB and Simulink were employed to provide a more advanced and dynamic simulation output. Rather than attempting to construct a simulation output for the flight path of the UAV within MSOSA, an activity diagram was created to develop a shared workspace with MATLAB and Simulink. Cameo Simulation Toolkit was utilized to call MATLAB/Simulink functions directly from Magic Systems of System Architect.

MATLAB is one of the supported evaluation tools, meaning that SysML model parameters can be input and run through MATLAB/Simulink models, resulting in outputs that can be integrated back into the SysML models. After the parameters have been imported using a block definition diagram, the necessary MATLAB and Simulink files can be loaded via the activity diagram, which provides a simulation output of the UAV's flight path. To achieve a complete simulation, five files must be executed or loaded, as shown in Figure 16. This integration of MSOSA, MATLAB, and Simulink enables the creation of more sophisticated simulation outputs, facilitating the assessment and improvement of the UAV's flight route.

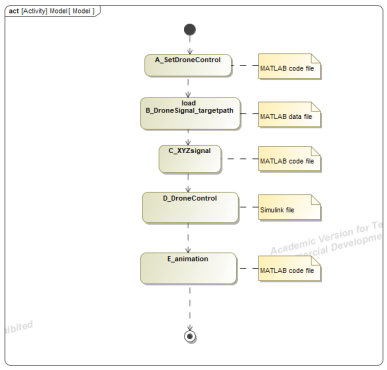

Figure 16: MATLAB/Simulink Simulation Activity Diagram

The initialization of the UAV's parameters is achieved through the use of *A_SetDroneControl*. The target path for the drone is determined by *B_DroneSignal*, which establishes the X, Y, and Z coordinates, as well as the T (time) required to reach each set of coordinates using a matrix. After this is complete, the *C_XYZsignal* MATLAB script is executed to calculate the UAV's velocity and Euler's angles at each of the specified points.

Once these calculations are complete, the output values are input into the Simulink File *D_DroneControl*, and subsequently sent into *E_animation*. The chronological sequence of the files' execution is depicted in the activity diagram, with each step clearly illustrated in Figure 17.

After all the stages are completed, Figure 18 shows the successful conclusion of the simulation. By following this process, the simulation can accurately determine the UAV's flight path and ensure that it follows the specified target path.

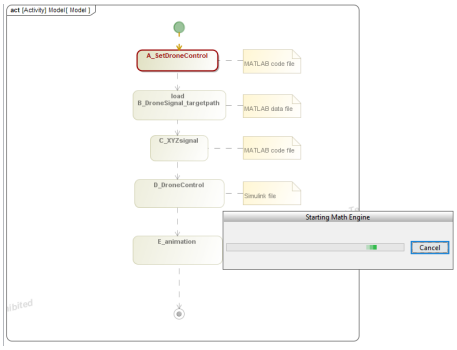

Figure 17: Simulating Activity Diagram and Establishing Shared Workspace

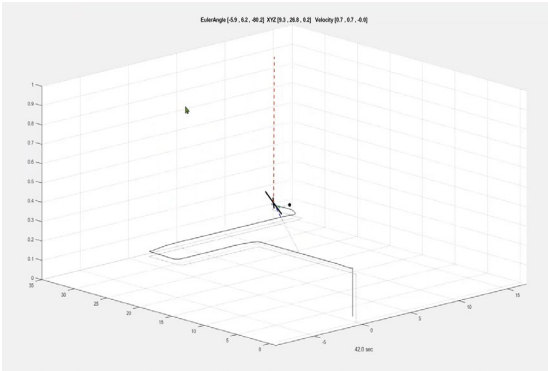

Figure 18: UAV Flight Path Simulation

MSOSA, MATLAB, and Simulink work together to provide a unified workspace that enhances the capabilities of simulations. When simulating the flight route of a UAV, the number of required files and input parameters can lead to errors. However, MSOSA ensures that the necessary input values are not only visualized but also maintained in MATLAB and Simulink, even when user or system requirements change. This functionality not only helps to prevent errors but also makes it easier for other users to replicate the simulation process.

Thanks to the shared workspace created by MSOSA, even a user with no prior knowledge of MATLAB can execute the corresponding files. This greatly enhances the accessibility of the simulation process and allows for more collaboration between different stakeholders.

The overall objective of Model-Based Systems Engineering (MBSE) is to provide a framework that makes complex system models more understandable and manageable throughout the development process. By using MSOSA, users and stakeholders can be confident that the architecture of the system satisfies their requirements. The models and shared workspace created by MSOSA allow for a more comprehensive representation of the system, ensuring that all users can understand its behavior and interactions.

E. Future Work

Real-time data can be obtained from a variety of operational factors, such as velocity, battery mAh, picture resolution, and GPS coordinates. These variables can be simulated using game engines such as Unity or Unreal Engine.

State machine diagrams can be created and exported as code, which can then be utilized to set game engine parameters. While both MSOS and MATLAB offer simulation features, they have limits in terms of visualization. However, using algorithms and scripting languages in a game engine like Unity 3D can significantly enhance simulations.

Unity 3D is a well-known game engine that enables the creation of 2D and 3D visual effects, making it an ideal tool for dynamic simulations. By integrating MSOS and MATLAB with Unity, it is possible to create a cross-platform shared workspace where any scenario or system can be recreated.

In order to create a dynamic simulation, reliable and confirmed data as well as time are required for modeling. This involves discovering the necessary computations, equations, and data for the simulation. While this process may be time-consuming, the use of a game engine like Unity can help to create more engaging and dynamic simulations.

V. Conclusion

MBSE tools, languages, and methodologies offer a framework for organizing system data, which can be used for quantitative analysis and simulation. The development of models using SysML is motivated by the belief that it provides a comprehensive architecture that illustrates the behavior characteristics of an operationally realistic situation. This methodology enables the creation of system blocks with specific properties and behaviors that can be modified continuously, thus avoiding wastage of resources.

To simulate scenarios with greater dynamic complexity, the parameters for the surveillance scenario are classified into their corresponding diagrams. The next stages of the study involve the development of more detailed models, which will be simulated using a gaming engine like Unity 3D or Unreal Engine. The aim is to extract data from generated SysML models and export it to MATLAB and the appropriate game engine, and vice versa, to develop a digital twin.

Testing the established framework is a crucial objective to validate the procedures necessary for creating an MBSE-enabled digital twin. Displaying the created surveillance scenario in greater detail using a game engine will enable stakeholders to gain a better understanding of the ideal flying settings, weather conditions, and scenario test runs. Game engines employ algorithms that more accurately imitate the behavior of real-world objects such as unmanned aerial vehicles (UAVs) or combat vehicles. These methods are essential for verifying the SysML-developed parameters and requirements. This experimentation eliminates the need for physical testing, saving both money and physical resources.

Acknowledgment

The authors would like to acknowledge the funding and support provided by the CREST Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (MECIS) under NSF Award No. 2112650.

References

[1] Friedenthal, Sanford, Regina Griego, and Mark Sampson. "INCOSE model-based systems engineering (MBSE) initiative." In INCOSE 2007 symposium, vol. 11. 2007.

[2] MagicDraw - CATIA - Dassault Systèmes®. “MagicDraw - CATIA - Dassault Systèmes®.” www.3ds.com. Accessed May 13, 2022. https://www.3ds.com/products-services/catia/products/no-magic/magicdraw/.

[3] Delligatti, Lenny. SysML distilled: A brief guide to the systems modeling language. Addison-Wesley, 2013.

[4] Akundi, Aditya, Viviana Lopez. " A Conceptual Model-based Systems Engineering (MBSE) approach to develop Digital Twins." In 2022 IEEE International Systems Conference (SysCon), pp. 1-5. IEEE, 2022.

[5] Congressional Budget Office. “Projected Acquisition Costs for the Army’s Ground Combat Vehicles | Congressional Budget Office.” www.cbo.gov, April 1, 2021. https://www.cbo.gov/publication/57085.

[6] US Marine Corps. “Marine Corps Tank Employment”. marines.mil. Accessed May 13, 2022. https://www.marines.mil/Portals/59/Publications/MCWP%203-12%20Marine%20Corps%20Tank%20Employment.pdf.

[7] Mhenni, Faïda, Jean-Yves Choley, Olivia Penas, Régis Plateaux, and Moncef Hammadi. "A SysML-based methodology for mechatronic systems architectural design." Advanced Engineering Informatics 28, no. 3 (2014): 218-231.

[8] Friedenthal, Sanford, Alan Moore, and Rick Steiner. A practical guide to SysML: the systems modeling language. Morgan Kaufmann, 2014.

[9] Ahmed, Shibbir, Baijing Qiu, Chun-Wei Kong, Huang Xin, Fiaz Ahmad, and Jinlong Lin. 2022. "A Data-Driven Dynamic Obstacle Avoidance Method for Liquid-Carrying Plant Protection UAVs" *Agronomy* 12, no. 4: 873. https://doi.org/10.3390/agronomy12040873

[10] MathWorks. (n.d.). Simulink - simulation and model-based design. Simulation and Model-Based Design - MATLAB &. Retrieved September 14, 2022, from https://www.mathworks.com/products/simulink.html

[11] Pavalkis , Saulius. “Aircraft Radar Display SysML MagicGrid Sample with Simulation and Analysis.” YouTube, 2021. Retrieved April 2, 2022, from https://www.youtube.com/watch?v=JtWZQM-yamk&t=559s.