Model Based Test Engineering: Incorporating Test Engineering into the Digital Engineering Transformation

Johnston A. Coil[1]
LinQuest Corporation
Colorado Springs, CO, 80921, USA
johnston.coil@linquest.com

Maj Steven T. Wachtel
United States Air Force
Peterson SFB, CO, 80914, USA
steven.wachtel.1@us.af.mil

Robert L. Riley, Jr.
Air Force Research Laboratory
Edwards AFB, CA, 93524, USA
robert.riley.12@spaceforce.mil

As more programs adopt digital engineering transformation, test engineers must take advantage of the opportunities this change presents. Utilizing Model Based Systems Engineering (MBSE) principles and applying them to test engineering enables the test program to access authoritative structural, behavioral, and requirements data on the System Under Test (SUT) and take advantage of the MBSE principles of simplifying complexity, relationships between model elements, and model element re-use. This paper presents a Model Based Test Engineering (MBTE) metamodel that federates with SUT models; provides improved linkages between requirements, test procedures, safety, and test resources; improves test program insights through automated reporting; and is applicable for testing of physical systems, hardware in the loop, digital twins, or hybrid testing. Application and lessons learned are provided from case studies of reverse-engineering document-based test plans and from the execution of a MBTE pilot project at the 780th Test Squadron, Eglin AFB, FL.

Any views, opinions, findings, conclusions, or recommendations expressed in this document are the authors' and do not necessarily reflect the views of Department of the Air Force, the Department of Defense, or the U.S. government.

1. Introduction

1.1. Problem Statement

Digitally-born weapon systems are on their way. Beginning as early as the Request for Proposal (RFP) phase, acquisition programs are pivoting towards Digital Engineering (DE) and Digital Material Management (DMM) to leverage the benefits of a coherent digital fabric. This shift from

[1] SFTE Member

document-based processes to digital, model-based processes stands to change the way test, particularly Development Test (DT), is approached. Members of the DT community must understand what this shift means and how they will fit within this new paradigm, where DT may begin before any hardware is ever built.

Some weapon programs are building DE into their RFPs now, setting initial DE requirements in the Statement of Objectives (SOO). These requirements include establishing a DE Ecosystem for the program; completing preliminary design, evaluation, and Systems Engineering Technical Reviews (SETRs) in the DE ecosystem; granting the government access to the ecosystem; and participating in a consortium with other prime contractors to develop a Government Reference Architecture (GRA) to be used by other weapon programs.

As the Executing Test Organization (ETO) for some of these programs, the 780th Test Squadron at Eglin AFB has begun to explore ways to participate in the digital ecosystem. A key effort in this exploration has been a Pilot Project, applying Model-Based Systems Engineering (MBSE) tools and principles to the Test Engineering domain. This paper presents the approach and lessons learned from this "Model-Based Test Engineering" (MBTE) Pilot Project.

1.2. What is the Digital Transformation?

Former Assistant Secretary of the Air Force for Acquisition, Technology and Logistics, Dr. Will Roper, describes the Digital Transformation, consisting of digital engineering and management, agile software, and open architecture, as "the next big paradigm shift for military tech dominance." [1] Ansys provides a simple example of Digital Transformation benefits by contrasting it with more traditional processes: "If an engineer changed a single dimension on one document, they had to go through the entire stack of documents and make sure that same change was made in every other document that contained that dimension. If other copies existed, you could not be sure that the change was recorded in all copies." [2] The Digital Transformation aims to solve problems like this by using software to automatically propagate the changes throughout the system's entire lifecycle management ecosystem.

Dr. Roper describes another key concept under the Digital Transformation umbrella: the Digital Thread. "Digital Threads are extensible analytic frameworks to ***connect*** models — and all associated data, software, and functional support that govern system lifecycle phases — to create an authoritative, digital source of truth with one-to-one real-world traceability." [1] If you "pull" on the Digital Thread (for example, by upgrading the capacity of the battery in your small UAS) everything connected to that thread feels the tug (you may now have more range, but also more mass, more heat, and higher cost). Again, contrasting this with a traditional process, some of these impacts could be easily overlooked and left undiscovered until testing beings.

1.3. Digital Twins and Their Impact to Testers

The original concept associated with what would later be termed a "digital twin" was first proposed in 2002 by Dr. Michael Grieves, an Engineering Professor at the University of Michigan. He used the term "mirrored spaces model" but through collaboration with Mr. John Vickers, the term

"digital twin" was coined and used in association with the concept from then on. Mr. John Vickers, a Senior Engineer for the National Aeronautical and Space Administration (NASA) is credited with coining the term "digital twin." [3]

The digital twin is intended to serve as the "information construct" of the physical system it was designed to represent. [4] In the highest state of actualization for a digital twin, it should provide equivalent or greater levels of information than could otherwise be acquired through physical connection or possession of the actual system, or physical twin. This should establish a basis for addressing "what-if scenarios" with data-driven solutions derived from an ability to now predict expected behavior of the physical twin. [3] Figure 1 shows the connections between physical and virtual systems for developing a digital twin. A digital twin can have several layers of fidelity depending on the simulation application (e.g., physics, mission, or campaign level).

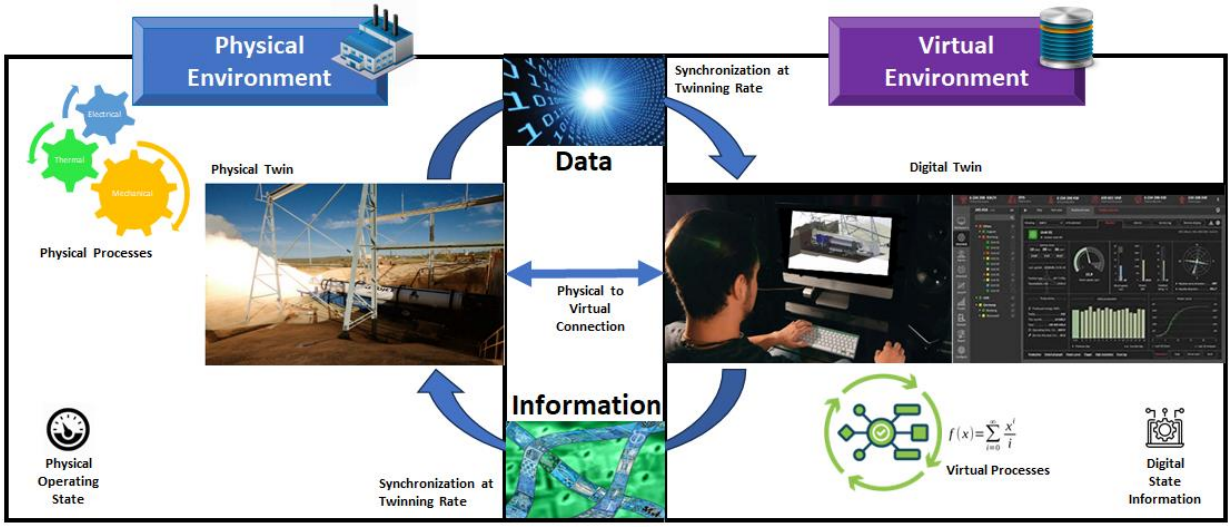

Figure 1. Physical Systems and Virtual Systems Connected for the Development of a Digital Twin (Adapted from [3])

For testers, the advent of digital twins will affect test programs in different ways depending on the state of the system or program being supported. Data collection specifically meant for validating and verifying the digital twin template model(s) will have to occur during a system's development. Testers will also have to adapt to conducting testing on digital twins, although this will likely occur during the operations and sustainment phase of a system lifecycle.

2. Background

2.1. What is MBSE?

Model Based Systems Engineering is a cornerstone of the digital engineering transformation. At its most basic level, MBSE aims to improve the traditional approach to systems engineering, where multiple documents covering various aspects of a program are updated independently with a

descriptive system model in the form of a relational database. Figure 2 illustrates the differences between a document-based systems engineering approach and a model-based approach.

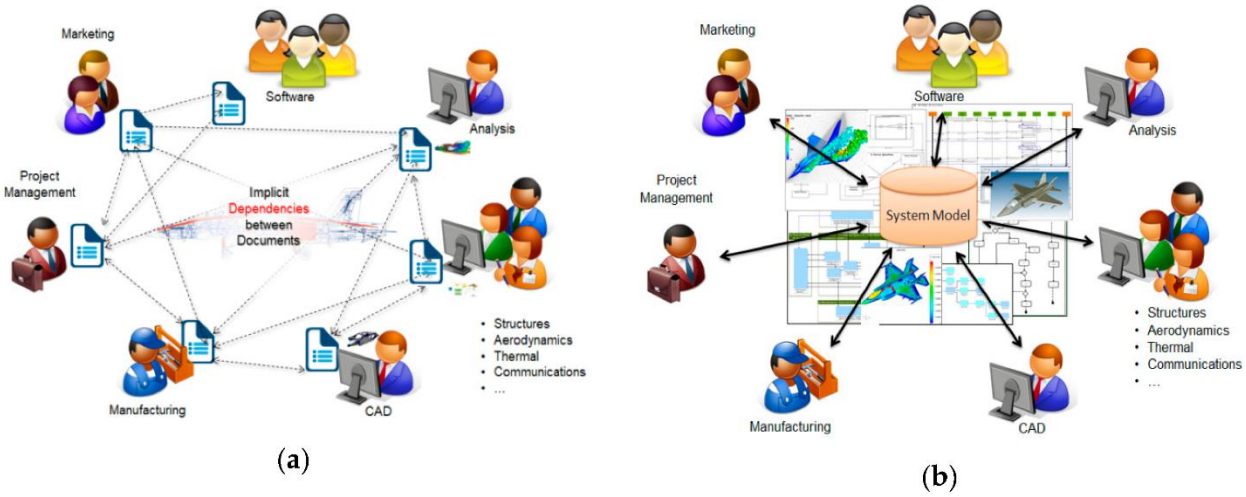

Figure 2. Traditional Systems Engineering (a) and Model-Based Systems Engineering (b) [5]

A key advantage of MBSE is that a complex problem, such as describing all aspects of a system and its design, can be represented in a single model with simplified views into that model showing a particular subset or discipline of the whole while remaining integrated into a single overall system description model. Model elements may be reused once created, reducing overall workload. Also, MBSE enables improved communication across the entire system design or project team, as changes propagate across the model automatically.

2.2. Systems Modeling Language (SysML)

The Systems Modeling Language (SysML) is a formal language that is tailor made for MBSE. It is a graphical language whose standards are managed by the Object Management Group (OMG). As of the writing of this paper, the current version is 1.6, which is based on and extended from the Unified Modeling Language (UML), which was developed and popularized in the software development field. OMG is working on SysML 2.0, which will expand the functionality of the language, but switch the basis of the language from UML to the Kernel Modeling Language (KerML). This transition will not affect this particular discussion of using SysML models to aid in test engineering or the overall interaction of a test model with the system model, although the semantics of both models will need to be in the same SysML basis (i.e., 1.x or 2.x).

A SysML model describes the structure, behavior, and requirements of a system, system of systems, enterprise, or (as described here) test program and the relationships between those model elements. Users view the model using diagrams, which represent particular aspects of the model. The diagram types of SysML are shown in Figure 3, along with notation on whether it is the same as, modified from, or new compared to UML 2.

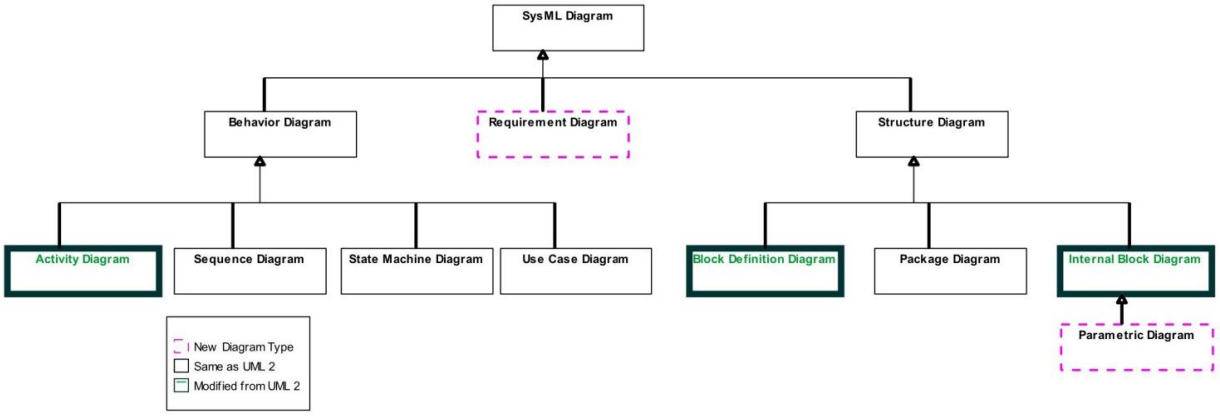

Figure 3. SysML Diagram Taxonomy [6]

2.3. Current State of "Model Based Testing"

As a language, SysML comes with some test-related semantics built into it. Figure 4 illustrates most of these semantics in a diagram. The first feature is the «testCase» stereotype, which can be applied to any kind of behavior model element. This emphasizes that a test is something that a tester *does* and is not a thing a tester *has* or *is*. The specification of the «testCase» stereotype states that when the stereotype is applied to an element, there is a "verdict" output of the behavior of type VerdictKind. VerdictKind is simply an enumeration of possible test results: "Pass", "Fail", "Inconclusive", "Error" and "None". Finally, the «verify» dependency links requirement model elements with elements that determine whether the system meets the requirements (i.e., a test).

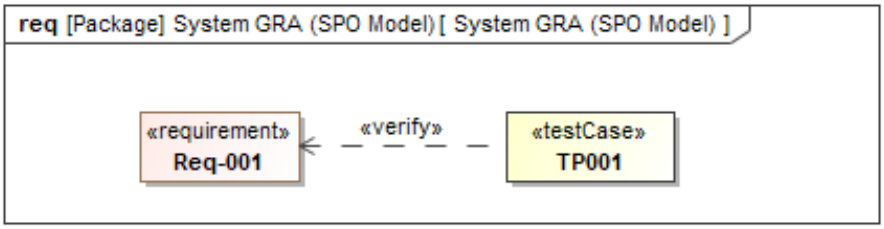

Figure 4. Basic SysML Test Related Semantics

Implementing these aspects of the SysML language has not to date been a priority within the MBSE community, as MBSE implementation has been largely within the design and program management community. The most extensive implementation found when researching the topic was using internal simulation within the MBSE software that showed that if a system were built in accordance with the modeled design, then it should meet the requirements. Figure 5 shows a screenshot from an online video explaining this usage. While certainly a worthy practice during the design process, few in the test community would consider results from such a simulation sufficient for concluding whether a system meets requirements or not.

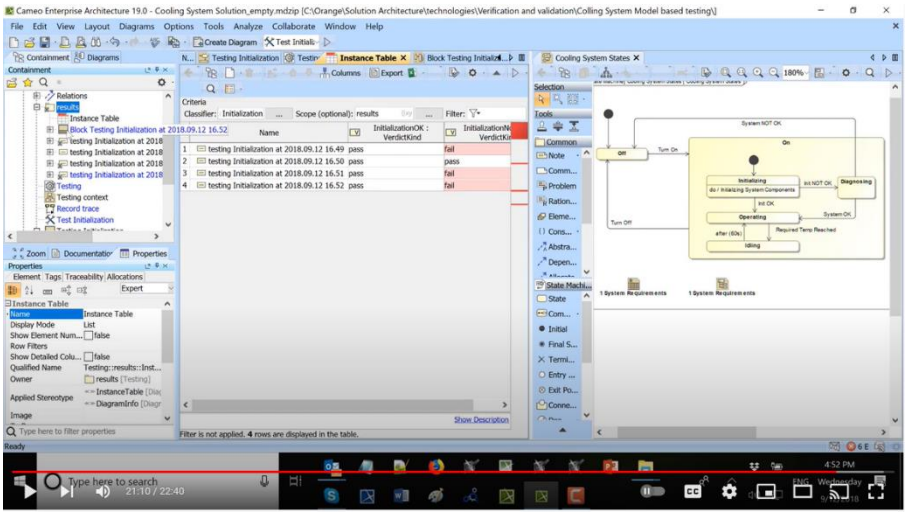

Figure 5. Screenshot from "Model-Based Testing Using SysML" YouTube Video [7]

The other application discussed in literature regarding testing and MBSE involves potentially linking MBSE with Modeling, Simulation and Analysis (MS&A) tools. This has relevance in both evaluating a particular architecture prior to hardware development at the beginning of a product lifecycle and when using digital twins for testing once those twins are verified and validated. There was a distinct gap in how to integrate data from testing conducted on physical hardware in a relevant operational environment into the digital engineering thread. This gap is what motivated the development of the Model Based Test Engineering approach.

3. Model Based Test Engineering (MBTE) Approach

The MBTE solution is an independent SysML model that is federated to the model of the System Under Test (SUT). The test model has access to the "Authoritative Source of Truth" (ASOT) for the structure, behavior, and requirements of the SUT, and feeds back test results, values determined experimentally through testing, requirement verification status, and test conclusions and recommendations. The approach is designed to be flexible in order to allow for full-system physical testing in real environments, hardware in the loop testing in the lab or in a live-virtual-constructive (LVC) environment, and fully digital testing utilizing digital twins. When completed, the MBTE model is the data repository for test planning; aids in organization, resource allocation, data acquisition, and risk management during execution; and automates and streamlines the reporting process.

The separate test model allows testers to maintain independence from the program management office or organization, a common policy requirement for military test organizations, and enables the development and re-use of libraries for test procedures as well as test-specific resources such as instrumentation, range, and test support resources such as control rooms and chase aircraft.

3.1. Why use SysML for MBTE?

When looking to bring test into the digital engineering fold, a reasonable question is "What is the best modeling technique to support test engineering?" The MBTE approach uses SysML to model the test process for three primary reasons:

- **Native access to System Under Test model:** Any time data transitions from one platform to another within a digital engineering ecosystem, it is necessary to transform or translate the data into the format for the target platform. Since SysML is the standard way for modeling a system architecture, modeling the test data in SysML eliminates this requirement and ensures the test model can utilize all features of the structure, behavior, and requirements models without risk of data loss or corruption.
- **Harness advantages of MBSE**: As discussed in the "What is MBSE?" section above, using MBSE methodology for test engineering improves testers' ability to address large, complex problems and test programs. It improves the interconnectedness of test design, linking test resources, risk mitigations, test conditions and limits, data requirements, and detailed test procedures. Changes propagate throughout the model, eliminating the need to cut-and-paste or search-and-replace multiple places throughout a test plan. It also allows test organizations to re-use test specific model elements, reducing errors and test planning time.
- **Test Engineering is a sub-discipline of Systems Engineering:** This applies in two contexts. Program system engineering should take testing into account in the overall system engineering plan. Also, planning, managing, and executing a test program is similar in many respects to traditional systems engineering, bringing together multiple engineering disciplines, competing requirements and resource constraints, and having to devise solutions that meet technical (data) requirements. Since SysML accommodates systems engineering by design, it is fully capable of meeting all the needs of test engineering.

3.2. Basic Model Structure and Federation

As stated earlier, the most fundamental construct of the MBTE approach is an independent model that is federated to the MBSE model of the SUT. In this context, a federated relationship between models indicates an ability to see changes to model elements from the linked model, but those changes are not automatically propagated into the hosting model without user intervention. Depending on the MBSE tool, this typically consists of notifications of modified elements in the linked model and what elements of the host model have relationships to those elements, allowing the architect of the host model to review the changes before accepting them.

The test model reads in the structure, behavior and requirements of the SUT model. The structural elements provide the ASOT of the test item description. The behavioral elements are used in test card development, when a test point is designed to trigger a specific system response. The requirements model provides the verification linkage from the test points. From an output perspective, the test model provides test results which can influence system architecture values,

verification status of requirements, and tester conclusions and recommendations. Figure 6 illustrates a simplified view of the data exchanges between the models.

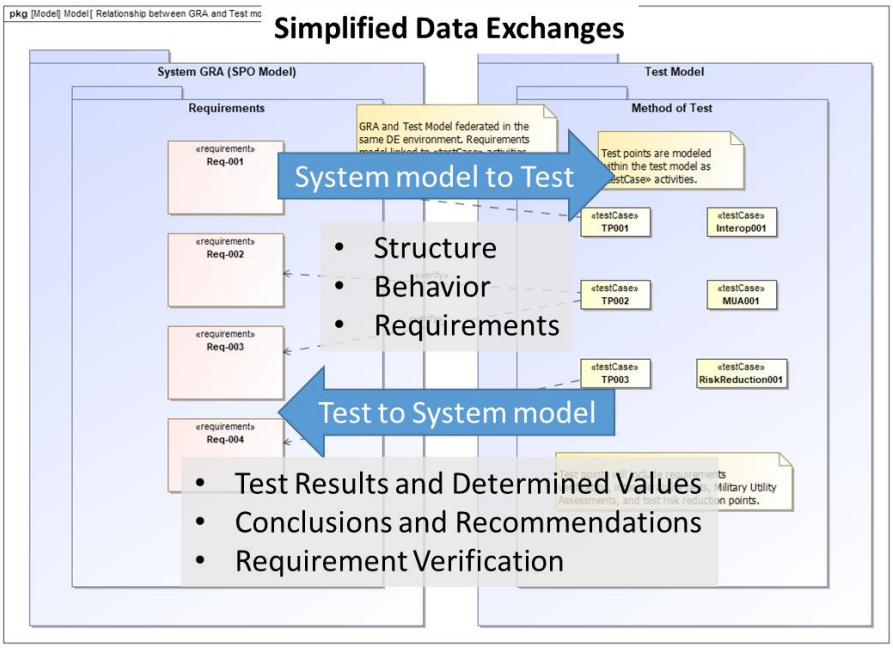

Figure 6. Simplified Data Exchanges Between System and Test Models

3.3. MBTE Metamodel Overview

3.3.1. Test Cases

The central feature of a MBTE model is the Test Case, which represents a test point. The basic model for this is an activity with actions representing the Planning, Execution, Analysis, and Reporting stages of testing. Figure 7 shows the basic structure for a test case activity and the data exchanges that occur over the execution of a test point.

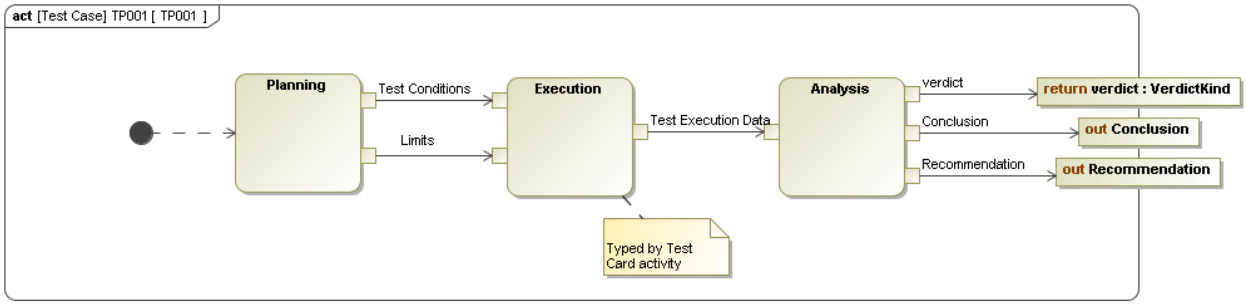

Figure 7. MBTE Test Case Basic Model Structure

The planning action represents the creation of the MBTE model elements for the test point. The MBTE model contains elements that specify the test conditions and limits for the execution of the

test point. The test conditions and limits can have multiple values for a single test procedure if the same test procedure is repeated under different conditions for safe test conduct (e.g., envelope expansion), execution of a design of experiments matrix, or other reasons to repeat a test point under varying conditions. The execution action is typed by the behavior that describes the detailed test procedure or test card. It accepts the test conditions and limits as inputs, and outputs the data collected during test execution. The analysis action processes the data collected during execution and outputs the verdict, conclusions, recommendations, and any other test output data required from the test point such as actual system performance values.

Each of these testing stages are modeled separately for each test point to provide maximum flexibility for physical or digital execution of each step that is most appropriate for the test program. The planning stage is always digital by virtue of using the MBTE approach. The execution action can automatically generate test cards for execution of tests on a physical system or can automatically drive the execution of a digital twin test in an MS&A environment or a hybrid physical/digital hardware-in-the-loop test. If it is possible to automate the analysis step, the model can drive the execution of the analysis given the test data and automatically produce determined values and verdicts; if the data requires the interpretation of a test engineer, the analysis action can represent that analysis the takes place outside of the model. Ultimately, the reporting outputs should be entered into the model to take full advantage of the MBTE approach.

The actual execution of a test point becomes an instance of the test case activity and records the test conditions, limits, and outputs in that instance. The Cameo/MagicDraw family of MBSE tools from 3DS, which have been used in the MBTE development work, does not currently allow storage of slot data in instances of behaviors. As a workaround, test point structural elements (in SysML, these are blocks) represent the test points and have the test case activities as the classifying behavior. The blocks have value properties that mirror the test case activity outputs and store the values in the block instances. Figure 8 shows an example of a test point block with additional properties to more fully describe the test point, such as success criteria, evaluation criteria, expected results, and data band and tolerance.

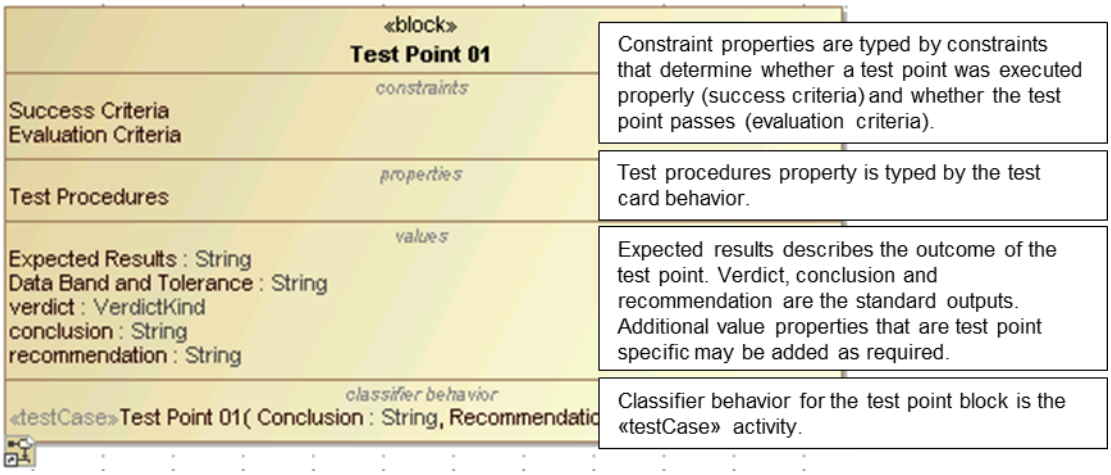

Figure 8. MBTE Test Point Block Structure Example

3.3.2. Test Resources

A second key feature of a MBTE model is the modeling of test resources. This includes instrumentation, range, personnel, and other test support resources. When creating the MBTE methodology and metamodel, the development team created several models by reverse-engineering pre-existing document-based test plans. In each of those cases, the MBTE model revealed shortcomings in the resource allocation planning of the test programs that were missed in the document-based approach.

Resources utilized in a test program are modeled as part properties of blocks representing required resources and desired resources. Optionally, this can be refined between resources and personnel. These part properties are elements of usage, representing instances of described resources. Ideally, a large, shared test enterprise will have a Test Resource Library containing elements of definition with all the specifics of the test resources available for use by multiple test teams. Figure 9 illustrates a simplified example of a resource diagram in a MBTE model.

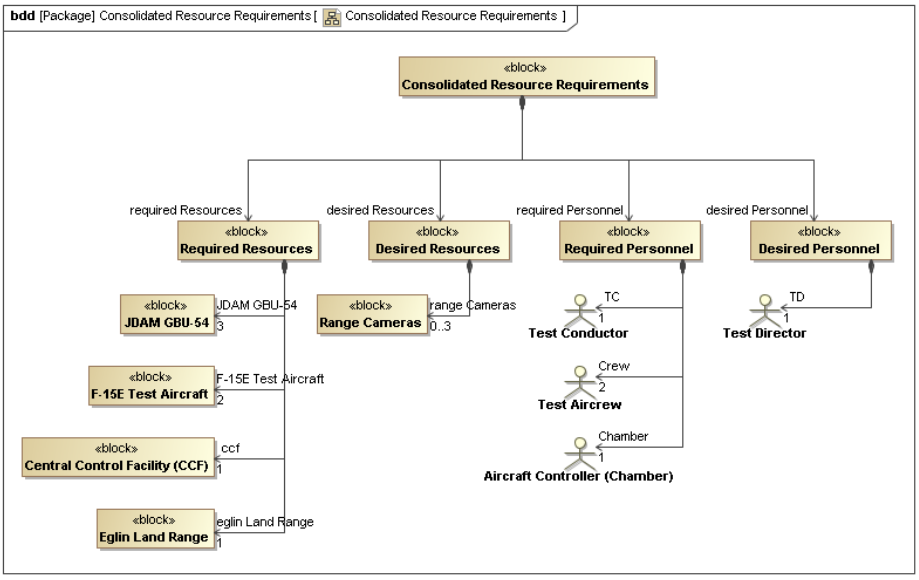

Figure 9. Sample MBTE Resource Diagram

Resource specific roles for various functions within a test design also use the resource elements to indicate how those functions are fulfilled. To avoid duplication of resource modeling, shared aggregation relationships are used to show the linkage between the resources and the test functional elements, as shown in Figure 10.

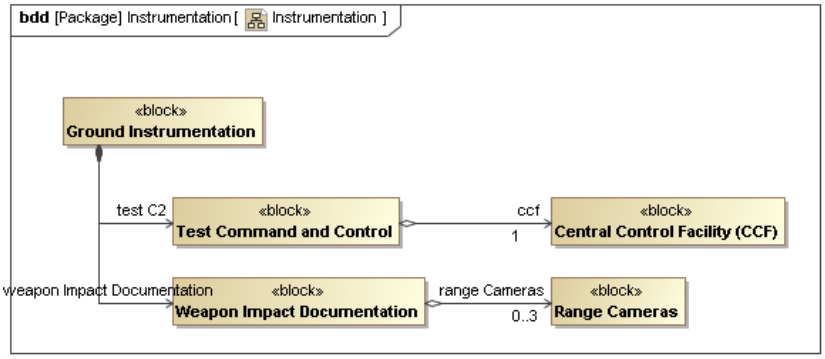

Figure 10. MBTE Resource Usage for Test Design Elements

This results in two sets of elements of usage: part properties of the resource model element, and reference properties of the test design role elements. Binding connectors in an internal block diagram indicate that the two instantiations are the same physical instance or item, as shown in Figure 11. This modeling approach consolidates all resources required for a complete design of a test program and allows for a straightforward allocation matrix of participating organizations and all the test resources to ensure no resources are overlooked or unassigned. An example of such an allocation follows in the Pilot Project section.

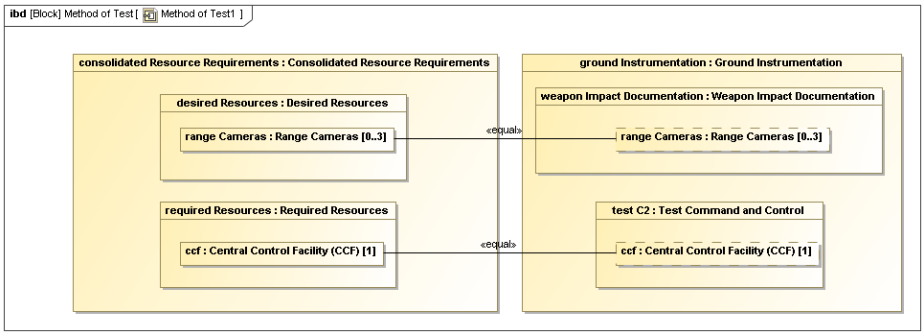

Figure 11. Linking Test Resources with Role Based Modeling

Building off these two foundational ideas, all the data of a document-based test plan and the execution of a test program become contained within the MBTE model. Through relationships between model elements, the model can, for example, show specific data collection, allocate test objectives to ground or flight test events, or organize a test card deck for a sortie. These are best illustrated with concrete examples, and therefore are further discussed within the context of the pilot project.

3.4. Advantages to Testers

Using the MBTE approach provides several advantages over the document-based approach for testers. These advantages include:

- **Access to System Under Test ASOT**: The test model being directly linked to the system model ensures the test team has access to and is notified of any changes to the structure, behavior, or requirements models that could affect the overall test plan and that the test item description matches exactly with the latest system modifications.
- **Automated Reporting**: The idealized version of testing has a system delivered to testers and once the test program is complete, the testers return a comprehensive technical report with all relevant data, analysis, conclusions, and recommendations for the system. The reality is that programs cannot tolerate having so little insight about the system and its progress through testing. This has resulted in programs requiring various additional reports: preliminary reports, quick look reports, post-flight reports, et cetera. Using MBTE, once analysis of a test point is complete, the results can be entered into the model and delivered automatically to the program office as the test program continues execution. Model configuration management tools, which are built into the MBSE platforms, and organizational procedures ensure that conclusions and recommendations have appropriate levels of review before delivery.
- **Increased technical rigor**: The relational nature of MBSE models gives natural traceability between requirements, test points, test procedures, limits and conditions, data parameters, and test resources. This simplifies technical review and reduces the risk of test point design iteration resulting in not collecting the correct quantity or quality of data.
- **Harness MBSE Principle of Re-use**: A separate Test Model not only ensures independence for test, but it allows the test community to develop test-specific model elements and create libraries of elements to be re-used across the test enterprise. Resources such as range assets, instrumentation, control rooms, and test support assets only need to be modeled once and may be used on multiple test programs. There is also the potential to re-use test procedures for similar tests (e.g., integrating different weapons onto a fighter aircraft or installing a single system on several different aircraft platforms).

3.5. Advantages to Programs

In addition to aiding the testers, MBTE will provide advantages to programs as well. Program advantages include:

- **Increased Insight throughout Test Program Execution**: This is essentially the other side of the automated reporting coin listed above. Because all data will be in linked models, the program will receive timely updates on test execution status, requirements verification, and overall system performance.
- **Linkage between System Model and Test Data**: Because the data are linked directly, a program can directly compare observed performance reported from the test team with system design parameters and goals. Parametric modeling could automatically update system values with those that are measured through testing, aligning the entire architectural model with observed performance.
- **Integrate Entire Test and Verification Strategy**: Complex programs, especially those managed within government program offices, often have layered approaches to verifying

requirements. This can include component level tests, system level tests, hardware in the loop tests, ground tests and flight tests. They are conducted by multiple organizations including sub-contractors, prime contractors, and government test organizations. MBTE can bring all these various test data sources together in a cohesive way and easily show interdependencies among the test points with supporting data results and prerequisite links between test points.

4. Pilot Project

4.1. Test Program Overview/Reason for selection

As the 780th Test Squadron prepares to serve as the Executing Test Organization (ETO) for several digitally-born weapons, it became clear that MBSE training alone would not be sufficient to provide personnel with the knowledge and expertise to plan a test in a digital environment. Some practical experience and a lower-risk experiment was the logical next step.

The Advanced 2K-lb penetrator, Universal Low-profile Arming Generator (A2K ULAG) test program was chosen as the first opportunity to employ the MBTE approach in practice. While the full details of the test cannot be shared publicly, this program was an ideal candidate for several reasons: it was a relatively small program, involving one sortie with six weapon releases; many of the details of the test were unclassified, allowing easier access to the modeling environment and software; the program office had a model of the weapon itself, albeit relatively simplistic; and the execution timeline aligned well with the development of the Test Metamodel.

4.2. Pilot Project Objectives

The 780 TS identified five objectives for the MBTE Pilot Project:

1) **Exercise skills obtained through SysML training.** With many program offices moving towards digitally-born weapons, the squadron identified the need to build some digital competency within the workforce. Two tracks were created: “basic,” for the majority of test professionals supporting digital programs, and “expert,” for a small subset of the squadron expected to actually perform the modeling. The basic track consisted of a series of videos from the Air Force Digital Transformation Office, a self -paced Digital Acquisition University (DAU) course, and a beginner-level SysML/MBSE Course. The expert track included an additional 40-hr SysML course. The pilot project was the first opportunity to exercise the skills from these training programs in practice.
2) **Gain experience interacting with a Government Reference Architecture (GRA).** Many of the program office the 780th supports have begun using a GRA to guide and constrain the instantiations of weapon models. Very simply, the GRA can be thought of as the “template” from which the weapon model starts. For the pilot project, the weapon system in question already had a rudimentary model constructed in accordance with the GRA. This allowed the team to connect the test model to the weapon system’s ASOT and learn how to interact with the GRA.

3) **Learn to work within an Integrated Digital Environment (IDE).** An Integrated Digital Environment consists of a collection of data, models, and tools for collaboration, analysis, and visual representation of work activities across all functional domains. [8] For the context of the pilot project, an IDE was critical to enable collaboration between the test team members and modeling experts, and for connecting the test model to the weapon system model to maintain the Digital Thread. For the pilot project, the 780th used the Air Force Research Lab's (AFRL) RogueONE environment. RogueONE was established to serve as an impact level 5 (IL-5) compliant, innovation hub for the development of weapon-centric software applications using principles associated with agile development, Development Security Operations (DevSecOps), and open system architecture. It has been utilized by AFRL scientists & engineers, AFLCMC engineers, and AFTC test engineers as demonstrated with this pilot project. This IDE provided access to both Cameo, the MBSE environment, and Teamwork Cloud, a model repository that handles versioning and configuration management. Working in a collaborative environment is critical for maximizing the value of MBSE, but it brings added complexity that warranted deliberate focus for the pilot project.
4) **Provide feedback on the GRA and the Test Metamodel.** The pilot project team were some of the first external users of the GRA, and the very first users of the Test Metamodel in practice. Both models are intended to improve iteratively. A close working relationship between the 780th and the owners of both models provided a channel for feedback for each. The nature of the test and safety planning process also exposed the models to several new audiences, allowing even more feedback opportunities from various stakeholders.
5) **Capture lessons learned for testing future digital programs.** It was fully expected that the pilot project would uncover many roadblocks, so much so that the discovery of "unknown unknowns" was identified as a distinct objective of the effort. These lessons are expected to dramatically improve the ability of the 780th to support future digital-born programs.

4.3. Highlights of pilot project modeling effort

4.3.1. Test Objectives and Test Cards

4.3.1.1. Test Objectives

This test program was selected for the pilot project because of its overall simplicity: one ground test point and one flight test point. The 96 TW methodology of test planning defines an Overall Test Objective for the entire program, Specific Test Objectives for refining that main idea, and then Measures of Performance for specific test points. The initial modeling effort looked to directly mirror the document-based project. Figure 12 shows this initial modeling structure.

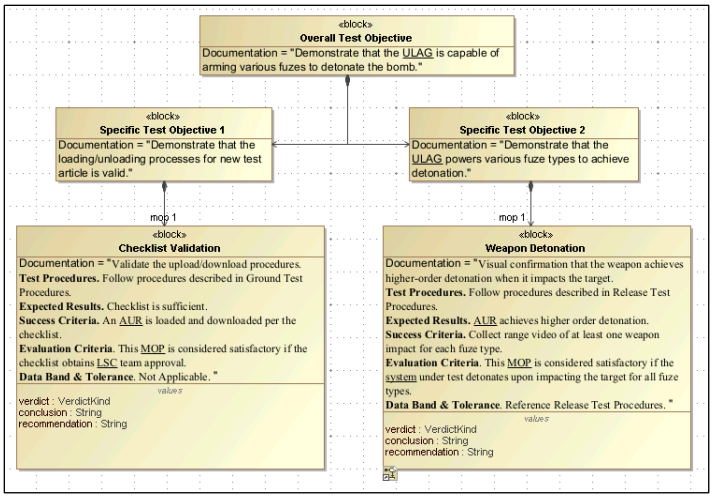

Figure 12. Pilot Project Test Objectives

While the approach of placing the text based descriptive data in the documentation field worked for reproducing a standard test plan document, it failed to make any kind of data-rich modeling impact. For this reason, the modeling approach was modified to align with the metamodel described in the Test Resources metamodel description. Figure 13 shows the updated version.

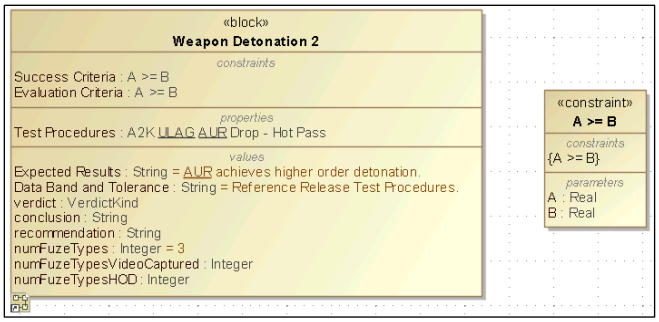

Figure 13. Data Driven Measure of Performance

4.3.1.2. Test Cards

The test case model for this pilot project is shown in Figure 14.

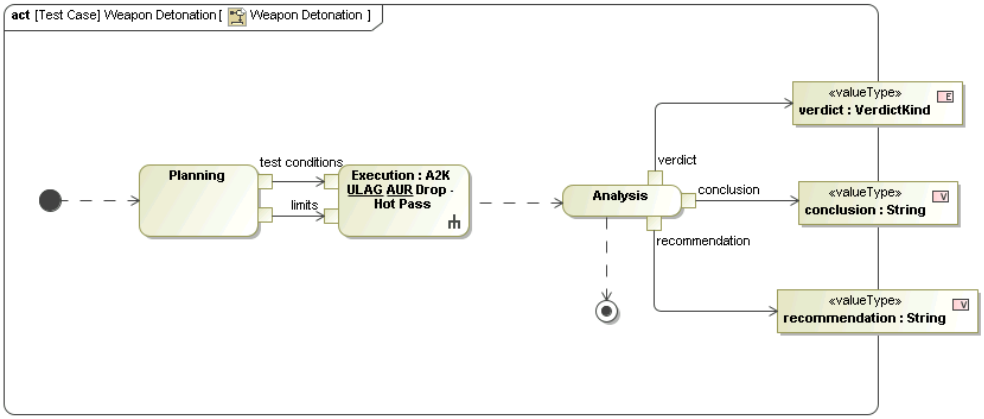

Figure 14. A2K ULAG Test Case Model

The pilot project flight test has two test cards: a cold pass and a hot pass. As the test case shows, the hot pass is the card that provides the data to meet the Weapon Detonation measure of performance under Specific Test Objective 2. The test card activity models the specific steps to accomplish the card and is presented in Figure 15.

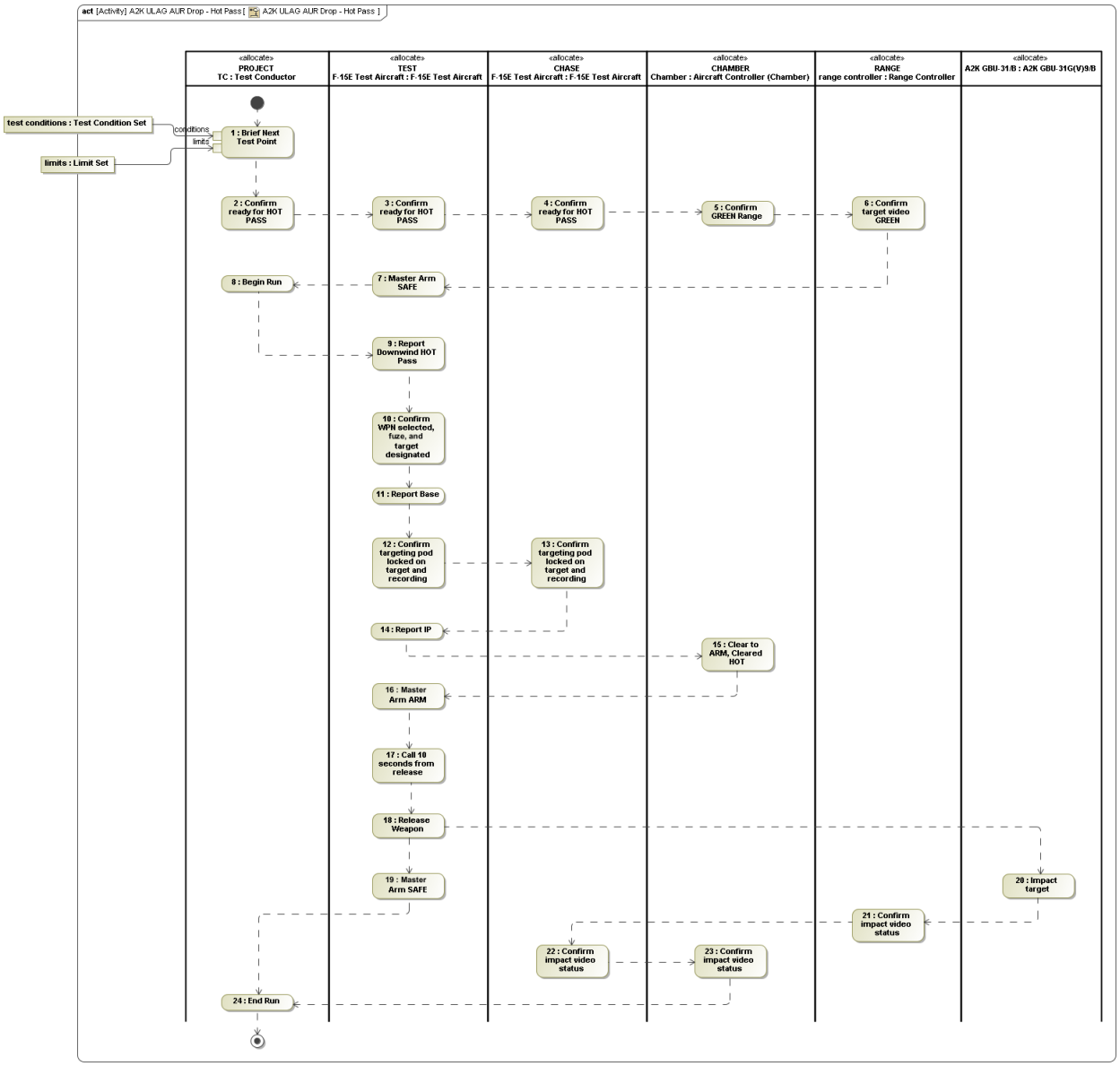

Figure 15. A2K ULAG Hot Pass Test Card Activity Model

4.3.1.3. Test Conditions and Limits

As the test card diagram shows, the inputs are the standard inputs for test cards: test conditions and test limits. This was one of the areas where describing the MBTE modeling approach was much simpler with a specific example. Test conditions and limits needed to be flexible enough to accommodate flight parameters, system settings, test-to-target geometry, or any other key values required for the correct execution of the test point. It is also important, especially when going from digital planning to live execution, to limit the specified test conditions and limits to those that are essential for test crews to focus attention on during executing the test point, not an exhaustive list

of all possible parameters. From a modeling perspective, this is accomplished by assembling a test condition instance composed of several key value instances. Key values are instances that specify a parameter and value for that parameter. An example of key values is shown in Figure 16.

#	Name	Key	Value
1	1	Altitude	15,500 ± 500 ft MSL
2	2	Mach	0.60 ± 0.05 M
3	3	Impact Angle	90 degrees

Figure 16. Sample Test Condition Key Values

Test condition and limit instances combine to form the specific test conditions for a test card. The modeling approach for this is that a test condition set or limit set is composed of 1 or more key value parameters by specifying it with a multiplicity of (1..*). In this manner, the test team can model the specific test conditions needed for the test card. The sets of test conditions for the MBTE Pilot Project are shown in Figure 17.

#	Name	test conditions.Key	test conditions.Value
1	test condition 1	Mach	0.60 ± 0.05 M
		Impact Angle	90 degrees
		Fuze	FMU-167/B
		Altitude	15500 +/- 500 ft MSL
2	test condition 2	Mach	0.60 ± 0.05 M
		Impact Angle	90 degrees
		Fuze	FMU-152 A/B
		Altitude	15500 +/- 500 ft MSL
3	test condition 3	Mach	0.60 ± 0.05 M
		Impact Angle	90 degrees
		Fuze	FMU-139D/B
		Altitude	15500 +/- 500 ft MSL

Figure 17. MBTE Pilot Project Test Conditions

4.3.1.4. Linking Value Properties to Test Conditions

During the review process, the safety profile was changed to an alternate profile on the Eglin Range in order to safely drop live ordinance on the land range and ensure sufficient time for the test item to operate. These parameters were updated in the Release Test Procedures block's value properties, shown in Figure 18.

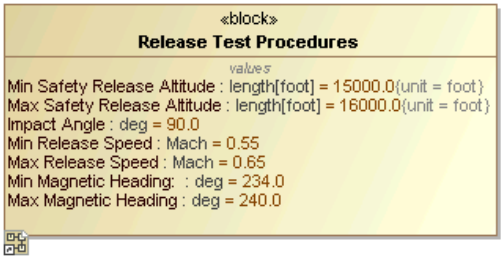

Figure 18. Release Test Procedures with Safety Profile Value Properties

The change in safety profile necessitated a change in the test conditions. There was no linkage between the value property and the test conditions instances, so they had to be updated separately. The test team noted that this deficiency fell short of the goal of updates propagating through the model, so the MBSE team looked to find a way to link the value property to the test condition instances. Figure 19 shows the structure and behavior models of an approach for linking them and updating the test conditions. This approach also was able to perform a conversion from a maximum and minimum value to a target test execution parameter with data band for the test card.

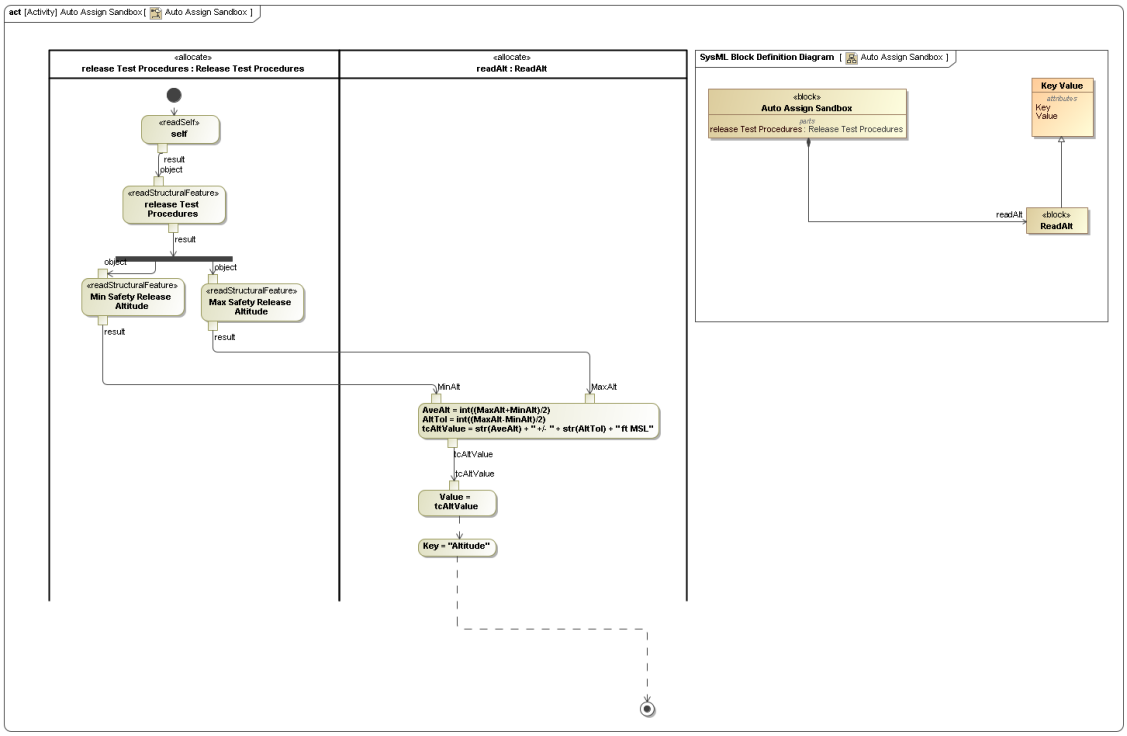

Figure 19. Utilizing Cameo Simulation Toolkit to Link Value Properties to Test Condition Instances

4.3.2. Test Resources

4.3.2.1. Resource Modeling

As discussed in the Metamodel Overview, test resources are one of the key parts of a MBTE model. Figure 20 shows the resource model overview diagram for the Pilot Project. Most of this resource model is straightforward, but there are a few notable features to highlight. The primary test plan was to drop two of each of three fuze types with two F-15E aircraft sharing test aircraft and chase roles. However, the minimum data requirements were one of each fuze type which can be accomplished by a single F-15E and the safety chase role could be filled by an F-16. The resource model shows this with a multiplicity of 1 or 2 F-15E aircraft and the role of safety chase being filled by either the F-15E or and F-16. Similarly, the test plan required collection of aircraft Time Space Position Information (TSPI), but the Eglin instrumentation has several means of collecting this, all of which would be valid for the test. So a TSPI pod is listed as the required resource which may be fulfilled by any one of three options.

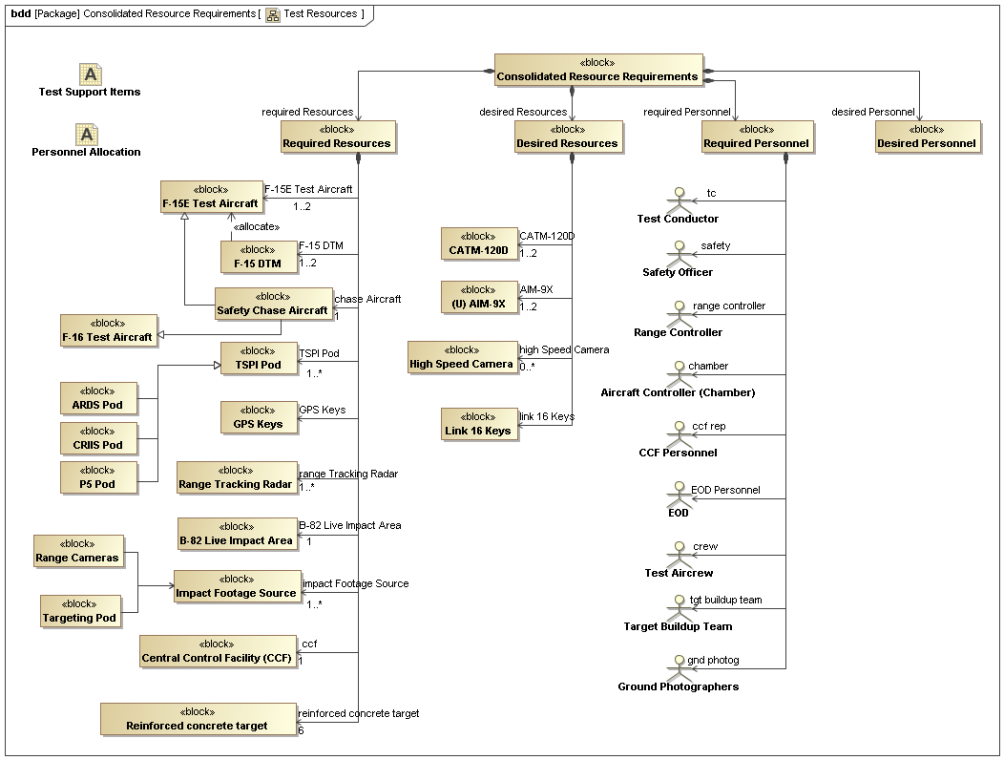

Figure 20. MBTE Pilot Project Resource Model

4.3.2.2. Resource Allocation

In order to make the test resource plan a reality during execution, these resources need to show up during execution. When test programs require the participation of several organizations, knowing who is bringing what to the test is essential to efficient test execution. A MBTE model ensures that the resources are matched to participating organizations through allocation relationships. Figure 21 shows the resource allocation matrices for both resources and personnel.

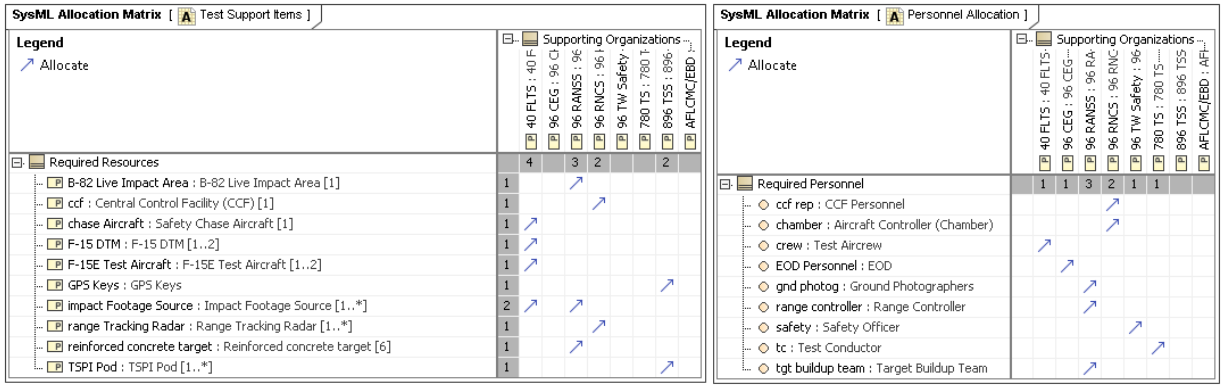

Figure 21. Resource Allocation to Organizations

4.3.2.3. Data Requirements

One of the features of a MBTE model not previously discussed is the use of requirement elements for identifying the data needed for test points. These are test specific requirements and not system requirements, which is another advantage of having a test model that is independent of the system model. These data requirements have a «satisfy» relationship with the test resources that collect the data specified. Figure 22 shows the pilot project data requirements, the resources used to satisfy the data requirements, and the organization responsible for providing the data to the program office with an allocation relationship.

#	Name	Text	Satisfied By	Allocated To
1	Detonation Video	Video of the weapons detonating on the target shall be provided electronically to the program office within 7 days of the flight test.	Range Cameras Impact Footage Source Weapon Detonation	780 TS
2	TSPI (Aircraft)	Aircraft TSPI data shall be provided electronically to the program office within 7 days of the flight test.	Test Aircraft Stores Test Aircraft Instrumentation	780 TS

Figure 22. MBTE Data Requirements

4.3.2.4. Sortie Loadout

The final highlight regarding test resources is the specific weapon loadout for the test sortie. In the Test and Evaluation section of the model, each flight and ground test event is represented by a block. The classifier behavior for the flight test blocks is where the cards are ordered for the flight in an activity model. Because of this structural element representing the sortie, an internal block diagram can represent the stores loadout when the resources are related to the flight test with directed aggregation relationships creating reference properties for the flight test sortie block. Figure 23 shows this internal block diagram with both test resources and the SUT model elements represented.

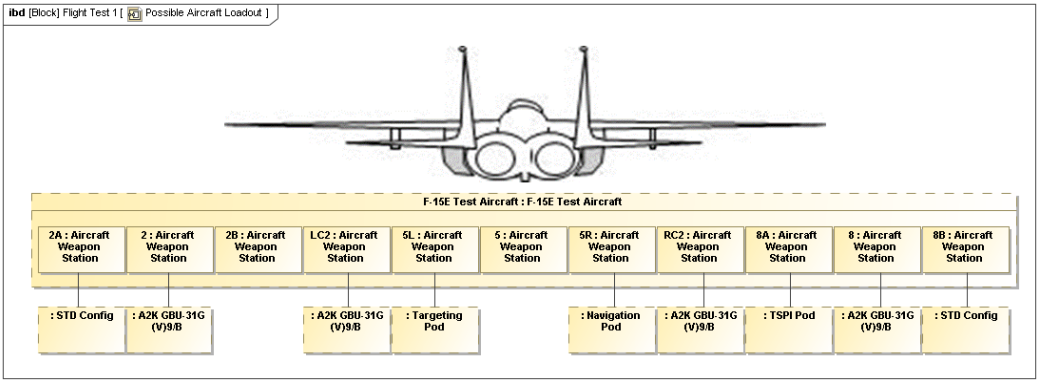

Figure 23. Test Sortie Aircraft Loadout Model

4.3.3. Safety Planning

The document-based method of developing safety plans and mitigation in the 96 TW consisted of the test team conducting risk analyses with a Test Hazard Analysis process, writing proposed hazards and mitigation steps into the Method of Test, having those reviewed by cognizant safety authorities, and those safety authorities documenting the final findings in a Safety Appendix to the overall test directive. This results in two places where safety information is documented inside the test directive: the draft information in the method of test and the authoritative information in the safety appendix. Two highlights of MBTE-based safety planning are highlighted below.

4.3.3.1. THA modeling

The Test Hazard Analysis allows a structured means of assessing the risk of conducting a test both before and after applying risk mitigation. As is common practice, the risk assessment is determined by assessing the severity and probability of the hazard. In the document based process, the test team often has to refer to a printed matrix to find the risk level according to published information. By coding this information in a constraint block, the model can automatically calculate the risk level for given severity and probability and store that information in a Test Hazard Analysis model element. Figure 24 shows an example of a modeled test hazard analysis.

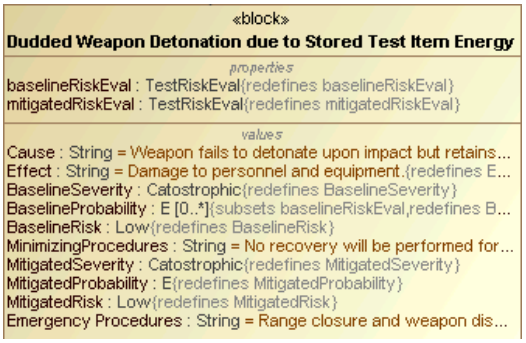

Figure 24. Example Test Hazard Analysis Model Element

4.3.3.2. Safety Requirements

The MBTE model must also capture the specific direction from the safety authorities. This is another area where using requirement model elements can serve a test-specific purpose. Figure 25 provides an example of a general risk mitigation safety requirement on the test program. A challenge some test teams face when using the document centric approach, especially with a large, complex test program, is determining which of the specific mitigation requirements apply to a particular test point or test sortie. Some mitigation requirements apply to all test points, while others only apply to a specific subset of test points. By creating relationships in the model between the safety requirements and the test points, these distinctions can be made during the test planning phase and the model can highlight, when a test sortie is created, what the specific safety requirements for that particular set of test points. This ensures the test team is focused on the most relevant risk mitigation on a sortie-by-sortie basis and also that no safety requirements are accidentally missed due to oversight.

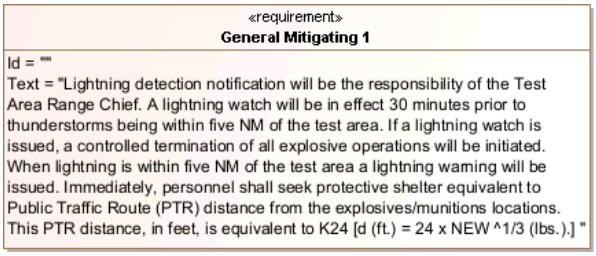

Figure 25. Example Safety Requirement

5. Lessons Learned & Assessment

5.1. What worked

5.1.1. Linking SUT model and test model

A common source of delays and confusion in test planning is a lack of truth data about the SUT. By connecting the Test Model to the SUT Model, and reusing those model elements as much as possible, these delays and potential inaccuracies can be avoided. This was successfully demonstrated with the MBTE pilot project, where the physical structure of the SUT was pulled directly into the Test Model.

5.1.2. Using MBSE techniques to develop a test plan

One of the key benefits of MBSE is the traceability and consistency that it enforces. Document-based test plans often require the test team to manually update the document on an ad-hoc basis as the SUT design matures or requirements change. By adopting a model-based approach, changes automatically propagate throughout the test plan and any inconsistencies are easily identified. This was demonstrated several times throughout the pilot project by uncovering inconsistencies in data requirements and range resources.

5.1.3. Linking detailed test procedures to test objectives and requirements

Test cards often require the synthesis of data from dozens of disparate sources, including flight manuals, flight clearances, safety packages, and SUT technical requirements. The overlapping "Venn Diagram" region where all these factors allow test execution can often be very small, and a change to any one of these data sources could drive the test point outside of that region. By modeling test procedures as activities, constrained by the parameters of the various competing requirements, the test team can ensure that the cards reflect a plan for safe and effective execution. This was illustrated in the pilot project when the allowable release parameters for the range safety profile were found to conflict with the desired technical release condition (see Linking Value Properties to Test Conditions).

5.1.4. Resource/personnel tracking and allocation

For large test programs, tracking test points and data requirements across dozens of sorties can become very cumbersome. It can also be a challenge to ensure that all of the required resources and personnel are scheduled for each sortie. They SysML Allocation Matrix is ideal for this application and was demonstrated successfully in the pilot project for both resource and personnel requirements. While somewhat trivial for the scale of the pilot project, these allocation matrices represent a powerful benefit of MBTE.

5.1.5. Ability to generate document-based artifacts from the model

Access to MBSE tools is not yet ubiquitous and may never be, due to the expense of licenses and the time commitment required to build SysML fluency. However, stakeholders who are not part of the core test team may still need access to the test plan. The MBSE created a Velocity Template Language (VTL) script to output the model contents into a "legacy" style test plan. This proved to be necessary for the pilot project, where stakeholders like the Explosive Ordnance Disposal (EOD) technicians needed a hard copy they could take with them to the range.

5.2. What didn't work

5.2.1. Limited access to IDE

Collaboration is an important component of MBSE, and many collaboration tools require access to a cloud-based IDE. Due to budget and manpower constraints, several members of the test team were unable to have accounts on the IDE. This resulted in configuration management challenges, uneven workload, and frequent confusion about the status of the Test Model. For future digital programs, access by all key stakeholders to the IDE will be critical.

5.2.2. Disconnected SUT development model and published model

A similar infrastructure limitation led to an "air gap" between two version of the SUT model used for the pilot project. The development model was hosted on a contractor's cloud server, while the published model (the version linked to the Test Model), was hosted on the same server used by the test team. The published model was updated infrequently, roughly once per month. This led to

out-of-date references in the Test Model as updates were made to the SUT development mode. This example further highlights the need for all program stakeholders to have access to the same IDE. Without it, the Digital Thread gets broken and may of the benefits of MBSE are diminished.

5.3. What can get better with technique refinement?

5.3.1. Training and expertise needs in a test organization

The training plan implemented by the 780th was found to helpful, but not fully sufficient for the purposes of the pilot project. Several gaps were identified throughout, including the ability to create a complex model from scratch, the use of collaboration tools like Teamwork Cloud, building diagrams optimized for technical reviews, and configuration management processes. These topics should be included in training for any organization adopting MBTE.

5.3.2. Sharing with stakeholders

Acknowledging the previously mentioned limitations of IDE access, in most organizations, it will not be practical to expect every single stakeholder to have access to the IDE directly. Therefore, it is imperative to have a plan to share the model, or at least its contents, with those other stakeholders. For the pilot project, this was done by periodically exporting the model in its native .mdzip format, as an HTML report, or simply as a Word document by running the VTL script. The .mdzip file is the highly preferred format, because it includes the full contents of the model. However, it requires the user to have a Cameo license on their local machine. The HTML report avoids this requirement, but strips some of the details to which the reviewer could otherwise navigate. Since the Word document has the least fidelity, it is best reserved for those who have no SysML exposure at all, or who require a hard copy of the test plan.

5.3.3. Configuration management

Configuration management became a challenge throughout the pilot project, exacerbated by the lack of IDE access by many of the test team members. 96 TW processes require test plan and safety plan amendments for most types of changes after initial approval. Document-based test plans are digitally-signed and archived in an online repository. This repository is not compatible with .mdzip files, so a memorandum was routed instead, specifying the approved version of the model. Once changes were found to be necessary, it became difficult for the entire test team to track the status of the model updates due to limited IDE access. Teamwork Cloud provides "branch and trunk" and functionality that was not utilized efficiently during the pilot project. A recommended future practice is to merge the branches back to the trunk at each of the test and safety planning milestones (Technical Review Board [TRB], Safety Review Board [SRB], Test Approval Brief [TAB], etc.), and to update organizational processes to eliminate the additional document-based process which adds little value.

5.3.4. Technical and safety reviews from the model

For the pilot project, all technical and safety reviews were conducted directly within the model, without using any PowerPoint slides. The main benefit of this approach is that the presenter is

showing the content of the *actual* test plan – not a paraphrased version. It also allows the presenter to navigate anywhere within the model in real-time as required by the discussion. The test team created TRB-, SRB-, and TAB-specific diagrams to reflect the content that would normally be presented in the "legacy" format. This approach was met with mixed reactions from various stakeholders. The most common critiques were that the diagrams were difficult to read and that read-ahead materials were difficult to navigate for those unfamiliar with Cameo. The test team improved significantly from TRB to TAB by adjusting the formatting, including images, and using Presentation mode (available starting with Cameo version 2021x and later). This is an important aspect for a test team to get right if an organization is to move towards a model-based approach. These reviews maybe be the first exposure to MBSE for many stakeholders, and if they are unable to read or navigate the model, they are less likely to buy in to the new process.

5.3.5. Data-centric vs documentation-centric mindset

The dynamic nature of MBSE models is what makes them so powerful. The pilot project test team missed a few opportunities to take advantage of this capability by continuing to use a document-centric mindset throughout the initial stages of test planning. This is illustrated well in the Test Objectives section, where the Measures of Performance were originally modeled as static text, rather than data-rich value properties. This data-centric mindset should be considered at the organizational level as well. Document-centric processes may force a test team to create suboptimal models by forcing compliance with incompatible processes. Data- or model-centric processes may also leverage the benefits of collaboration tools, like Teamwork Cloud, to accelerate test approval timelines.

6. Conclusions

As digital engineering becomes more standard in system development, the test community must be ready to integrate into digital environments. This paper presented a metamodel and methodology for Model Based Test Engineering using SysML that gives the test community a means to take advantage of MBSE principles, increases technical rigor, and provides a dynamic, data-driven linkage between the system model and the test model. The MBTE approach presented allows for fully physical tests, fully digital tests, or any hybrid approach that the testers of the System Under Test deem appropriate. This paper also presented highlights of a pilot project executed at the 780th Test Squadron at Eglin AFB, FL illustrating practical examples of how to implement MBTE, along with lessons learned by the team during the pilot program.

7. Future work

Due to external factors not related to MBTE, the pilot project execution was delayed beyond the required date for inclusion in this paper. So there remains opportunity for learning on the initial pilot project, especially with regards to automated reporting. While the experience gained during metamodel development and the pilot project work to date provide a solid foundation for MBTE, there are several topics that are obvious areas for future work. These areas include:

- **Adapt metamodel to other organizational test processes:** The metamodel development was deliberately focused on being adaptable and currently works with flight and ground tests in mind. However, the framework is specifically formatted with the labels, language, and processes of the 96th Test Wing. Within the Air Force, the model should be adaptable for the variations of the test process at the 412th Test Wing and AFRL's Rocket Propulsion Division both located at Edwards AFB, CA and to the Air Force Operational Test and Evaluation Center (AFOTEC) processes for complete documentation of a program developmental and operational test strategy. The scope of this future work could further expand to other services, commercial test, or allied countries.
- **Larger, more complex pilot projects:** The scope of the pilot project was deliberately kept small and simple to reduce the risk and enable troubleshooting, metamodel refinement, and new development during execution. There may be unknown stumbling blocks, inefficiencies, or areas of opportunity as the methodology scales up to more complex test programs. These larger pilot projects are a prudent step before widescale adoption.
- **Digital twin test execution:** The MBTE approach was designed to accommodate fully physical tests, fully digital tests, or any hybrid combination that best suits the system test strategy. The pilot project was able to exercise the physical test side of the methodology, but a fully automated test of a digital twin system is appropriate for exploring the full scope of the intended uses of MBTE.
- **Data mining/dashboards:** DE makes many data sources on a system and program execution available to various stakeholders. Future MBTE development should take advantage of the availability of these data to provide program/test management insight in the form of data driven dashboards.
- **APIs to external systems:** DE ecosystems are meant to make authoritative data sources available to program stakeholders. While utilizing MBSE techniques made sense for test planning, there are undoubtedly test-relevant data sources where a MBSE model is not the most efficient manner to store, manipulate, or access data required. Tool integration is essential in a full digital transformation, so designing Application Programming Interfaces (APIs) to those other data sources will lead to a more integrated digital test enterprise.

REFERENCES

[1] W. Roper, "There is No Spoon: The New Digital Acquisition Reality," Department of the Air Force, Washington, DC, 2020.

[2] N. McBee, "What is Model-Based Systems Engineering," Ansys, 25 05 2022. [Online]. Available: https://www.ansys.com/blog/model-based-systems-engineering-explained.

[3] S. V. Nath and P. v. Schalkwyk, Building Industrial Digital Twins, Birminham, UK: Packt Publishing Ltd., 2021.

[4] M. W. Grieves, "Virtually Intelligent Product Systems: Digital and Physical Twins," *Complex Systems Engineering: Theory and Practice,* vol. 256, pp. 175-200, 01 01 2019.

[5] A. M. Madni and S. Purohit, "Economic Analysis of Model-Based Systems Engineering," *Systems,* vol. 7, no. 1, 20 February 2019.

[6] Object Management Group, "OMG Systems Modeling Language (OMG SysML™) Version 1.6," OMG, Needham, MA, 2019.

[7] MBSE Execution, "Model-Based Testing Using SysML," 13 09 2018. [Online]. Available: https://www.youtube.com/watch?v=bDIa9ooW4nI.

[8] E. Holmes, "Integrated Digital Environment provides glue for digital campaign," Air Force Material Command Public Affairs, 29 July 2021. [Online]. Available: https://www.afmc.af.mil/News/Article-Display/Article/2712000/integrated-digital-environment-provides-glue-for-digital-campaign/.

BIOGRAPHIES

Johnston A. Coil. Johnston "CASS" Coil is a Digital Systems Engineer and MBSE Architect for the LinQuest Corporation. Prior to joining LinQuest, he had a 21-year career in the US Air Force, with 16 of those years as a Flight Test Engineer. He supported the testing of the F-22 Raptor, CP140 Aurora, E-8C JSTARS, and multiple other C4ISR and weapons test programs. He earned a B.S. in Astronautical Engineering from the US Air Force Academy, a M.S. in Mechanical Engineering with aerospace focus from the University of Colorado at Colorado Springs, and is an FTE graduate of the US Air Force Test Pilot School.

Steven T. Wachtel. Maj Steven "Rumble" Wachtel is a Flight Test Engineer assigned to the 718th Test Squadron, Operating Location Peterson SFB, CO. He previously served as the Assistant Director of Operations at the 780th Test Squadron. He earned a M.S. in Flight Test Engineering from the U.S. Air Force Test Pilot School in 2020, a M.S. in Systems Engineering from the Air Force Institute of Technology in 2017, and a B.S. in Mechanical Engineering from The Ohio State University in 2012.

Robert L. Riley, Jr. Robert "Rob" Riley is the Digital Engineering Lead for the Experimental Demonstrations Branch within the AFRL Aerospace Systems Directorate. Across his 20-year federal civilian career, he has served in various R&D, test, systems engineering and acquisition management roles. He has supported multiple test activities including lab flight test projects for

weapon data links and tactical network management technologies. He holds dual B.S. degrees in Electrical Engineering and Physics from Tuskegee University, a M.S. degree in Electrical Engineering from North Carolina A&T State University, and a M.Eng. degree in Systems Engineering from Cornell University. He is also a qualified Joint Interface Control Officer.