Recent Advances in Aerospace Actuation Systems and Components, September 25-27, 2023, Toulouse, France

BUILDING A REQUIREMENTS DIGITAL THREAD FROM CONCEPT TO TESTING USING MODEL-BASED STRUCTURED REQUIREMENTS APPLIED TO THRUST REVERSER ACTUATION SYSTEM DEVELOPMENT

HERBER Daniel R.
Colorado State University
Fort Collins, CO, USA
Phone: +19704911491
Email: daniel.herber@colostate.edu

EFTEKHARI-SHAHROUDI Kamran
Woodward, Inc.
Fort Collins, CO, USA
Colorado State University
Fort Collins, CO, USA
Email: kamran.eftekhari_shahroudi@colostate.edu

ABSTRACT

To address the concerns regarding textual requirements in general, a recently introduced concept of model-based structured requirements (MBSRs) was put forth that combines textual structured requirements and SysML modeling concepts. In this paper, we will discuss the advances made to MBSRs in the context of the development of the Aero-actuation Systems Engineering Test (ASET) Lab testbed. By using MBSRs within the system model, a digital thread is created between the various stages of system development, leveraging the single-source-of-truth model. This approach will highlight how MBSR (and MBSE more generally) can help organizations better realize their goals for advanced aero-actuation systems, thus demonstrating the potential value of model-centric system development practices.

KEYWORDS

Aero-actuation, model-based systems engineering, requirements, system development, systems engineering, testing, verification and validation

I INTRODUCTION

As organizations seek to become more efficient, reduce risk, and bring ever-complex products to market, they become increasingly interested in cutting-edge approaches for more effective systems development that will help realize their goals. Critical to successful product research and development (R&D), deployment, and sustainment is effective *testing*, providing essential verification and validation (V&V) knowledge. To realize this part, a catalog of testbeds, plans, procedures, cases, conditions, requirements, standards, results, etc., must be created and managed along with considerations for the humans in the loop. Therefore, there is much to consider to make this happen, especially for newer product concepts with less heritage to inform effective development and testing. In this paper, we will describe a technical approach to help with this challenge through a motivating case study of developing and managing a testbed for electric aero-actuation components.

1.1 Requirements

A requirement is a statement that "translates or expresses a need and its associated constraints and conditions" (IEEE, 2018) and is essential throughout a product's lifecycle. Several sets of guidelines (2018; Pohl and Rupp, 2015; Wheatcraft et al., 2022) have been established, providing the following general characteristics of well-defined requirements: necessary, appropriate, unambiguous, complete, singular, feasible, verifiable, correct, and conforming. However, creating requirements that adhere to these characteristics takes time and effort. This risk to effective development is due to the ever-changing nature of system understanding, and the sheer volume of interrelated shall statements across different groups. Modern approaches to requirements development and management should help an engineer produce systematically well-defined requirements with lower effort, practical digital threads to R&D and testing results, and efficiently reusing relevant aspects of previous efforts and organization best practices.

1.2 Model-based Approaches

Central to this work is the usage of modern model-based perspectives of developing and understanding systems, often falling under the general term model-based systems engineering (MBSE). In general, mounting formal and informal evidence suggests that MBSE has the potential to improve efficiency, rigor, and quality during system development (Carroll and Malins, 2016; Huldt and Stenius, 2018; Madni and Purohit, 2019). The model here is an abstract representation of a system and its development captured through structure, behavior, and rules and their relationships. Such a comprehensive model is often associated with the notion of systems architecture. The "model-centric" perspective is perhaps best characterized by the principle: for every concept, there is a singular, unique element in the model that is used as much as necessary. This perspective contrasts with more "document-centric" practices where static (potentially incorrect, incomplete, or convoluted) artifacts such as documents and spreadsheets are the primary means for capturing system development activities.

The benefits of model-based approaches are properly realized when they effectively provide support for various activities during a product lifecycle, including testing. Now, there is codified and disciplined support for MBSE in the form of methodologies, formal modeling languages, and tools (Borky and Bradley, 2019; Friedenthal et al., 2015). However, it is still often the case that there are gaps and other challenges that might preclude its immediate usage in a particular domain (Call and Herber, 2022; Huldt and Stenius, 2018). Fortunately, MBSE concepts and techniques are extensible to meet the needs of particular communities and organizations.

1.3 Systems Model Language (SysML)

The Systems Modeling Language (SysML) is a general-purpose, widely-supported modeling language for creating system architecture models. The SysML standard prescribes a set of constructs and allowable relationships, thereby supporting the creation of an integrated system model (OMG, 2019). The language supports modeling with object orientation (OO) principles of abstraction, encapsulation, modularity, generalization and inheritance, aggregation and composition, interfaces, and polymorphism (Borky and Bradley, 2019), supporting a rich and precise way to capture systems. Software tools are the key enablers of this form of system modeling and MBSE, providing means to interact with the model through diagrams and other forms. One popular tool (and the one used in this work) is Cameo Systems Modeler (CSM), which implements SysML along with other features to support MBSE activities (No Magic, 2020).

SysML is often described at a high level through the four pillars of structure, behavior, requirements, and parametrics (Friedenthal et al., 2015). There are many modeling constructs and diagram types in SysML, and it supports customized extensions through the stereotype mechanism (denoted «*stereotype*») and other tool-specific features.

While classical SysML requirements provide a diverse set of traceability and hierarchy relationships to the SysML model, the definition of the requirement is primarily left to the *text* attribute, which needs to be interpreted, can be prone to errors, and is not directly linked to the current system understanding. In modern fast-paced, budget-conscious development, this definition may cause issues.

1.4 Overview

This paper will utilize a modified model-based structured requirement (MBSR) approach in SysML from Herber et al., 2022 to develop high-quality requirements with a focus on testing-related concerns. The rest of the paper is organized as follows: Sec. 2 describes MBSRs and its current implementation; Sec. 3 presents an aero-actuation testbed case study using MBSRs; and Sec. 4 provides some conclusions.

II MODEL-BASED STRUCTURED REQUIREMENT (MBSR) IN SYSML

This section presents the MBSR concept from 2022 and its minor modifications. Please see the original reference for a more detailed overview and description of how the MBSR fits within classical SysML requirements modeling.

2.1 Structured Requirement

A structured requirement (SR) (Carson, 2015) or template requirement (Pohl and Rupp, 2015) defines an orderly structure with specified attribute placeholders (i.e., a sentence blueprint) to help capture the precise meaning and communicate the required information to define a complete requirement. The particular SR statement template based on Carson, 2015 considered here is:

[**Who**] *shall* [**What**] [**How Well**] under [**Condition**]. (1)

where we break down the pieces of Eq. (1), including each of four bold attributes, as:

- [**Who**]: The element (product, service, user, activity, etc.) which is subject to the remainder of the statement. If the requirement is satisfied, this element would be said to have the desired characteristic or is capable of performing the intended function.
- *shall*: Indicates that this entire statement is mandatory. However, as discussed in Pohl and Rupp, 2015, this could be replaced by alternative "legal obligations", such as *should*, *will*, and *may*.
- [**What**]: Refers to an element, either as an observable characteristic or function, that will now have a measurable legal obligation in relation to [**Who**].
- [**How Well**]: A measurable condition to assess how well the legal obligation for the [**What**] attribute is met.
- [**Condition**]: Describes the stipulated complete set of operational scenarios, states, and environmental conditions when the legal obligation must be met. This set can include triggering or initiating events (Carson, 2015).

As an example, the following is a SR written in natural language:

[**The testbed**] *shall* [**transition to safe mode**] [**within 2 sec**] under [**enclosure door open event**]. (R1)

With this example, we might have arrived at the same final statement without the SR template. Still, it is less of a guarantee as well as there is a more immediate comprehension of the requirement statement.

Equation (1) is not the only SR form. Different general templates, such as the ones in Pohl and Rupp, 2015 or ones based on the 29148-2018 standard (IEEE, 2018), are possible. Furthermore, templates with structure and attributes tailored to different requirement types (e.g., functional, non-functional, interface, design constraint, and operational) may also be of use (Carson, 2015; Pohl, 2010).

Overall, this approach to requirements creation helps improve their quality by helping meet the general characteristics of well-defined requirements in Sec. 1.1 (in particular, appropriate, unambiguous, complete, singular, correct, and conforming). However, as currently presented, these techniques offer only a refinement to the classical textual requirements and have no relationship to a system (SysML) model and only an imprecise relationship to the broader system context.

2.2 Model-based Structured Requirement

Directed by the textual nature of both SRs and classical SysML requirements, MBSRs provide a mechanism for the usage of SysML modeling elements to define a SysML SR (Herber et al., 2022). Using the original SysML «*Requirement*» stereotype (which defines the requirement element type), a «*Structured Requirement*» is defined that inherits all the properties of «*Requirement*» using the generalization relationship. Then the four attributes from the SR format in Eq. (1) are added to «*Structured Requirement*». These attributes for a given MBSR can be linked to other model elements that represent aspects of the physical system, precise mathematical conditions, test cases, etc. Selected model elements will have much more than their name (as is the case in textual SRs) defining what it is and where it fits within the system. This capture of meaning can include more SysML constructs but also documentation and links to external artifacts, input data, and testing results. Complete examples will be shown in Sec. 3.

2.3 Changes to SysML MBSR Stereotype

This section will describe the changes made to the original MBSR SysML stereotype definition based on additional work using MBSRs and feedback from the community. These changes are in the CSM model at 2023. The four main changes are:

1. While the base names of the four attributes have remained the same (although there has been some good discussion regarding an update to [**Agent**] or [**Entity**] instead of [**Who**]), numbered prefixes were added to each to attribute based on their ordering in Eq. (1). Furthermore, adding these numbers ensures that a reader of the attributes list immediately knows their intended order.
2. The two additional Requirement Type and Rationale attributes are now removed from the «*Structured Requirement*» as it was determined that they are not strictly necessary for MBSRs. These can always be included as organizational requirement attributes (2022). Furthermore, the SysML non-normative requirement extensions already include several requirement types and the source attribute (OMG, 2019, Sec. E.3).
3. One advantage of the model-based definition of the attributes is the ability to (forcefully) guide the user only to use certain element types to define specific attributes (Herber et al., 2022). Originally, [**Who**] was typed as Block, [**What**] as Behavior, [**How Well**] as ConstraintBlock, and [**Condition**] as NamedElement. While these types are perhaps the most common selections, the development of a more diverse set of requirements necessitated broader options. For example, [**Who**] might need to be an Activity that is being developed but not explicitly assigned to a structural element yet, [**What**] a ValueProperty to select the mass of a component, or [**How Well**] an instance of a ConstraintBlock with specific limit values defined. Therefore, all attributes are now typed with NamedElement, allowing for a broader selection of model elements. This change is also consistent with other requirement relationships satisfiedBy and verifiedBy in the SysML specification (OMG, 2019). Modifications to the MBSR definition (perhaps for specific requirement types) could still restrict attribute types as needed.
4. Multiplicities in SysML with the MBSR definition specify the number of selectable elements for a given attribute. For example, [**Who**] was designated [1] to indicate exactly one element needs to be selected. This value is not changed here to maintain the singular nature of a given MBSR. With a similar rationale, [**What**] (which previously had no multiplicity) is assigned [1] to convey that only one observable characteristic or function is part of the MBSR. Continuing, [**How Well**] (which previously had no multiplicity) is assigned [1..*] to state that there must be at least one condition to be checked, but more than one might be necessary to assess the satisfaction of the requirement. When there are multiple elements, the interpretation is that all conditions must not fail. Finally, [**Condition**] remains [1..*]. If the requirement must hold for all considered system circumstances, it is now recommended that a model element is defined along the lines of All System Conditions and derived requirements for specific condition scenarios are made as needed.

III ASET LAB TESTBED CASE STUDY

Shown in Fig. 1, the testbed in the Aero-actuation Systems Engineering Test (ASET) Lab was designed for flight controls integrators, component suppliers, and researchers collaborating on complex anomaly detection and anomaly tolerant control topics where cost and study turn-around time are a major concern. It can create complex electrical, software, and mechanical anomalies resulting from the interplay of sensors, actuators, power supply (converters), and loads in electrically and/or mechanically synchronized load-sharing actuation systems. The initial design was a collaboration between CSU Systems Engineering faculty/students, Woodward aero engineers, and interns (from INSA Toulouse and Lyon). What is shown here is only a simplified version of a more complete SysML model in CSM used to support the development and running of the testbed.

3.1 Door Open Shutdown MBSR Example

We will first illustrate the MBSR concept with a single requirement intended to capture a safety concern, named Door Open Shutdown. Illustrated in Fig. 2, we see the MBSR with

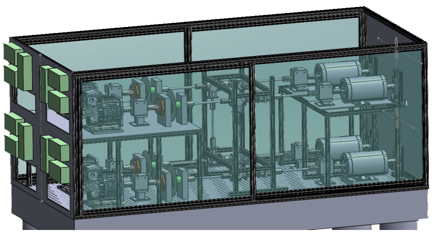


Figure 1. Reconfigurable ASET Lab testbed with complex anomaly generation capabilities at the CSU Powerhouse Energy Campus

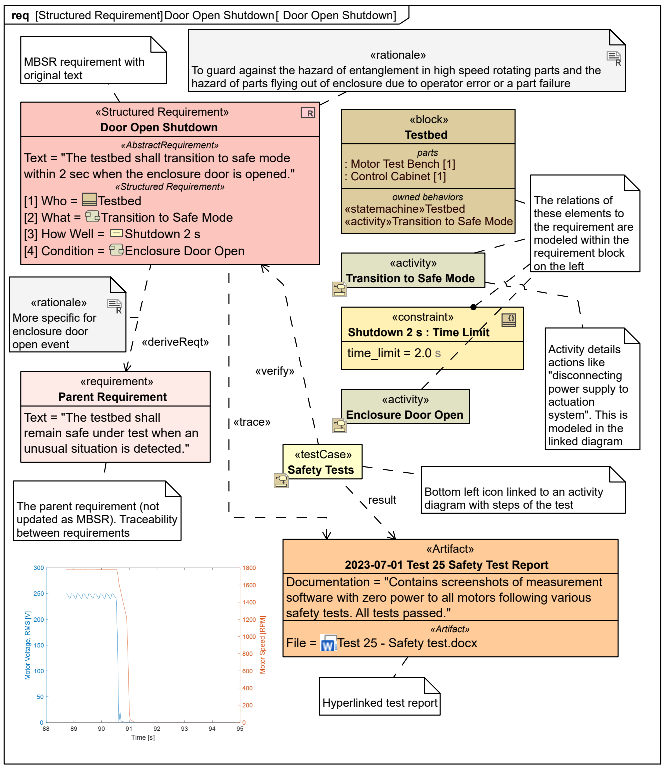
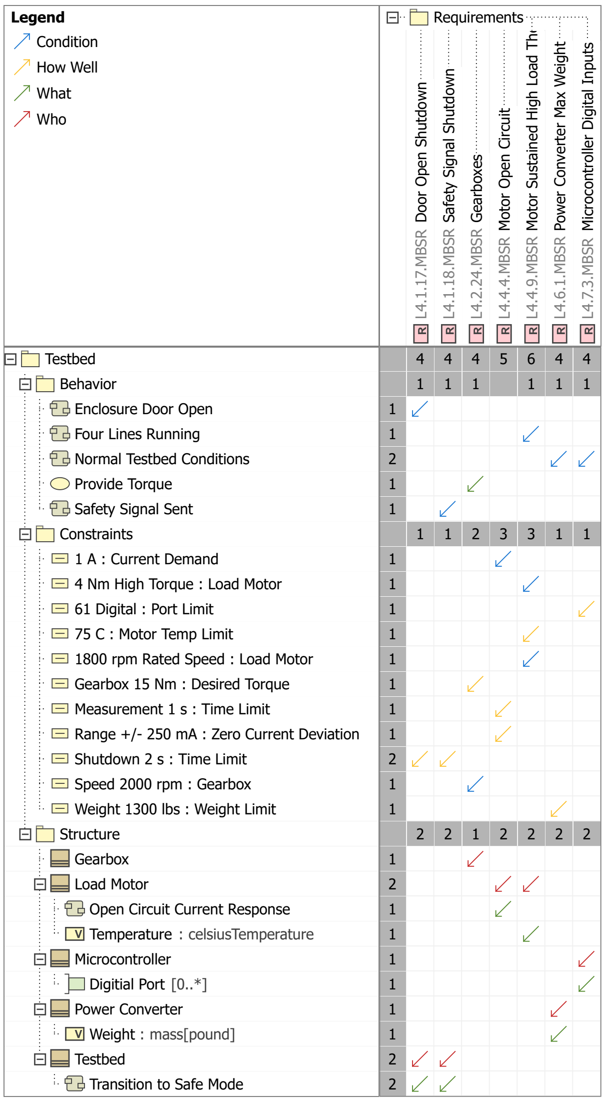

req [Structured Requirement]Door Open Shutdown[Door Open Shutdown]
MBSR requirement with original text
«rationale» To guard against the hazard of entanglement in high speed rotating parts and the hazard of parts flying out of enclosure due to operator error or a part failure
«Structured Requirement» Door Open Shutdown
«AbstractRequirement»
Text = "The testbed shall transition to safe mode within 2 sec when the enclosure door is opened."
«Structured Requirement»
[1] Who = Testbed
[2] What = Transition to Safe Mode
[3] How Well = Shutdown 2 s
[4] Condition = Enclosure Door Open
«block» Testbed
parts
: Motor Test Bench [1]
: Control Cabinet [1]
owned behaviors
«statemachine»Testbed
«activity»Transition to Safe Mode
The relations of these elements to the requirement are modeled within the requirement block on the left
«activity» Transition to Safe Mode
«constraint» Shutdown 2 s : Time Limit
time_limit = 2.0 s
«activity» Enclosure Door Open
Activity details actions like "disconnecting power supply to actuation system". This is modeled in the linked diagram
«rationale» More specific for enclosure door open event
«deriveReqt»
«verify»
«trace»
«requirement» Parent Requirement
Text = "The testbed shall remain safe under test when an unusual situation is detected."
«testCase» Safety Tests
Bottom left icon linked to an activity diagram with steps of the test
result
The parent requirement (not updated as MBSR). Traceability between requirements
«Artifact» 2023-07-01 Test 25 Safety Test Report
Documentation = "Contains screenshots of measurement software with zero power to all motors following various safety tests. All tests passed."
«Artifact»
File = Test 25 - Safety test.docx
Hyperlinked test report
Motor Voltage, RMS [V]
Motor Speed [RPM]
Time [s]

Figure 2. Door Open Shutdown as an MBSR summarized in a requirements diagram with related model elements and relationships

the original textual description and the four attributes filled in with the appropriate model elements to define the parts of the MBSR. First, the single [**Who**] element is the Testbed, which itself is a block with its own parts and behaviors (and diagrams, etc.). The [**What**] attribute is linked to the Transition to Safe Mode activity that describes what needs to happen to be in a safe state (in this case, it primarily involves disconnecting the power supply). If other actions need to occur (e.g., log the safety event), they would be captured in this model element that is directly linked to a requirement that depends on its definition. Next comes [**How Well**] captured with an instance of a Time Limit where the time_limit value is set to 2 s. The Time Limit, which can contain information on how to perform and record the measurement, can be reused as necessary with different limiting values. Finally, the [**Condition**] based on the Enclosure Door Open activity that models how this event occurs and is detected by the system.

As MBSRs fit within SysML modeling concepts, we also have other types of content that we can use to document each requirement fully. We often want to capture how requirements are derived from other parent requirements, and this is visualized in the figure with a «*deriveReqt*» relationship with a rationale comment stating "More specific for enclosure door open event". Rationales, general comments, and documentation can be attached to elements as needed. For each requirement, we typically want a test case to verify if the requirement is true; here, we have a relationship to Safety Tests. Once the tests have been run, we can capture test results with custom «*Artifact*» classes (see 2023-07-01 Safety Test Result Report), either directly in the model or with a link to the location of the report


Legend
🟦 ↗ Condition
🟨 ↗ How Well
🟩 ↗ What
🟥 ↗ Who

Requirements		L4.1.17.MBSR Door Open Shutdown	L4.1.18.MBSR Safety Signal Shutdown	L4.2.24.MBSR Gearboxes	L4.4.4.MBSR Motor Open Circuit	L4.4.9.MBSR Motor Sustained High Load Th	L4.6.1.MBSR Power Converter Max Weight	L4.7.3.MBSR Microcontroller Digital Inputs
Testbed		4	4	4	5	6	4	4
Behavior		1	1	1		1	1	1
Enclosure Door Open	1	🟦↙						
Four Lines Running	1					🟦↙		
Normal Testbed Conditions	2						🟦↙	🟦↙
Provide Torque	1			🟩↙				
Safety Signal Sent	1		🟦↙					
Constraints		1	1	2	3	3	1	1
1 A : Current Demand	1				🟦↙			
4 Nm High Torque : Load Motor	1					🟦↙		
61 Digital : Port Limit	1							🟨↙
75 C : Motor Temp Limit	1					🟨↙		
1800 rpm Rated Speed : Load Motor	1					🟦↙		
Gearbox 15 Nm : Desired Torque	1			🟨↙				
Measurement 1 s : Time Limit	1				🟨↙			
Range +/- 250 mA : Zero Current Deviation	1				🟨↙			
Shutdown 2 s : Time Limit	2	🟨↙	🟨↙					
Speed 2000 rpm : Gearbox	1			🟦↙				
Weight 1300 lbs : Weight Limit	1						🟨↙	
Structure		2	2	1	2	2	2	2
Gearbox	1			🟥↙				
Load Motor	2				🟥↙	🟥↙		
Open Circuit Current Response	1				🟩↙			
Temperature : celsiusTemperature	1					🟩↙		
Microcontroller	1							🟥↙
Digitial Port [0..*]	1							🟩↙
Power Converter	1						🟥↙	
Weight : mass[pound]	1						🟩↙	
Testbed	2	🟥↙	🟥↙					
Transition to Safe Mode	2	🟩↙	🟩↙					

Figure 3. Dependency matrix between select MBSRs and model elements

a server. This artifact is then linked to both the test case itself (as it is a result of it being run) and the requirement Door Open Shutdown to document its verification.

3.2 Summarizing MBSRs in the Model

Requirement diagrams like Fig. 2 are great for completely visualizing one to several requirements, but MBSE tools support various visualizations of the model, including the traditional requirements table. Table 1 shows twelve requirements with their MBSR attributes. The specific cell values are populated with references to model elements, and a user can readily select elements from a "Select, search for, or create elements" GUI from within CSM. Additionally, the MBSR attributes support more granular queries of the requirements/model. For example, an engineer could filter the table to only list the three requirements related to Microcontroller.

With careful attention to Table 1, one will observe that many elements are reused to define the MBSRs, which is common in system development, but often not formally enforced. One of the benefits of MBSRs and model-based approaches is

Table 1. Requirements table with original requirements text, MBSR attributes, and SysML verify by relationships

		Original Requirement Text	Model-based Structured Requirement (MBSR) Attributes				
#	Name	Text	△ [1] Who	[2] What	[3] How Well	[4] Condition	Verified By
1	Induction Motors	All four (4) testbed lines shall include one induction motor.	Actuation Line	Test Motor	Induction Motor	Normal Testbed Conditions	Parts Inspection
2	Starting Capacitors	The output of the power supply shall connect to six 24 mF and two 16 mF capacitors rated for 350 V in parallel to provide the requisite power to start up the induction machines.	Capacitor Bank	Capacitance : capacita...	Capacitor 1 24 mF : Capacitor Desiren Capacitor 2 24 mF : Capacitor Desiren Capacitor 3 16 mF : Capacitor Desiren Capacitor 4 16 mF : Capacitor Desiren Capacitor 5 16 mF : Capacitor Desiren Capacitor 6 16 mF : Capacitor Desiren	Normal Testbed Conditions	Induction Machine Startup Parts Inspection
3	Gearboxes	The gearbox on each actuation line shall be rated to 15 Nm at 2000 rpm.	Gearbox	Provide Torque	Gearbox 15 Nm : Desired Torque	Speed 2000 rpm : Gearbox	Fixed Speed Motor Test
4	Motor Sustained High Load Thermal Test	With all four lines running simultaneously, all BLDC motors shall remain below 75 degC when operated at rated speed of 1800 RPM at high torque of 4 Nm.	Load Motor	Temperature : celsius...	75 C : Motor Temp Limit	Four Lines Running 4 Nm High Torque : Load Mc 1800 rpm Rated Speed : Loa	Motor Temperature Test
5	Motor Open Circuit	The rms value of motor coil current shall be greater than 250 mA when the rms value of coil current demand is greater than 1 A to simulate an open circuit in the motor winding.	Load Motor	Open Circuit Current R...	Range +/- 250 mA : Zero Current Dev Measurement 1 s : Time Limit	1 A : Current Demand	Motor Open Circuit Test
6	Inrush Inductance	The microcontroller shall ramp up the voltage from the power supply to the testbed via a slew rate of 30 V/s to charge the starting capacitors.	Microcontroller	Ramp Up Voltage	Startup 30 V/s : Slew Rate Desiremen	Startup	Induction Machine Startup
7	Microcontroller Digital Inputs	The microcontroller shall have digital (61) input ports.	Microcontroller	Digitial Port [0..*]	61 Digital : Port Limit	Normal Testbed Conditions	Parts Inspection
8	Microcontroller Outputs	The microcontroller shall broadcast system status, error conditions, and RMS system parameters over UART every 1 second using 10 digital PWM outputs.	Microcontroller	Broadcast Messages	1 per sec : Message Rate Desirement Microcontroller : Message Payloads	Normal Testbed Conditions Working	Microcontroller Verification
9	Power Converter Max Weight	The power converter assembly shall not exceed the weight limit of 1300 lbs.	Power Converter	Weight : mass[pound]	Weight 1300 lbs : Weight Limit	Normal Testbed Conditions	Weight Summation Analysis
10	Power Converter Output	The power converter shall deliver 10kW power in total to the four (4) actuation lines using an AC supply.	Power Converter	Provide Power	10 kW : Power Desirement	Normal Testbed Conditions Working	Induction Machine Startup Fixed Speed Motor Test
11	Door Open Shutdown	The testbed shall transition to safe mode within 2 sec when the enclosure door is opened.	Testbed	Transition to Safe Mode	Shutdown 2 s : Time Limit	Enclosure Door Open	Safety Tests
12	Safety Signal Shutdown	The testbed shall transition to safe mode within 2 sec when the safety signal is transmitted.	Testbed	Transition to Safe Mode	Shutdown 2 s : Time Limit	Safety Signal Sent	Safety Tests

that there is a *single definition* of a test case, physical component, state, etc., formally linked throughout the model. This single definition, multiple uses, is automatically visualized in Fig. 3.

3.3 Discussion Points

This section considers several discussion points relating to the usage of MBSR and model-based approaches.

Point 1: Complete yet Focused Requirement Statements. It is still too common to try to include as much as possible in the text statement, with the justification being that this information is required to understand the requirement. However, the rationale for the requirement and other contextual factors are optional for the legal obligation to be defined. It has been observed that MBSRs help focus the requirement definition on the attributes while supporting the full contextual description with details in those attribute elements and SysML constructs like rationales.

Point 2: Reusability. As previously mentioned, the reuse of authoritative, single-source-of-truth model elements (e.g., blocks, test activities, etc.) throughout the definition of the requirements can bring many benefits. For example, a simple name change in this approach to an element such as Load Motor to BLDC Motor would only require it to be updated in one location, and the change propagates automatically everywhere it is needed. We also often reuse or derive requirements, test plans, conditions, etc., from our previous efforts. The use of general elements with placeholders immediately supports reuse. For example, the actual temperature limit value used in the requirement might be pulled from the specific element with the temperature limit placed on it (which can change depending on the selected component, a design decision). Model-based approaches directly support this type of reuse, potentially reducing development costs, time, and risk.

Point 3: Consistent Visualizations. As a graphical language, SysML provides an opportunity to standardize and reduce work in creating visualizations. Changes to an element in one diagram are automatically reflected in all other diagrams.

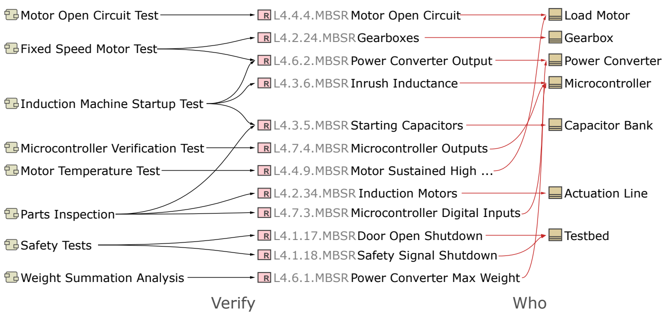

Motor Open Circuit Test
Fixed Speed Motor Test
Induction Machine Startup Test
Microcontroller Verification Test
Motor Temperature Test
Parts Inspection
Safety Tests
Weight Summation Analysis
L4.4.4.MBSR Motor Open Circuit
L4.2.24.MBSR Gearboxes
L4.6.2.MBSR Power Converter Output
L4.3.6.MBSR Inrush Inductance
L4.3.5.MBSR Starting Capacitors
L4.7.4.MBSR Microcontroller Outputs
L4.4.9.MBSR Motor Sustained High ...
L4.2.34.MBSR Induction Motors
L4.7.3.MBSR Microcontroller Digital Inputs
L4.1.17.MBSR Door Open Shutdown
L4.1.18.MBSR Safety Signal Shutdown
L4.6.1.MBSR Power Converter Max Weight
Load Motor
Gearbox
Power Converter
Microcontroller
Capacitor Bank
Actuation Line
Testbed
Verify
Who

Figure 4. Relation map between the test cases, their related requirements, and [**Who**] attributes

Visual styles can be made consistent throughout all flow charts (as activity diagrams) and wiring diagrams (as internal block diagrams) in the model. Certain types of diagrams can be automatically generated (e.g., Figs. 3 and 4) with the engineer moving more towards posing queries on the system model to create the desired visualization rather than manually creating (a potentially incomplete or incorrect) version.

Point 4: Design for Testability (DfT). In DfT approaches, testability issues, such as gaps between design decisions and practical testing of those choices, are identified much early in the system development process (Grout, 2006). System development is an incremental process, but siloing work restricts potential pathways for interaction and more efficient identification of errors. Here, testing and V&V concerns and decisions can be brought earlier into the development through a system model that captures "all" aspects of system development. Requirements might only be considered complete if the MBSR attributes and test cases linked to the requirement are approved by testing stakeholders.

Point 5: Automated Test Plans and Other Documents. We will often need to communicate aspects of the system development to various personnel (e.g., test technicians) while still following organizational best practices. Many MBSE tools support the generation of various artifacts from the model (e.g., tables, matrices, and maps but also full-featured documents and presentations). For a basic illustration of this concept, see Fig. 4. This relation map summarizes the test cases, what requirements are being verified by them individually, and what parts of the system are specifically of interest (i.e., [**Who**]). Tailored approaches could meet organizational guidelines for documents like test plans.

Point 6: Digital Threads and Completeness. Creating documentation and keeping track of changes and decisions throughout system development can be challenging and may be seen by some as costly with limited realized benefits. However, when it is a natural part of the engineer's activities, its advantages can be realized without many of the potential consequences. Developing a complete system model with the appropriate relationships identified as the system evolves starts to build digital threads between previously disconnected parts of the system understanding. With these relationships, activities like change propagation analysis to understand what *might* be impacted by a change to a particular element are straightforward and automated. For example, if Transition to Safe Mode was updated, an engineer would know that two requirements are impacted and then implicitly that the test case Safety Tests might need to be revisited.

Quality can be measured in many ways, including the completeness of the system model. As discussed in Herber et al., 2022, various completeness and quality checks can be made on the requirements when using MBSRs and supported MBSE tools. One approach to adding some additional rigor would be restricting the MBSR attributes to elements with the «*verified*» stereotype to ensure that no requirement includes poor quality or simply incomplete elements.

IV CONCLUSION

This paper discussed model-based structured requirements (MBSRs) in SysML as a modern, model-based approach for developing high-quality requirements. Illustrated on the ASET Lab testbed, MBSRs can help release complete yet focused requirement statements, reusability, design for testability, digital threads, and more. It is not a secret that even low complexity safety-critical aerospace actuation systems can, in practice, easily cost several man-years at both the aircraft OEM and top tier suppliers to just agree on a requirement development and management plan, communicate requirements, evaluate their quality, and compute compliance. This heavy cost and time delay burden causes unproductive tension between program management and systems engineering functions on both sides, causing late discovery of requirements, compliance, and critical product issues. MBSRs have the structure to incrementally automate and reduce this tension today while making our systems engineering machinery ready to better take advantage of digital transformation and AI as they become available.

REFERENCES

Borky, J. M., and Bradley, T. H. (2019). *Effective Model-Based Systems Engineering*. Springer. doi: 10.1007/978-3-319-95669-5

Call, D. R., and Herber, D. R. (2022). Applicability of the diffusion of innovation theory to accelerate model-based systems engineering adoption. *Syst. Eng.*, *25*(6), 574–583. doi: 10.1002/sys.21638

Carroll, E. R., and Malins, R. J. (2016). *Systematic literature review: How is model-based systems engineering justified?* (Tech. rep. SAND2016-2607). Sandia National Laboratories. doi: 10.2172/1561164

Carson, R. S. (2015). Implementing structured requirements to improve requirements quality. *INCOSE International Symposium*, *25*(1), 54–67. doi: 10.1002/j.2334-5837.2015.00048.x

Friedenthal, S., Moore, A., and Steiner, R. (2015). *A Practical Guide to SysML* (3rd). Elsevier. doi: 10.1016/c2013-0-14457-1

Grout, I. A. (2006). *Integrated Circuit Test Engineering*. Springer. doi: 10.1007/1-84628-173-3

Herber, D. R., Narsinghani, J. B., and Eftekhari-Shahroudi, K. (2022). Model-based structured requirements in sysml. *IEEE International Systems Conference*. doi: 10.1109/SysCon53536.2022.9773813

Herber, D. R., Narsinghani, J. B., and Eftekhari-Shahroudi, K. (2023). https://github.com/danielrherber/model-based-structured-requirements

Huldt, T., and Stenius, I. (2018). State-of-practice survey of model-based systems engineering. *Syst. Eng.*, *22*(2), 134–145. doi: 10.1002/sys.21466

IEEE. (2018). *ISO/IEC/IEEE international standard - systems and software engineering – life cycle processes – requirements engineering*. doi: 10.1109/ieeestd.2018.8559686

Madni, A., and Purohit, S. (2019). Economic analysis of model-based systems engineering. *Systems*, *7*(1). doi: 10.3390/systems7010012

No Magic. (2020). Cameo Systems Modeler 19.0 SP4. https://docs.nomagic.com/display/NMDOC/19.0+LTR+SP4+Version+News

OMG. (2019). *OMG SysML version 1.6* (Standard formal/19-11-01). https://www.omg.org/spec/SysML/1.6

Pohl, K. (2010). *Requirements Engineering*. Springer.

Pohl, K., and Rupp, C. (2015). *Requirements Engineering Fundamentals* (2nd). Rocky Nook.

Wheatcraft, L., Katz, T., Ryan, M., and Wolfgang, R. B. (2022). *Needs, requirements, verification, validation lifecycle manual* (Tech. Prod. INCOSE-TP-2021.002-01). INCOSE Requirements Working Group.

ACKNOWLEDGEMENTS

The authors would like to thank 1) the CSU/Woodward MRDA program for the creation of the ASET Lab and continued support of MBSR research, and 2) McKenzy Johnson for her model and diagram contributions. The opinions expressed herein do not necessarily represent the views of Woodward, Inc. or any related organization.