Model-to-Text Requirements Engineering Using SysML Model-Based Structured Requirements Applied to Flight Control System Development

Michael Kellogg
Colorado State University
Department of Systems Engineering
Fort Collins, CO 80523
michael.kellogg@colostate.edu

Daniel R. Herber
Colorado State University
Department of Systems Engineering
Fort Collins, CO 80523
daniel.herber@colostate.edu

Abstract

System engineers engage in requirements engineering to manage the complexity of developing products and processes. Requirements engineering involves three main tasks: *Generate* validated requirement specifications, *Plan* the methods used to verify that the developed system satisfies those requirements, and *Manage* the evidence that the developed system satisfies the requirements. When performing these tasks, Systems Engineers generally use natural language to generate specifications, plans, and tables. Natural language is inherently human-centric and accessible to many stakeholders. However, its inherent flexibility, the inconsistency in systems engineers' writing skills, and variations in how humans interpret it can lead to confusion and errors in natural-language documentation. Model-based systems engineering (MBSE) is an approach that emphasizes the creation of a system model consisting of elements and relationships. MBSE has the potential to address these issues; however, not all stakeholders have access to MBSE tools or have the training to understand them. So, the information in MBSE models must be translated into natural language text in the form of artifacts (e.g., documents) to be accessible to these stakeholders. In this paper, we develop an MBSE approach to requirements engineering that leverages Model-Based Structured Requirements (MBSR) created in Systems Modeling Language (SysML) models for automatic generation of natural-language requirements and more, directly from the model using the Velocity Template Language (VTL). Therefore, this approach facilitates exporting information from the MBSE tool to a word-processing document containing specifications, plans, and tables, thereby eliminating the need to write them explicitly. This approach is demonstrated through a case study of a flight control system and an aileron actuation subsystem. Using this approach, the systems engineer can leverage the benefits of MBSE, maintain the human-centric accessibility of natural language, improve the quality of written documents, and reduce preparation time.

Keywords

Requirements Engineering, Systems Engineering, Model-Based Systems Engineering, Model-Based Structured Requirements, Model-to-Text Translation.

Introduction

Requirements Engineering

Requirements are the foundation for system definition and form the basis for new system development (Walden et al., 2023, Chapter 2.3.5.3). Requirements are statements that translate or express needs,

along with their associated constraints and conditions (ISO29148, 2018, Chapter 4.1.17). During system development, requirements must be defined at each level of the system hierarchy (e.g., system, subsystem, component, etc.) with sufficient detail to be realizable through making, buying, or reusing (Walden et al., 2023, Chapter 2.3.5.3).

Requirements engineering is the subset of systems engineering (SE) concerned with *Generating* a hierarchy of validated requirements, *Planning* the methods used to verify that the developed system satisfies those requirements, and *Managing* evidence that validates that the developed system satisfies the requirements (Hull et al., 2011; ISO29148, 2018).

Textual statements have been an enduring method for capturing requirements, verification methods, and validation evidence. However, the inherent ambiguity that comes from the unstructured use of natural language is a constant challenge. English can be imprecise, and poorly written textual statements can be interpreted by the reader in ways that the author did not intend. This can be a significant concern when statements are the basis of design or contractual obligations (Wheatcraft, Ryan, et al., 2023).

Best practices and guidelines for writing requirements have been developed over decades and continue to be updated as the industry tries to solve this challenge (Hooks and Farry, 2001; Hull et al., 2011; ISO29148, 2018; Wheatcraft, Ryan, et al., 2023). However, the implementation of these best practices by systems engineers to generate precise requirements, verification instructions, and compliance evidence has proven difficult. Engineers practicing requirements engineering may not have received sufficient training in these practices. Institutions may not have procedures in place to ensure adherence to these best practices. In addition, it takes time to evaluate each statement for adherence to the best practices. For a complex system with thousands of requirements, the time commitment to 'inspect in' requirement quality could become a significant portion of the overall development effort. Evolving system solutions can cause subsystem-level requirements to become obsolete, resulting in a need for time-intensive change management efforts to keep the written requirements up-to-date. Thus, resource-limited organizations may be required to settle for requirement engineering efforts that are 'good enough' and accept the program risks that result from poorly constructed or verified requirements.

Model-Based Systems Engineering

Model-based systems engineering (MBSE) is a methodology for system development that emphasizes the use of a system model as the authoritative source of truth for performing SE tasks such as defining, analyzing, and documenting a system (Borky and Bradley, 2019, Chapter 1).The guiding principles for MBSE are that: (1) It includes a cross-discipline model that is the authoritative source of truth and (2) systems engineering documents originate in, or are derived from, that model (Madni et al., 2023, Chapter 1). This contrasts with more traditional document-based approaches, in which documents serve as the source of truth, and models are generated only on an ad hoc basis to validate portions of the documents. With greater emphasis on modeling, the outputs of the SE activities using MBSE methodologies include a coherent model of the system, which enhances understanding of the system (Friedenthal et al., 2014, Chapter 2).

A common modeling language used to support MBSE activities is Systems Modeling Language (SysML) (OMG, 2024). SysML is a graphical language that is used to model systems. The language is built around representing the elements of a system using graphical blocks and representing the relationships between these elements with graphical paths. These blocks and paths are categorized into different stereotypes based on their unique properties. SysML includes the definition of several types of diagrams, which are used to create and communicate the system model. Many MBSE tools have been created to implement and support MBSE using the SysML language.

SysML supports modeling requirements within the system model. It includes a stereotype for requirements and several relationship stereotypes specific to requirements. SysML also defines a requirement diagram, which is used to depict requirements and their relationships to other system elements (OMG, 2024, Chapter 16).

Current approaches for modeling requirements in SysML are not intended to replace a database of textual requirements, but rather to provide a bridge between the text-based requirements and the system model. A database of textual requirements is still often needed in MBSE and may be maintained within a multifunctional MBSE tool or in a separate tool. This capability of SysML is intended to facilitate requirements engineering through robust traceability between the text-based requirements and the model

elements such as designs, behaviors, test cases, etc. (Friedenthal et al., 2014, Chapter 13).

When MBSE is used to support the development of a system-of-systems, the requirement database will include both system-level and subsystem-level requirements. Figure 1 shows how system and subsystem-level requirements relate to the system. With SysML, a system model can be created, and satisfy relationships can be established between the system-level requirements and the SysML model elements. These relationships are used to validate that the model correctly implements the system requirements. The system model then becomes the authoritative source for the subsystems. Whereas a system-level requirement specification is an input to the MBSE process, a subsystem-level definition is an output. This output could take the form of SysML diagrams; however, there will likely be stakeholders who are unfamiliar with SysML. So, it becomes necessary to generate text-based requirement specifications for the subsystems based on the information contained in the SysML system model. Figure 2 shows how SysML modeling fit into the overall system engineering process and how it supports the requirement engineering processes of generating verification plans and subsystem specifications.

As noted in the previous section, writing textual requirements can be time-consuming and error-prone. The methods presented in this paper were developed to improve the process of generating subsystem-level textual documentation from a system model.

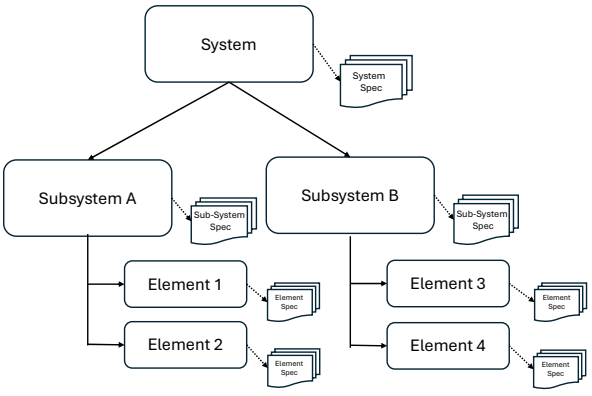


Figure 1. System Structural Hierarchy and Related Specifications

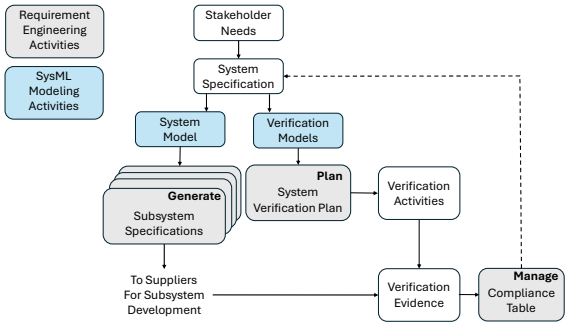


Figure 2. MBSE supported Requirement Engineering Process

Model-Based Structured Requirements

It has been recognized that well-constructed textual requirements follow standard structures (Carson, 2015; ISO29148, 2018). For example, in ISO 29148, the pattern is:

When *[Condition]* **The** *[Subject]* **shall** *[Action] [Object] [Constraint]*

These structural elements can be used to construct several types of requirements, although not all requirement types require all the structural elements to be well-formed. These structural elements have also been described as boilerplates, as the same content is often reused multiple times within a set of requirements (Hull et al., 2011).

Recognizing that requirements follow standard structures with boilerplate content, many researchers are developing techniques that use large language models to extract boilerplates and parts of speech from unstructured text-based requirements and present them in machine-readable forms that can be incorporated into an MBSE database (Riesener et al., 2021; Tikayat Ray et al., 2023, 2024).

Some limited efforts have also been made to develop dynamic linking of numerical values contained within a model to text contained within a document. The numerical values form boilerplate content for the requirements, and the links keep the values aligned with the model every time the link is refreshed (Ballard et al., 2020). However, a systematic review of the literature revealed that most research in the field of information transfer between models and text has focused

on converting text into model elements, whereas much less attention has been paid to converting models into text (Santos et al., 2025).

To support efforts to distinguish the structural elements of a requirement in an MBSE environment, Herber et al. introduced a SysML customization termed a Model-Based Structured Requirement (MBSR). This language customization creates a new requirement stereotype which augments the base SysML «Requirement» stereotype with attributes from the structured requirement format (Herber et al., 2022; Wheaton and Herber, 2024). These customized attributes are then populated by elements of the SysML model, such as blocks and relationships. Populating these attributes is a means of validating that the model has been well constructed and satisfies the system-level requirements (Herber and Eftekhari-Shahroudi, 2023).

This approach suggests that a well-formed SysML model will contain within it the system's boilerplate values, implying that the SysML model itself can be used as a boilerplate database. From that boilerplate database, a set of well-formed structured requirements can be generated (Hull et al., 2011).

In system development, the system requirements are input, driving the system model creation, so having a boilerplate database of the system will not aid in creating system requirements. However, in a hierarchical system, there may be many sub-systems contained within the parent system. A SysML model generated for the parent system, with satisfy relationships to the parent system requirements, could be used to support the creation of requirements for the sub-systems. Using the boilerplate information contained within the model and the structured requirement stereotype, model elements can be used to auto-generate textual requirements for these subsystems. Significant savings in requirement generation and validation effort could be achieved by doing so. The generated requirements would follow a well-formed structure that would not be dependent upon the systems engineer's writing skills. The textual requirements would be aligned with the SysML model as the single source of truth, so updates to the SysML model would automatically propagate to the textual requirements.

The focus of this paper is to present the methods we have developed that leverage MBSE and MBSR in the requirements engineering process. In the methods section, we present the techniques developed, and in the results section, we present a case study of those techniques being applied to the development of a notional flight control system and an aileron actuator subsystem.

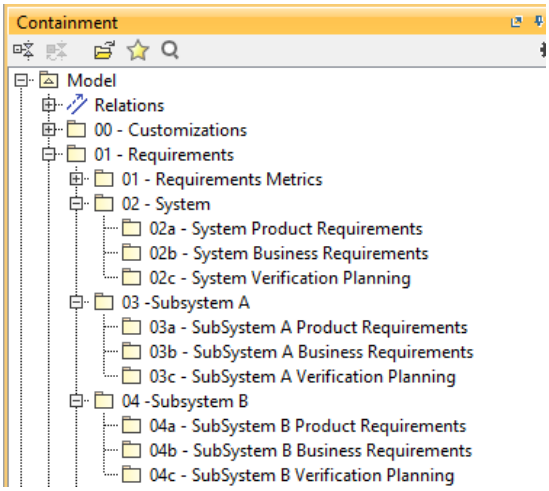


Figure 3. SysML Model Containment Tree Focusing on Requirements Content

Methods

Requirement Organization

To develop this process for building requirements from the SysML model, we used Dassault Systèmes CATIA Magic Systems of Systems Architect (MSOSA) as our MBSE tool for generating SysML Models and Microsoft Word as our word-processing application. Within the MBSE tool, we organized the requirements into a specification tree that mirrored the system's hierarchical structure as suggested in (Friedenthal et al., 2014, Chapter 13.8). In addition to separating the requirement packages by system hierarchy level, we created, at each level, separate packages for product requirements, business requirements, and verification plans, as shown in Figure 3.

As a baseline for this effort, we started with the «Structured Requirement» stereotype, created in the publicly available MBSR profile (Wheaton and Herber, 2024). Next, this stereotype was extended to include the following five new custom requirement stereotypes:

- «RqmtProduct» – used to identify product type requirements
- «RqmtWork» – used to identify business type requirements
- «RqmtVerf» – used to identify verification information
- «RqmtComment» – used to identify clarifying

comments

- «RqmtHeading» – used to identify section headings

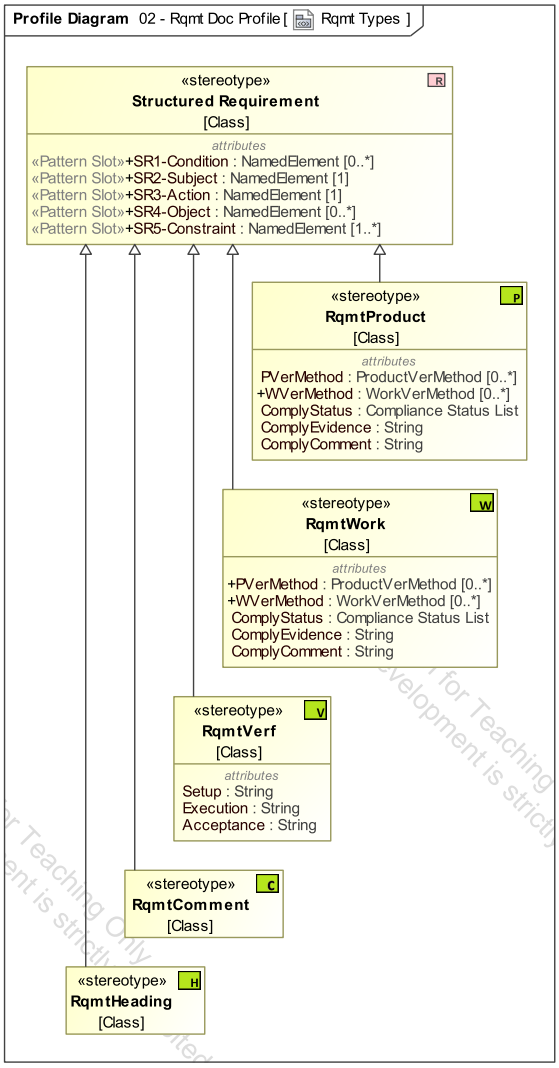


Figure 4. Custom Requirement Stereotypes with Attributes to Support Automated Document Generation

The primary purpose of these new stereotypes is to organize the requirements for exporting to the word-processing application. However, additional attributes were added to some stereotypes to support verification planning and the compilation of compliance evidence. These custom requirement stereotypes with their additional attributes are shown in Figure 4.

Three additional comply attributes, ComplyStatus, ComplyEvidence, and ComplyComment, were added to the «RqmtProduct» and «RqmtWork» stereotypes to aid in compiling compliance information. The ComplyStatus attribute uses the Compliance Status List value type defined in the MBSR profile. This value type has an enumerated list of Expected Compliance, Compliant, NonCompliant, and TBR (To Be Refined). The ComplyComment and ComplyEvidence are string-type attributes to capture any clarifying comments regarding the compliance status and references to evidence of compliance, which may be generated outside of the MBSE tool.

Two additional verification method attributes, PVerMethod and WVerMethod, were added to the «RqmtProduct» and «RqmtWork» stereotypes to aid in the creation of requirement verification tables. The «Extended Requirement» stereotype, upon which the «Structured Requirement» is an extension, has an attribute called verifyMethod to define verification methods. This attribute has enumerated options of Inspection, Demonstration, Analysis, and Test. PVerMethod allows further refinement to the test verification method to include Acceptance and Qualification types of testing. WVerMethod allows further refinement of the inspection verification method to include Reviews, Data items, and Hardware types of inspections.

The «RqmtVerf» requirement stereotype was created to refine the test cases used to verify the requirements. The information captured in the «RqmtVerf» element is derived from the requirement. To help track this relationship between the requirement and the verification planning, an extension of the relationship stereotype «DeriveReqt» was created called «VerifPlan» as shown in Figure 5.

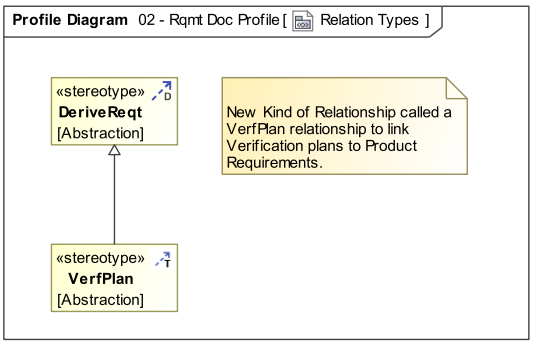


Figure 5. Custom Relationship Types

Verification plans define how, where, and by whom, each requirement will be verified. Additionally they define the pass / fail criteria by which acceptance will be determined (ISO29148, 2018, chapter 6.4). Verification planning information is captured in the «RqmtVerf» element and divided into three string-type attributes: Setup, Execution, and Acceptance. Figure 6 shows the relationship between the requirement, the test case, and the «RqmtVerf» element.

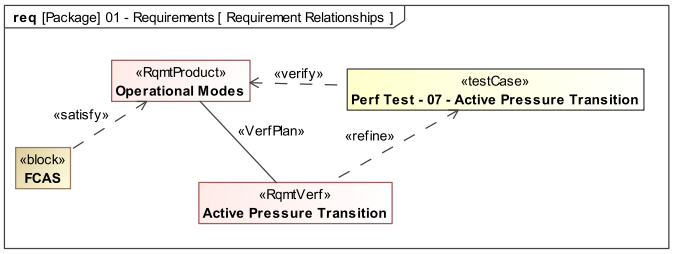


Figure 6. Requirement Verification Relationships

Text Requirement Export

To create text documents from requirement blocks, MSOSA has a built-in report generation tool. This tool relies on a Java-based scripting language, Apache Velocity Engine, to export requirements from the model into the word processing application, Microsoft Word. MSOSA includes templates, which are written in the Velocity Template Language (VTL) and stored as Microsoft Word files. These templates include scripting instructions for exporting the information from the model to a Microsoft Word file. More information on this export tool can be found in the MSOSA user guide (No Magic, 2024) and in the Apache Velocity user guide (The Apache Software Foundation, 2020).

Using the built-in templates will generate a generic Requirement Report. However, customization of those templates is necessary to remove unwanted requirement information, to add desired information, and to format the export as desired. Figure 7 shows the portion of our customized script that provides instructions on how to export information from the new requirement stereotypes. Using the proposed custom requirement elements with the stereotype of «RqmtHeading», the report generator will only export the requirement's name and will format it as a header in the exported document. Requirement elements with the stereotype of «RqmtComment» will only export the requirement's text and will format it with a 1-inch indent. Requirement elements with the stereotype of «RqmtProduct» or «RqmtWork» will export the requirement's identification number and the requirement's text, with the identification number left justified and the text indented 1 inch. Requirement elements with the stereotype of «RqmtVerf» will export the requirement's name as a header, the identification number left justified, and the text of the attributes Setup, Execution, and Acceptance.

An additional component of a textual requirement specification is a requirement verification table, which provides guidance on the acceptable means for verify-

```
## Text generation code
## requirement heading section
#if ($object.elementType == "rqmtheading")
        #writeHeader($object.Name)
#end
## requirement comment section
#if ($object.elementType == "rqmtcomment")
                $object.text
#end
## Structured Requirement output
#if ($object.elementType == "rqmtproduct")
$object.id      $object.text
#end
## Structured Work Requirement output
#if ($object.elementType == "rqmtwork")
$object.id      $object.text
#end
## Structured Verif Requirement output
#if ($object.elementType == "rqmtverf")
#writeHeader($object.Name)
$object.id      Setup
                $object.Setup
                Execution Instructions
                $object.Execution
                Acceptance Criteria
                $object.Acceptance
#end
```

Figure 7. Part of the Requirement Report Template Written in Velocity Template Language (VTL) for Generating a Requirement Text

Table 1: Requirement Verification Matrix

ID	Name	Verify Method
#forrow($r in $sorter.humanSort($ VerifiedRequirement, 'id'))$r.id	$r.name	#if ($r.wVerMethod.size() > 0) #foreach ($from in $sorter.sort($r.wVerMethod, "Id")) $from.name #end #end #if ($r.pVerMethod.size() > 0) #foreach ($from in $sorter.sort($r.pVerMethod, "Id")) $from.name #end #end #endrow

Figure 8. Part of the Requirement Report Template Written in Velocity Template Language (VTL) for Generating a Requirement Verification Table

```
$object.text
#end
## Structured Requirement output
#if ($object.elementType == "rqmtproduct")
$object.id      #if ($object.SR1-Condition.size() > 0)#foreach ($from in $sorter.sort($object.SR1-
                Condition, "Id"))When in $from.name the#end #else The #end$object.SR2-
                Subject.name shall $object.SR3-Action.name#foreach ($from in
                $sorter.sort($object.SR4-Object, "Id")) $from.name#end#foreach ($from in
                $sorter.sort($object.SR5-Constraint, "Id"))$from.text#end.
#end
## Structured Work Requirement output
#if ($object.elementType == "rqmtwork")
$object.id      The $object.SR2-Subject.name shall $object.SR3-Action.name #foreach ($from in
                $sorter.sort($object.SR4-Object, "Id"))$from.name#end#foreach ($from in
                $sorter.sort($object.SR5-Constraint, "Id")) $from.text#end.
```

Figure 9. Part of the Requirement Report Template Written in Velocity Template Language (VTL) for Generating a Text Statement from a Model-Based Structured Requirement with colors

```
## Structured Requirement output
#if ($object.elementType == "rqmtproduct")
$object.id      #if ($object.SR1-Condition.size() > 0) #foreach ($from in $sorter.sort($object.SR1-
                Condition, "Id"))(When in $from.name) the#end #else The #end [$object.SR2-
                Subject.name] shall *$object.SR3-Action.name* #foreach ($from in
                $sorter.sort($object.SR4-Object, "Id")) <$from.name> #end #foreach ($from in
                $sorter.sort($object.SR5-Constraint, "Id"))+$from.text+ #end.
#end
## Structured Work Requirement output
#if ($object.elementType == "rqmtwork")
$object.id      The [$object.SR2-Subject.name] shall *$object.SR3-Action.name* #foreach ($from in
                $sorter.sort($object.SR4-Object, "Id"))<$from.name> +#end#foreach ($from in
                $sorter.sort($object.SR5-Constraint, "Id"))$from.text#end+.
#end
```

Figure 10. Part of the Requirement Report Template Written in Velocity Template Language (VTL) for Generating a Text Statement from a Model-Based Structured Requirement with symbols

ing the requirements. Figure 8 shows the portion of our customized script that provides instructions on how to export a requirement verification table from the information stored for each requirement in the PVerMethod and WVerMethod attributes.

Structure Requirement Export

When an MBSR approach is used for creating subsystem requirements within the MBSE environment, the requirements rely on the five MBSR attributes assigned to the «Structured Requirement» stereotype and inherited by the «RqmtProduct» and «RqmtWork» stereotypes. These five attributes: SR1-Condition, SR2-Subject, SR3-Action, SR4-Object, and SR5-Constraint correspond to the five requirement structural elements identified in ISO 29148 (Wheaton and Herber, 2024). When using an MBSR approach to requirement generation, these attributes are defined, not through written text, but through links to model objects.

The scripting instructions in the report generating export templates can be written to build natural language requirements from the information stored in these attributes. Figures 9 and 10 show the portion of our customized script that provides instructions on how to generate the natural language requirements from the model.

The code uses two methods to construct the textual requirement based on the requirement's stereotype. SR1-Condition does not apply to business-type requirements, so it is not referenced in the code for requirements of «RqmtWork» stereotype. In the code shown in Figure 9 the different structural elements are color-coded to facilitate their identification in the generated text. Colors were selected to maximize contrast with adjacent colors. Different VTL scripts can be created to generate reports targeted for different audiences. For example, the code shown in Figure 10 has the different structural elements identified by symbols instead of colors to facilitate their identification in generated black and white text.. Additionally, the code is flexible enough to accommodate multiple conditions or constraints in the generated text.

Results

System Description

To verify the methods developed in this paper, a case study was created around the development of a Flight

Control Actuation System (FCAS) and an Aileron Actuation Subsystem (AAS).

Two SAE Aerospace Information Reports (AIR4094, 2016; AIR4253, 2018) were used for guidance in developing the notional FCAS and AAS presented in this case study. In practice, an FCAS would have multiple subsystems, but only the AAS was developed in detail. This case study is sufficiently complex to encompass multiple types of product and business requirements at both the system and subsystem levels. The FCAS for this case study is a significantly simplified actuation system for an aircraft with the following architectural assumptions:

- The development of the FCAS is carried out by an organization with the business elements necessary to support development activities.
- The FCAS is composed of Electro-Hydraulic Servo Actuators.
- Each actuator has two modes of position control: active and damped.
- The FCAS controls the positions of three unique types of flight control surfaces: ailerons, elevators, and rudders.
- The aileron and elevator surfaces each have two attached actuators for redundancy.
- The rudder surface has three attached actuators for redundancy.
- Each surface is operated with one actuator in active mode and the other actuator(s) in damped mode.
- Each actuator has a single channel for electrical and hydraulic components.
- Each actuator interfaces independently with a flight control computer.
- Each actuator interfaces independently with a hydraulic system.

Notional overviews of the FCAS and the AAS are shown in Figure 11 and Figure 13, respectively. SysML Models of the FCAS and AAS were modeled in MSOSA using SysML. The block definition diagrams of their part structures can be seen in Figure 12 and Figure 14.

System Textual Requirements

49 product requirements and 22 business requirements were generated for the FCAS system. This number of requirements was selected to provide a large variety of requirement types, including functional, operational, interface, physical, performance, environmental, design constraints, and business. These requirements were organized by type for ease of access

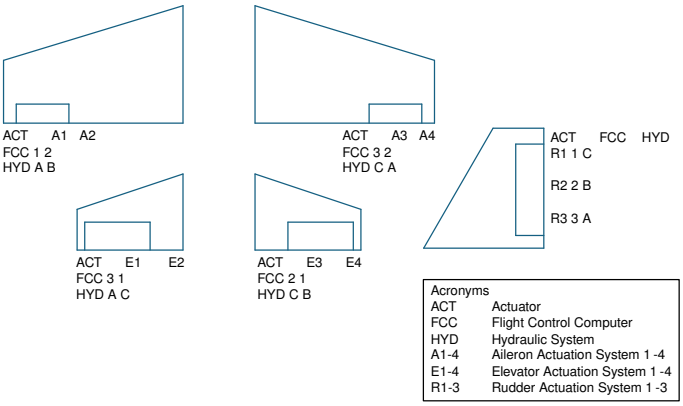


Figure 11. Flight Control Actuation System (FCAS) Overview

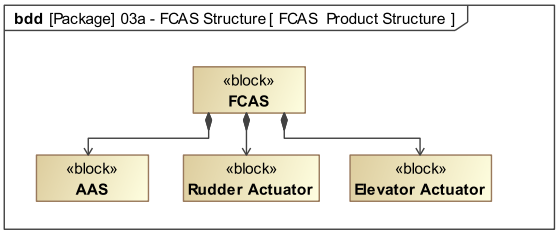


Figure 12. FCAS SysML Structure

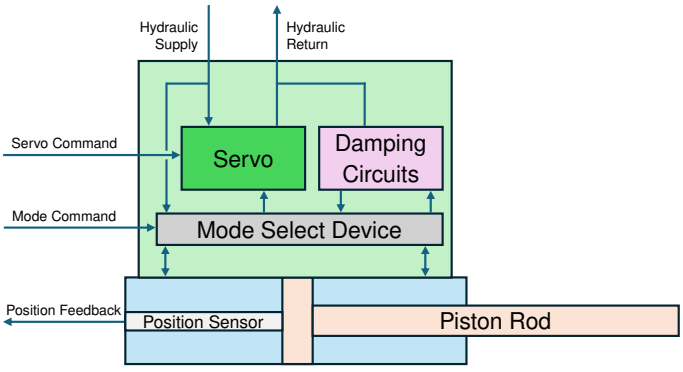


Figure 13. Aileron Actuation Subsystem (AAS) Overview

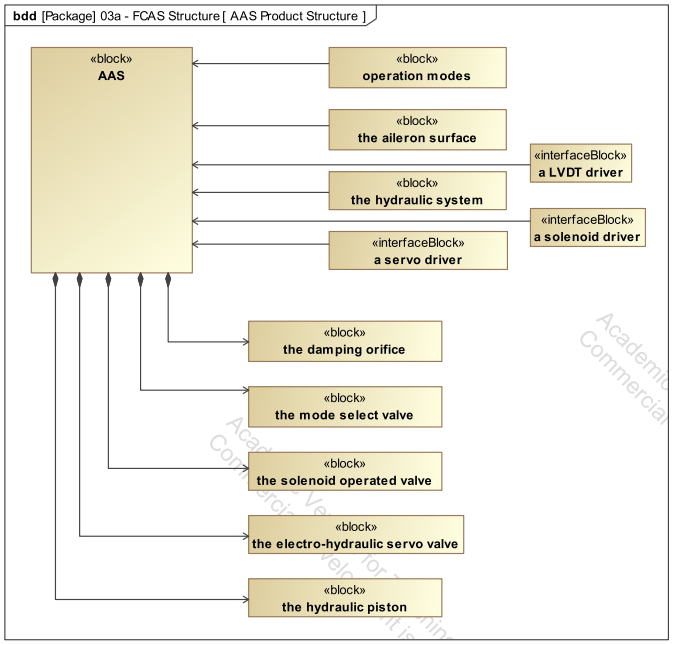


Figure 14. AAS SysML Structure

in the requirement specification. Figure 15 shows how the system requirements look in MSOSA when viewed as a table. Figure 16 shows how the requirements look when exported from MSOSA to a requirement specification using the customized template described in the methods section. The Requirement Specification preserves the organization of the requirement set in the MSOSA table and presents both the identification numbers and the requirement text.

Additionally, using the customized template, the requirement verification table, which identifies the acceptable verification method for each requirement, was generated for the requirement set and included in the exported requirement specification as shown in Figure 17.

In addition to the VTL scripts that provide instructions for exporting data from the model, custom templates can also be populated with static text that may include information relevant to the system but is not dependent on the MSOSA model for its content. The interested reader can find examples of this static text in the MSOSA default report templates (No Magic, 2024).

Subsystem Structure Requirements

The subsystem-level requirements, exemplified by the AAS, are derived through analysis in the model from the system-level requirements.

Rather than representing the subsystem requirements as text per OMG, 2024, Chapter 16. Requirements are

#	Name	Text
1	FCAS-1 Operational Requirements	
2	FCAS-71 Modes and States	
3	FCAS-32 Operational Modes	The FCAS shall operate each flight surface in two modes: Active and Damped.
4	FCAS-72 Physical Architecture	
10	FCAS-73 Functions	
11	FCAS-38 Aileron Position Control	The FCAS shall control the position of the aileron surfaces.
12	FCAS-39 Aileron Mode Control	The FCAS shall control the operating mode of the aileron surface.
13	FCAS-40 Elevator Position Control	The FCAS shall control the position of the elevator surfaces.
14	FCAS-41 Elevator Mode Control	The FCAS shall control the operating mode of the elevator surface.
15	FCAS-42 Rudder Position Control	The FCAS shall control the position of the rudder surface.
16	FCAS-43 Rudder Mode Control	The FCAS shall control the operating mode of the rudder surface.
17	FCAS-2 Interface Requirements	
18	FCAS-10 Mechanical Interface	
22	FCAS-11 Hydraulic Interface	
24	FCAS-12 Electrical Interface	
25	FCAS-48 Position Command Signal	The FCAS shall interface with a 25 mA rate command signals for each actuator.
26	FCAS-49 Mode Command Signal	The FCAS shall interface with a 28 VDC mode excitation signals, confirming with MIL-STD-704 for each actuator.
27	FCAS-50 Position Feedback Excitation	The FCAS shall interface with a five-wire LVDT driver, with the following characteristics, for each actuator.
28	FCAS-52	Excitation Voltage: 10 Vrms @ 3000 Hz.
29	FCAS-51 Position Feedback Output	The FCAS shall produce a five-wire ratio metric LVDT output, with the following characteristics, for each actuator.
30	FCAS-53	Full Scale Output: +/- 0.5 V/V Accuracy: +/- 1% F. S. Sum Voltage: 7.00 Vrms
31	FCAS-3 Physical Requirements	
32	FCAS-13 Weight	The FCAS shall have a dry weight that is less than 485 lbs.
33	FCAS-4 Performance Requirements	
34	FCAS-14 Active Mode Travel	
35	FCAS-54 Aileron Max Travel	When in Active mode, the FCAS shall move the aileron surfaces up to 60 degrees.
36	FCAS-55 Elevator Max Travel	When in Active mode, the FCAS shall move the elevator surfaces up to 50 degrees.
37	FCAS-56 Rudder Max Travel	When in Active mode, the FCAS shall move the rudder surfaces up to 90 degrees.

Figure 15. FCAS Requirements Table in MSOSA

3. Requirements

3.1. Operational Requirements

3.1.1. Modes and States

FCAS-32 The FCAS shall operate each flight surface in two modes: Active and Damped.

3.1.2. Physical Architecture

FCAS-33 The FCAS shall contain eleven servo controlled linear hydraulic actuators.

FCAS-34 The FCAS shall use two actuators to position each aileron surface.

FCAS-35 The FCAS shall use two actuators to position each elevator surface.

FCAS-36 The FCAS shall use three actuators to position each rudder surface.

FCAS-37 The FCAS shall use actuators which have two modes of operation: Energized and De-energized.

3.1.3. Functions

FCAS-38 The FCAS shall control the position of the aileron surfaces.

FCAS-39 The FCAS shall control the operating mode of the aileron surface.

FCAS-40 The FCAS shall control the position of the elevator surfaces.

FCAS-41 The FCAS shall control the operating mode of the elevator surface.

FCAS-42 The FCAS shall control the position of the rudder surface.

FCAS-43 The FCAS shall control the operating mode of the rudder surface.

3.2. Interface Requirements

3.2.1. Mechanical Interface

Figure 16. Part of the Exported FCAS Product Requirement Specification in a Word Document

4. Verification

Table 1: Requirement Verification Matrix

ID	Name	Verify Method
FCAS-13	Weight	Acceptance Test
FCAS-18	Low Pressure	Qualification Test
FCAS-19	High Temperature	Qualification Test
FCAS-20	Low Temperature	Qualification Test
FCAS-21	Temperature Shock	Qualification Test
FCAS-22	Humidity	Qualification Test
FCAS-23	Salt Fog	Qualification Test
FCAS-24	Vibration	Qualification Test
FCAS-25	Mechanical Shock	Qualification Test
FCAS-26	Banned Materials	Inspection
FCAS-28	Useful Life	Qualification Test
FCAS-29	MTBF	Analysis
FCAS-30	Storage Life	Analysis
FCAS-32	Operational Modes	Qualification Test
FCAS-33	Servo Controlled Actuators	Inspection
FCAS-34	Aileron Actuators	Inspection
FCAS-35	Elevator Actuators	Inspection
FCAS-36	Rudder Actuators	Inspection
FCAS-37	Actuator Modes	Inspection
FCAS-38	Aileron Position Control	Qualification Test
FCAS-39	Aileron Mode Control	Qualification Test
FCAS-40	Elevator Position Control	Qualification Test
FCAS-41	Elevator Mode Control	Qualification Test
FCAS-42	Rudder Position Control	Qualification Test
FCAS-43	Rudder Mode Control	Qualification Test
FCAS-44	Aileron Mechanical Interface	Inspection
FCAS-45	Elevator Mechanical Interface	Inspection
FCAS-46	Rudder Mechanical Interface	Inspection
FCAS-47	Hydraulic systems	Qualification Test
FCAS-48	Position Command Signal	Inspection

ID	Name	Verify Method
FCAS-49	Mode Command Signal	Inspection Qualification Test
FCAS-50	Position Feedback Excitation	Inspection Acceptance Test
FCAS-51	Position Feedback Output	Inspection Acceptance Test
FCAS-54	Aileron Max Travel	Acceptance Test
FCAS-55	Elevator Max Travel	Acceptance Test
FCAS-56	Rudder Max Travel	Acceptance Test
FCAS-57	Active Mode Operation	Qualification Test
FCAS-58	Aileron Active Operating Rate	Qualification Test
FCAS-59	Elevator Active Operating Rate	Qualification Test
FCAS-60	Rudder Active Operating Rate	Qualification Test
FCAS-61	Damped Mode Operation	Acceptance Test
FCAS-62	Aileron Damped Operation	Qualification Test
FCAS-63	Elevator Damped Operation	Acceptance Test
FCAS-64	Rudder Damped Operation	Qualification Test
FCAS-65	Active Transition Timing	Qualification Test
FCAS-66	Damped Transition Timing	Qualification Test
FCAS-67	Active Hold in voltages	Qualification Test
FCAS-68	Active Pressure Transition	Qualification Test
FCAS-69	Damped Pressure Transition	Qualification Test

Figure 17. Exported FCAS Product Requirement Verification Table (Pages 4 and 5 of the Generated Word Document)

represented using structural elements according to the MBSR profile (Herber et al., 2022; Wheaton and Herber, 2024).

The structured requirement blocks have 5 custom attributes that align with the 5 elements of a structured requirement:

- S1 = Condition
- S2 = Subject
- S3 = Action
- S4 = Object
- S5 = Constraint of Action

Each of these attributes of the requirement is populated by the name of the element of the model. The types of elements associated with each of the attributes are as follows:

- S1 (Condition) = State
- S2 (Subject) = Block
- S3 (Action) = Activity (Associated with the subject block)
- S4 (Object) = State or block
- S5 (Constraint) = Constraint block

Within the SysML tool, the subsystem does not have, and does not need, written textual requirements. The SysML model itself forms the content of the subsystem requirement. Those familiar with SysML will be able to understand the subsystem requirements by reviewing the model, and text requirements can be exported from the model using the export tool for those stakeholders who are unfamiliar with or do not have access to the SysML model.

31 product requirements and 22 business requirements were generated for the AAS subsystem. Like the FCAS, this number of requirements was selected to provide a variety of requirement types, including functional, operational, interface, physical, performance, environmental, design constraints, and business. These requirements were organized by type for ease of access in the requirement specification. Figure 18 shows how requirements look in the MSOSA table view with relationship links to model elements for the structured requirement attributes.

Figure 19 and Figure 20 show how the requirements look when exported from MSOSA to a requirement specification using the customized template described in the methods section. The Requirement Specification preserves the organization of the requirement set found in the MSOSA table and presents both the identification numbers and the generated text of the requirement. In the exports, the structural elements of the requirements are displayed with either different colors or delineated with symbols to highlight the requirement structure.

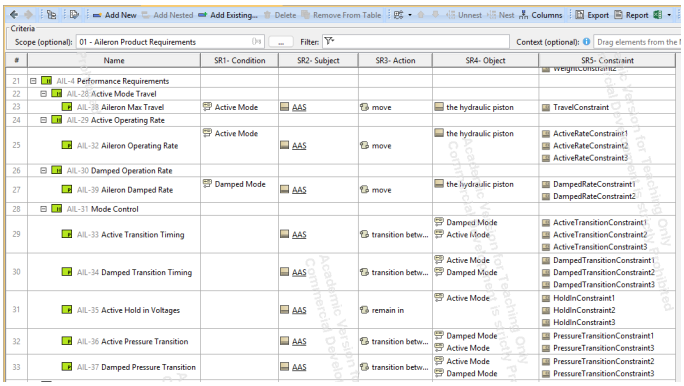
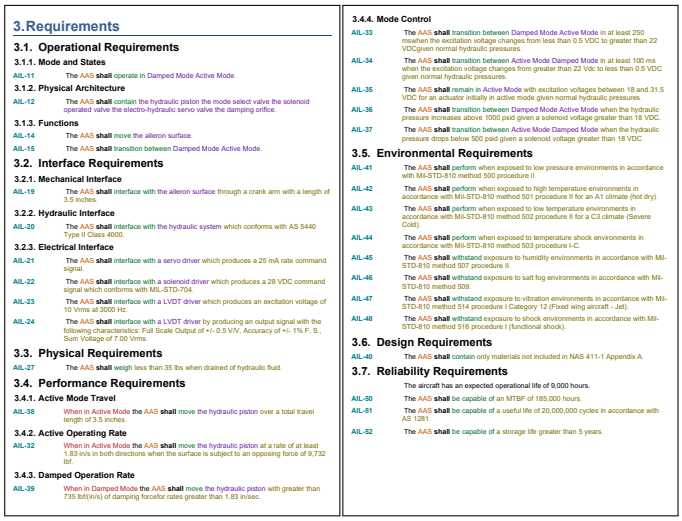


Figure 18. AAS Modeled Product Requirements Table in MSOSA Focusing on Performance Requirements


3.Requirements

3.1. Operational Requirements

3.1.1. Mode and States

AIL-11 The AAS **shall** operate in Damped Mode Active Mode.

3.1.2. Physical Architecture

AIL-12 The AAS **shall** contain the hydraulic piston the mode select valve the solenoid operated valve the electro-hydraulic servo valve the damping orifice.

3.1.3. Functions

AIL-14 The AAS **shall** move the aileron surface.

AIL-15 The AAS **shall** transition between Damped Mode Active Mode.

3.2. Interface Requirements

3.2.1. Mechanical Interface

AIL-19 The AAS **shall** interface with the aileron surface through a crank arm with a length of 3.5 inches.

3.2.2. Hydraulic Interface

AIL-20 The AAS **shall** interface with the hydraulic system which conforms with AS 5440 Type II Class 4000.

3.2.3. Electrical Interface

AIL-21 The AAS **shall** interface with a servo driver which produces a 25 mA rate command signal.

AIL-22 The AAS **shall** interface with a solenoid driver which produces a 28 VDC command signal which conforms with MIL-STD-704.

AIL-23 The AAS **shall** interface with a LVDT driver which produces an excitation voltage of 10 Vrms at 3000 Hz.

AIL-24 The AAS **shall** interface with a LVDT driver by producing an output signal with the following characteristics: Full Scale Output of +/- 0.5 V/V, Accuracy of +/- 1% F. S., Sum Voltage of 7.00 Vrms.

3.3. Physical Requirements

AIL-27 The AAS **shall** weigh less than 35 lbs when drained of hydraulic fluid.

3.4. Performance Requirements

3.4.1. Active Mode Travel

AIL-38 When in Active Mode the AAS **shall** move the hydraulic piston over a total travel length of 3.5 inches.

3.4.2. Active Operating Rate

AIL-32 When in Active Mode the AAS **shall** move the hydraulic piston at a rate of at least 1.83 in/s in both directions when the surface is subject to an opposing force of 9,732 lbf.

3.4.3. Damped Operation Rate

AIL-39 When in Damped Mode the AAS **shall** move the hydraulic piston with greater than 735 lbf/(in/s) of damping forcefor rates greater than 1.83 in/sec.

3.4.4. Mode Control

AIL-33 The AAS **shall** transition between Damped Mode Active Mode in at least 250 msewhen the excitation voltage changes from less than 0.5 VDC to greater than 22 VDCgiven normal hydraulic pressures.

AIL-34 The AAS **shall** transition between Active Mode Damped Mode in at least 100 ms when the excitation voltage changes from greater than 22 Vdc to less than 0.5 VDC given normal hydraulic pressures.

AIL-35 The AAS **shall** remain in Active Mode with excitation voltages between 18 and 31.5 VDC for an actuator initially in active mode given normal hydraulic pressures.

AIL-36 The AAS **shall** transition between Damped Mode Active Mode when the hydraulic pressure increases above 1000 psid given a solenoid voltage greater than 18 VDC.

AIL-37 The AAS **shall** transition between Active Mode Damped Mode when the hydraulic pressure drops below 500 psid given a solenoid voltage greater than 18 VDC.

3.5. Environmental Requirements

AIL-41 The AAS **shall** perform when exposed to low pressure environments in accordance with Mil-STD-810 method 500 procedure II.

AIL-42 The AAS **shall** perform when exposed to high temperature environments in accordance with Mil-STD-810 method 501 procedure II for an A1 climate (hot dry).

AIL-43 The AAS **shall** perform when exposed to low temperature environments in accordance with Mil-STD-810 method 502 procedure II for a C3 climate (Severe Cold).

AIL-44 The AAS **shall** perform when exposed to temperature shock environments in accordance with Mil-STD-810 method 503 procedure I-C.

AIL-45 The AAS **shall** withstand exposure to humidity environments in accordance with Mil-STD-810 method 507 procedure II.

AIL-46 The AAS **shall** withstand exposure to salt fog environments in accordance with Mil-STD-810 method 509.

AIL-47 The AAS **shall** withstand exposure to vibration environments in accordance with Mil-STD-810 method 514 procedure I Category 12 (Fixed wing aircraft – Jet).

AIL-48 The AAS **shall** withstand exposure to shock environments in accordance with Mil-STD-810 method 516 procedure I (functional shock).

3.6. Design Requirements

AIL-40 The AAS **shall** contain only materials not included in NAS 411-1 Appendix A.

3.7. Reliability Requirements

The aircraft has an expected operational life of 9,000 hours.

AIL-50 The AAS **shall** be capable of an MTBF of 185,000 hours.

AIL-51 The AAS **shall** be capable of a useful life of 20,000,000 cycles in accordance with AS 1281.

AIL-52 The AAS **shall** be capable of a storage life greater than 5 years.

Figure 19. Exported AAS Product Requirements With Color Coded Requirement Structure in Word Document

Verification Planning

Detailed instructions are needed for defining how to verify that an as-built product and/or business process satisfies the requirements. These detailed instructions may be called a verification plan. These details go beyond defining a verification method (e.g., Inspection, Analysis, Test), which is often included in the requirement specification. Instead, they take the form of instructions for how, where, by whom, at to what criteria each requirement will be verified. The requirements engineer derives these planning instructions from the requirements and through elicitation activities with other stakeholders in the verification activity (e.g., customer, designers, program managers, test engineers, etc.).

3. Requirements

3.1. Program Management

SOW-38 The [AAS Supplier] **shall** *develop the* <AAS> + as defined in the contract+.

SOW-39 The [AAS Supplier] **shall** *create* <a schedule> + with critical milestones and technical tasks+.

3.1.1. Program Reviews

SOW-42 The [AAS Supplier] **shall** *hold* <a kick off review> + with the Buyer to share the Seller's work plans and integration activities prior to proceeding into preliminary design+.

SOW-43 The [AAS Supplier] **shall** *hold* <a preliminary design review> + with the Buyer to show that the preliminary design is predicted to satisfy the technical requirements prior to proceeding into detailed design+.

SOW-44 The [AAS Supplier] **shall** *hold* <a critical design review> + with the Buyer to show that the detailed design is predicted to satisfy the technical requirements prior to proceeding into fabrication and assembly+.

SOW-45 The [AAS Supplier] **shall** *hold* <a test readiness review> + with the Buyer to show that the test hardware and test preparations are ready prior to proceeding into verification testing+.

SOW-41 The [AAS Supplier] **shall** *manage* <a risk register> + to identify and track development risks and their mitigation activities+.

3.2. Engineering

SOW-46 The [AAS Supplier] **shall** *create* <a compliance matrix> + showing verification of the product to the requirements of the specification+.

SOW-47 The [AAS Supplier] **shall** *create* <drawings> + as required to enable the manufacture of the product+.

3.2.1. Analysis Engineering

SOW-50 The [AAS Supplier] **shall** *analyze* <AAS> + to demonstrate that the product is expected to perform as required+.

SOW-51 The [AAS Supplier] **shall** *analyze* <AAS> + to demonstrate that the product is expected to not be damaged by the thermal environments+.

SOW-52 The [AAS Supplier] **shall** *analyze* <AAS> + to demonstrate the product is expected to satisfy the structural requirements+.

SOW-53 The [AAS Supplier] **shall** *analyze* <AAS> + to demonstate the product is expected to satisfy the reliability requirements+.

3.3. Assembly and Integration

SOW-54 The [AAS Supplier] **shall** *fabricate* <prototype units> + as defined in the contract to be shipped to the Buyer+.

SOW-55 The [AAS Supplier] **shall** *fabricate* <verification units> + as needed to verify the product satisfies the specification+.

SOW-56 The [AAS Supplier] **shall** *fabricate* <production units> + as defined in the contract to be shipped to the Buyer+.

3.4. Verification

SOW-57 The [AAS Supplier] **shall** *verify* <AAS> + satisfies the requirements of the specification using the verification methods defined in the specification+.

Figure 20. First Page of the Exported AAS Business Requirements With Symbol Coded Requirement Structure in Word Document

For this case study, planning information for the FCAS and AAS was captured for each requirement in a «RqmtVerf» stereotyped requirement block as textual information using the setup, execution, and acceptance criteria attributes. The «RqmtVerf» element was linked to the requirement through a «VerifPlan» relationship and linked to a test case through a refines relationship.

The «VerifPlan» elements were organized into a package, similar to the product and business requirements, to form a verification plan. Figure 21 shows an excerpt of the verification plan as seen in the MSOSA table view showing the relationship links to model elements and the planning content.

Figure 22 shows an excerpt of how the plan looks when exported from MSOSA to a verification plan document using the customized template described in the methods section. The plan document preserves the organization of the plan requirement set found in the MSOSA table and presents the identification numbers, name, and content of the setup, execution, and acceptance criteria attributes.

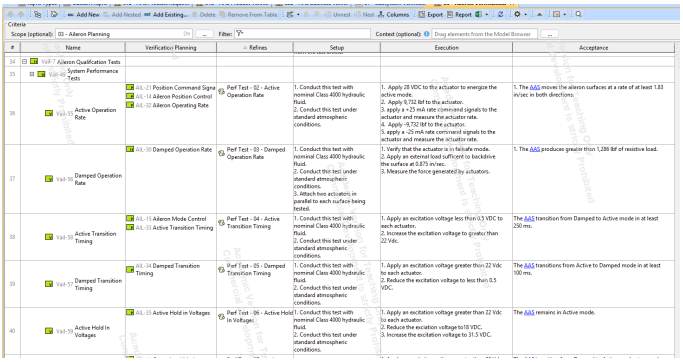


Figure 21. AAS Modeled Verification Planning

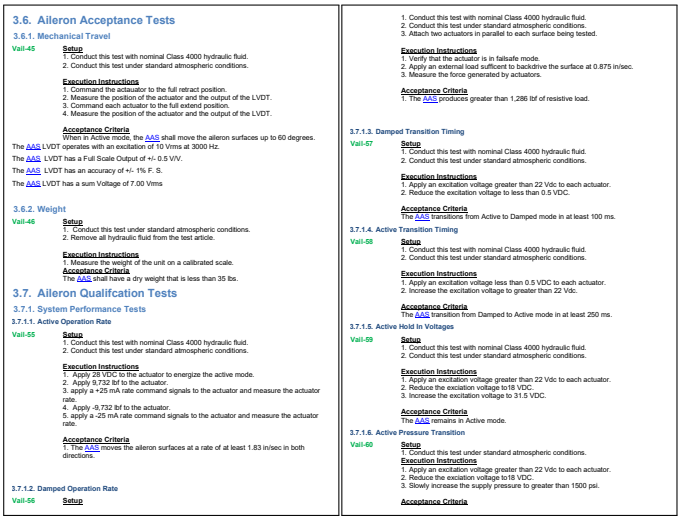
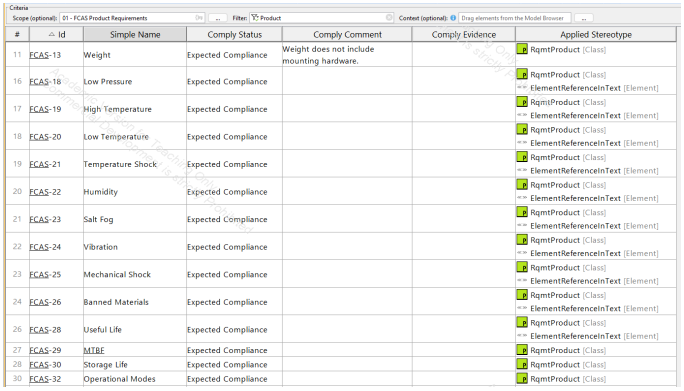


Figure 22. AAS Exported Verification Planning

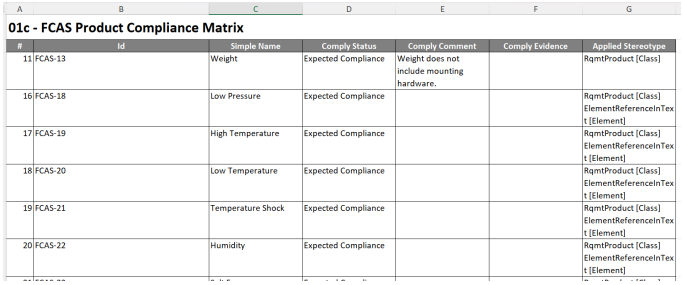
Compliance Evidence

Managing evidence that verifies that the developed system satisfies the requirements is another important requirements engineering task. One method of presenting this information is through a table in which a compliance statement and evidence are provided for each requirement. This type of reporting is well-suited to the MSOSA table view and its export tool. No template customization is required to generate text versions of this information from the model. Figure 23 shows how the table looks in the MSOSA table view with ID, Name, Compliance Status, Compliance Comment, and Compliance evidence shown. Figure 24 shows how the table looks when exported from the MSOSA table view to Microsoft Excel, a text-based spreadsheet. More information on this table export tool is available in the MSOSA user guide (No Magic, 2025).



#	Id	Simple Name	Comply Status	Comply Comment	Comply Evidence	Applied Stereotype
11	FCAS-13	Weight	Expected Compliance	Weight does not include mounting hardware.		RqmtProduct [Class]
16	FCAS-18	Low Pressure	Expected Compliance			RqmtProduct [Class] ElementReferenceInText [Element]
17	FCAS-19	High Temperature	Expected Compliance			RqmtProduct [Class] ElementReferenceInText [Element]
18	FCAS-20	Low Temperature	Expected Compliance			RqmtProduct [Class] ElementReferenceInText [Element]
19	FCAS-21	Temperature Shock	Expected Compliance			RqmtProduct [Class] ElementReferenceInText [Element]
20	FCAS-22	Humidity	Expected Compliance			RqmtProduct [Class] ElementReferenceInText [Element]
21	FCAS-23	Salt Fog	Expected Compliance			RqmtProduct [Class] ElementReferenceInText [Element]
22	FCAS-24	Vibration	Expected Compliance			RqmtProduct [Class] ElementReferenceInText [Element]
23	FCAS-25	Mechanical Shock	Expected Compliance			RqmtProduct [Class] ElementReferenceInText [Element]
24	FCAS-26	Banned Materials	Expected Compliance			RqmtProduct [Class] ElementReferenceInText [Element]
26	FCAS-28	Useful Life	Expected Compliance			RqmtProduct [Class] ElementReferenceInText [Element]
27	FCAS-29	MTBF	Expected Compliance			RqmtProduct [Class]
28	FCAS-30	Storage Life	Expected Compliance			RqmtProduct [Class]
30	FCAS-32	Operational Modes	Expected Compliance			RqmtProduct [Class]

Figure 23. Modeled Compliance Table



01c - FCAS Product Compliance Matrix

#	Id	Simple Name	Comply Status	Comply Comment	Comply Evidence	Applied Stereotype
11	FCAS-13	Weight	Expected Compliance	Weight does not include mounting hardware.		RqmtProduct [Class]
16	FCAS-18	Low Pressure	Expected Compliance			RqmtProduct [Class] ElementReferenceInTex t [Element]
17	FCAS-19	High Temperature	Expected Compliance			RqmtProduct [Class] ElementReferenceInTex t [Element]
18	FCAS-20	Low Temperature	Expected Compliance			RqmtProduct [Class] ElementReferenceInTex t [Element]
19	FCAS-21	Temperature Shock	Expected Compliance			RqmtProduct [Class] ElementReferenceInTex t [Element]
20	FCAS-22	Humidity	Expected Compliance			RqmtProduct [Class] ElementReferenceInTex t [Element]

Figure 24. Exported Compliance Table

Conclusions and Future Work

In this paper, a method based on MBSE principles was developed to perform requirements engineering tasks utilizing a system model through model-based structured requirements (MBSR). A case study was presented to show how these methods can be used to support the requirements engineering associated with the development of a flight control actuation system. The methods supported exporting text requirements, verification plans, and compliance tables from the model into text-based documentation. The methods also supported generating textual subsystem requirements directly from the model, without the need for writing the requirement text.

Some of the benefits of this approach include (1) ensuring alignment of the text documentation with the model, (2) less time spent writing and validating textual requirements, and (3) ensuring sub-system requirements consistently conform to the approved grammatical structures. These are benefits that aid system engineers when developing new systems or maintaining requirements for existing systems as they evolve over time.

Future work could include (1) report generation into text documents other than Microsoft Word, (2) improved modeling of the system to reduce reliance on textual content, (3) addition of determiner elements in the requirement structure for improved grammar, (4) generating the text requirements within the model elements directly within the tool, (5) incorporating LLM tools to improve the readability of the generated requirements, (6) generation of metric reports for tracking requirement quality, and (7) quantitative measurement of the identified benefits of the methods developed in this paper.

Acknowledgments

Writefull provided assistance with grammar correction. All research design, interpretation, and conclusions are original author work.

References

AIR4094. (2016). *AIR4094A - aircraft flight control systems descriptions* (Standard). SAE International. https://doi.org/10.4271/AIR4094A

AIR4253. (2018). *AIR4253B description of actuation systems for aircraft with fly-by-wire flight control systems* (Standard). SAE International. https://doi.org/10.4271/AIR4253B

Ballard, M., Peak, R., Cimtalay, S., & Mavris, D. (2020). Bidirectional text-to-model element requirement transformation. *IEEE Aerospace Conference*. https://doi.org/10.1109/AERO47225.2020.9172306

Borky, J. M., & Bradley, T. H. (2019). *Effective model-based systems engineering* (1st ed.). Springer. https://doi.org/10.1007/978-3-319-95669-5

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

Friedenthal, S., Moore, A., & Steiner, R. (2014). *A practical guide to SysML* (3rd ed.). Elsevier. https://doi.org/10.1016/C2013-0-14457-1

Herber, D. R., & Eftekhari-Shahroudi, K. (2023). Building a requirements digital thread from concept to testing using model-based structured requirements applied to thrust reverser actuation system development. *Recent Advances in Aerospace Actuation Systems and Components*.

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

Hooks, I. F., & Farry, K. A. (2001). *Customer-centered products: Creating successful products through smart requirements management*. AMACOM.

Hull, E., Jackson, K., & Dick, J. (2011). *Requirements engineering* (3rd ed.). Springer. https://doi.org/10.1007/978-1-84996-405-0

ISO29148. (2018). *29148-2018 - ISO/IEC/IEEE international standard - systems and software engineering – life cycle processes – requirements engineering* (Standard). ISO/IEC/IEEE. https://doi.org/10.1109/IEEESTD.2018.8559686

Madni, A. M., Augustine, N., & Sievers, M. (2023). *Handbook of model-based systems engineering*. Springer Switzerland. https://doi.org/10.1007/978-3-030-93582-5

No Magic. (2024). *Generating requirement reports*. Retrieved May 8, 2024, from https://docs.nomagic.com/spaces/CRMP2024xR1/pages/170459635

No Magic. (2025). *Requirement table*. Retrieved January 6, 2025, from https://docs.nomagic.com/spaces/CRMP2024xR2/pages/189138567

OMG. (2024). *SysML® — OMG systems modeling language* (Standard No. v1.7). Object Management Group. https://www.omg.org/spec/SysML/1.7

Riesener, M., Dölle, C., Becker, A., Gorbatcheva, S., Rebentisch, E., & Schuh, G. (2021). Application of natural language processing for systematic requirement management in model-based systems engineering. *INCOSE International Symposium*, *31*(1), 806–815. https://doi.org/10.1002/j.2334-5837.2021.00871.x

Santos, J. L., Martins, L. E. G., & Molléri, J. S. (2025). Requirements extraction from model-based systems engineering: A systematic literature review. *Journal of Systems and Software*, *226*, 112407. https://doi.org/10.1016/j.jss.2025.112407

The Apache Software Foundation. (2020). *Velocity user guide*. Retrieved 2020, from https://velocity.apache.org/engine/devel/user-guide.html

Tikayat Ray, A., Cole, B. F., Pinon Fischer, O. J., Bhat, A. P., White, R. T., & Mavris, D. N. (2023). Agile methodology for the standardization of engineering requirements using large language models. *Systems*, *11*(7). https://doi.org/10.3390/systems11070352

Tikayat Ray, A., Pinon Fischer, O. J., White, R. T., Cole, B. F., & Mavris, D. N. (2024). Development of a language model for named-entity-recognition in aerospace requirements. *Journal of Aerospace Information Systems*, *21*(6), 489–499. https://doi.org/10.2514/1.I011251

Walden, D. D., et al. (2023). *Systems engineering handbook* (5th ed.) [INCOSE-TP-2003–002-05]. Wiley.

Wheatcraft, L., Ryan, M., et al. (2023). *Guide to writing requirements* [INCOSE-TP-2010-006-04]. INCOSE.

Wheaton, J. S., & Herber, D. R. (2024). Digital requirements engineering with an INCOSE-derived SysML meta-model. *Conference on Systems Engineering Research*, 15–26. https://doi.org/10.1007/978-3-031-62554-1_2

Biography

Michael Kellogg

Michael Kellogg is a PhD Candidate in the Department of Systems Engineering at Colorado State University in Fort Collins, CO, USA. He is also a Senior Staff Engineer for Woodward HRT in Santa Clarita, CA where he supports development of high-precision electro-hydraulic control components for the aerospace industry. His research interests include requirements engineering, model-based systems engineering, system dynamic, control theory, and electro-hydraulic / electro-mechanical position control. He is an INCOSE member and has been CSEP certified since 2009.

Daniel R. Herber

Dr. Daniel R. Herber is an Associate Professor in the Department of Systems Engineering at Colorado State University in Fort Collins, CO, USA. His research interests and projects have been in design optimization, model-based systems engineering, system architecture, digital engineering, dynamics and control, and combined physical and control system design (control co-design), frequently collaborating with academia, industry, and government laboratories. His work has involved several application domains, including energy, aerospace, defense, and software systems. He teaches courses in model-based systems engineering, system architecture, controls, and optimization. He is a member of INCOSE, ASME, and AIAA. He studied at the University of Illinois at Urbana-Champaign, earning his B.S. (2011) in General Engineering and his M.S. (2014) and Ph.D. (2017) in Systems and Entrepreneurial Engineering. He held a postdoctoral position (2018-2019) with the NSF ERC for Power Optimization for Electro-Thermal Systems (POETS).