Model-Based Systems Engineering Approach to Model & Simulate Space Experiments using Teamwork Cloud

Capt Christopher Reed
AFRL Space Vehicles Directorate
Kirtland AFB, NM
christopher.reed.62@spaceforce.mil

ABSTRACT

The Air Force Research Laboratory Space Vehicles Directorate (AFRL/RV) has a digital transformation underway with Model-Based Systems Engineering (MBSE) as a central pillar of the overall effort. The goal of the MBSE implementation is to assist in and not constrain the normal systems engineering processes that were refined over decades for space flight experiments. Each MBSE model acts as the source of truth for mission/system design, decisions, and communications that leads to building a usable technical baseline. AFRL/RV has an MBSE Model Management Framework that centers around an openly usable and tailorable Space Vehicle Reference Architecture (SVRA) as the hub of integration and updates for the enterprise of project models. The SVRA contains project usages (i.e., read-only versions) of a Component Library of common metamodel elements and reusable, pre-defined components; and an MBSE Style Guide written in Systems Modeling Language (SysML). The various AFRL/RV program models, originating from the SVRA, are reviewed and adopted into the Enterprise System Model where AFRL/RV personnel run analyses on the collection of verified models, and common components can be moved to the Component Library. These Component Library updates are then immediately usable, if the update is locally accepted, in other models built on the SVRA. The SVRA is further used as the starting model in training AFRL personnel how to use MBSE and SysML. The MBSE processes and methods utilized in the AFRL/RV MBSE implementation are the result of many pathfinder efforts across the Department of Defense (DoD) which have led to much faster execution and a clear path forward.

ABOUT THE AUTHORS

Capt Christopher Reed is a spaceflight mission engineer working in the Air Force Research Lab's Space Vehicles Directorate in the Model Based Systems Engineering and Analysis branch at Kirtland AFB, NM. Capt Reed has quickly been recognized as the leading subject matter expert in Model Based Systems Engineering (MBSE) within the Space vehicles Directorate through his demonstrated skills and knowledge with modeling systems of systems for spaceflight experiments.

Chris Reed graduated from Embry-Riddle Aeronautical University with a B.S. in Aerospace Engineering in December 2017 and commissioned into the Air Force to start working in the Air Force Life Cycle Management Center for the Simulators Division. He started work with MBSE when he was handpicked to work on a $900M project. He quickly learned MBSE to understand and effectively communicate the intricacy of new systems. After his work with the Simulators Division, Chris was competitively selected to earn a Master's in Science for Systems Engineering at the Air Force Institute of Technology in 2020. Prior to his current position, he was a systems engineering student at AFIT/ENV, Wright-Patterson AFB, OH and completed his degree in March 2022.

Model-Based Systems Engineering Approach to Model & Simulate Space Experiments using Teamwork Cloud

Capt Christopher Reed
AFRL Space Vehicles Directorate
Kirtland AFB, NM
christopher.reed.62@spaceforce.mil

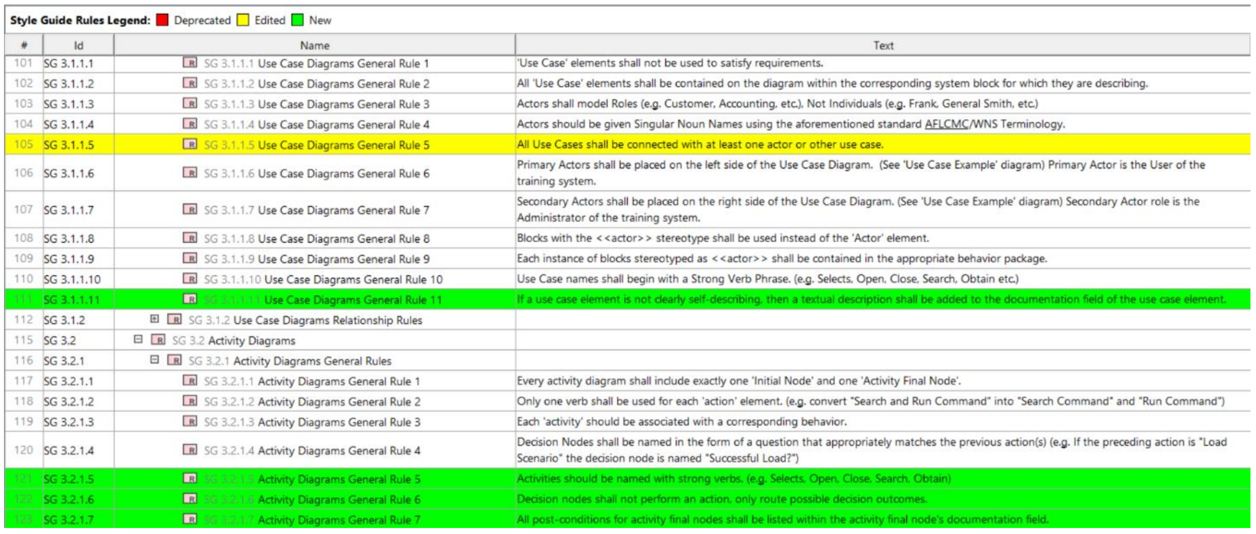
FOUNDATION

The digital engineering efforts within AFRL/RV are centered around the use of MBSE models as the single source of truth for our systems engineering efforts. We adapted this concept from best practices and lessons learned across the DoD to ensure that our digital engineering practices are in line with common application in the DoD, while also enabling innovation toward engineering in AFRL/RV space missions. A key partner in building these practices is the Air Force Institute of Technology (AFIT). The systems engineering department of AFIT has long been an advocate of MBSE and has educated hundreds of DoD professionals in the art of building and using models. Two key students that helped advance our MBSE practices are Captain Luke Farrel and Major Sean Kelly.

In 2020, then Lt Farrel started a CubeSat Reference Architecture with the objective of facilitating more “complete, streamlined, and transferable products throughout the course of a general CubeSat Development Process.” (Farrel, 2020) His work focused on the necessary MBSE elements and diagrams to design and perform preliminary analysis of the CubeSat system. This system focused approach is essential to AFRL/RV’s current spaceflight experiment design engineering work. Lt Farrel’s thesis was directly followed up by then Capt Kelly in 2021 with an expansion of the CubeSat Reference Architecture. Capt Kelly’s expansion focused on providing a reusable satellite template by capturing and reusing previous work in libraries, then continuing improvement by designing space for analysis templates into the architecture. (Kelly, 2021) His completed Satellite Reference Architecture (SRA) was adopted for MBSE research by AFRL/RV in 2021.

When I arrived in AFRL/RV in 2022 as an MBSE practitioner, I was impressed to see the SRA developed by Farrel and Kelly was used to design an AFRL/RV spaceflight experiment. I knew of the SRA from my time at AFIT but was more familiar with the Unmanned Aerial System Reference Architecture due to my thesis topic involving aerial systems. When I explored the SRA as part of my early work at AFRL/RV, I discovered that it had common heritage with many other DoD models that include SysML guides which relate back to the Air Force Life Cycle Management Center, Simulators Division (AFLCMC/WNS) MBSE Style Guide. This AFLCMC/WNS MBSE Style Guide has been published and presented previously at I/ITSEC in 2019 (Reed, 2019) and 2020 (Ayers et al, 2020), and it focuses on guiding engineers to create consistent models that can be easily integrated at an enterprise level. The epiphany of the common heritage happened when I reviewed the SRA modeling rules and recognized that my team back at AFLCMC/WNS had written most of the same rules in the MBSE Style Guide. Figure 1 is the modeling rules from the AFLCMC/WNS MBSE Style Guide, while Figure 2 is the modeling rules included in the SRA.

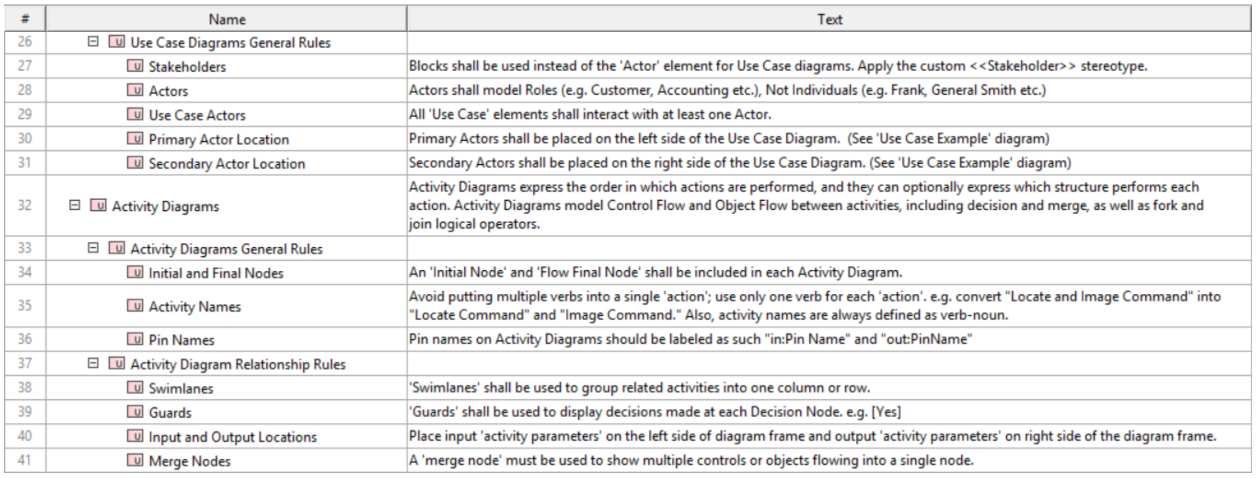
Another example of common modeling methodology across the U.S. Air Force is in the AFRL Systems Technology Office (AFRL/STO) MBSE Style Guide. Figure 3 is a set of nearly identical modeling rules used within another disparate organization. It is interesting to note that each organization has a different branch that has grown independently of the modeling rules modified throughout the evolution of the AFLCMC/WNS MBSE Style Guide. I believe the SRA had a branch from about V3.0 of the AFLCMC/WNS MBSE Style Guide, while AFRL/STO likely had a branch after V4.0 of the guide. Additionally, many organizations in the DoD have developed their own modeling guides, including AFRL/RV direct partners like the United States Space Force (USSF) Space Systems Command (SSC) and Space Warfare Analysis Center (SWAC). It is advantageous that multiple organizations have similar rules, however there will be a time when differences in rules will have to be smoothed out for a consistent DoD wide modeling standard.



Style Guide Rules Legend: 🟥 Deprecated 🟨 Edited 🟩 New

#	Id	Name	Text
101	SG 3.1.1.1	SG 3.1.1.1 Use Case Diagrams General Rule 1	'Use Case' elements shall not be used to satisfy requirements.
102	SG 3.1.1.2	SG 3.1.1.2 Use Case Diagrams General Rule 2	All 'Use Case' elements shall be contained on the diagram within the corresponding system block for which they are describing.
103	SG 3.1.1.3	SG 3.1.1.3 Use Case Diagrams General Rule 3	Actors shall model Roles (e.g. Customer, Accounting, etc.). Not Individuals (e.g. Frank, General Smith, etc.)
104	SG 3.1.1.4	SG 3.1.1.4 Use Case Diagrams General Rule 4	Actors should be given Singular Noun Names using the aforementioned standard AFLCMC/WNS Terminology.
105	SG 3.1.1.5	SG 3.1.1.5 Use Case Diagrams General Rule 5	All Use Cases shall be connected with at least one actor or other use case.
106	SG 3.1.1.6	SG 3.1.1.6 Use Case Diagrams General Rule 6	Primary Actors shall be placed on the left side of the Use Case Diagram. (See 'Use Case Example' diagram) Primary Actor is the User of the training system.
107	SG 3.1.1.7	SG 3.1.1.7 Use Case Diagrams General Rule 7	Secondary Actors shall be placed on the right side of the Use Case Diagram. (See 'Use Case Example' diagram) Secondary Actor role is the Administrator of the training system.
108	SG 3.1.1.8	SG 3.1.1.8 Use Case Diagrams General Rule 8	Blocks with the <<actor>> stereotype shall be used instead of the 'Actor' element.
109	SG 3.1.1.9	SG 3.1.1.9 Use Case Diagrams General Rule 9	Each instance of blocks stereotyped as <<actor>> shall be contained in the appropriate behavior package.
110	SG 3.1.1.10	SG 3.1.1.10 Use Case Diagrams General Rule 10	Use Case names shall begin with a Strong Verb Phrase. (e.g. Selects, Open, Close, Search, Obtain etc.)
111	SG 3.1.1.11	SG 3.1.1.11 Use Case Diagrams General Rule 11	If a use case element is not clearly self-describing, then a textual description shall be added to the documentation field of the use case element.
112	SG 3.1.2	⊞ SG 3.1.2 Use Case Diagrams Relationship Rules	
115	SG 3.2	⊟ SG 3.2 Activity Diagrams	
116	SG 3.2.1	⊟ SG 3.2.1 Activity Diagrams General Rules	
117	SG 3.2.1.1	SG 3.2.1.1 Activity Diagrams General Rule 1	Every activity diagram shall include exactly one 'Initial Node' and one 'Activity Final Node'.
118	SG 3.2.1.2	SG 3.2.1.2 Activity Diagrams General Rule 2	Only one verb shall be used for each 'action' element. (e.g. convert "Search and Run Command" into "Search Command" and "Run Command")
119	SG 3.2.1.3	SG 3.2.1.3 Activity Diagrams General Rule 3	Each 'activity' should be associated with a corresponding behavior.
120	SG 3.2.1.4	SG 3.2.1.4 Activity Diagrams General Rule 4	Decision Nodes shall be named in the form of a question that appropriately matches the previous action(s) (e.g. If the preceding action is "Load Scenario" the decision node is named "Successful Load?")
121	SG 3.2.1.5	SG 3.2.1.5 Activity Diagrams General Rule 5	Activities should be named with strong verbs. (e.g. Selects, Open, Close, Search, Obtain)
122	SG 3.2.1.6	SG 3.2.1.6 Activity Diagrams General Rule 6	Decision nodes shall not perform an action, only route possible decision outcomes.
123	SG 3.2.1.7	SG 3.2.1.7 Activity Diagrams General Rule 7	All post-conditions for activity final nodes shall be listed within the activity final node's documentation field.

Figure 1. AFLCMC/WNS MBSE Style Guide V5.0 Rules



#	Name	Text
26	⊟ Use Case Diagrams General Rules	
27	Stakeholders	Blocks shall be used instead of the 'Actor' element for Use Case diagrams. Apply the custom <<Stakeholder>> stereotype.
28	Actors	Actors shall model Roles (e.g. Customer, Accounting etc.), Not Individuals (e.g. Frank, General Smith etc.)
29	Use Case Actors	All 'Use Case' elements shall interact with at least one Actor.
30	Primary Actor Location	Primary Actors shall be placed on the left side of the Use Case Diagram. (See 'Use Case Example' diagram)
31	Secondary Actor Location	Secondary Actors shall be placed on the right side of the Use Case Diagram. (See 'Use Case Example' diagram)
32	⊟ Activity Diagrams	Activity Diagrams express the order in which actions are performed, and they can optionally express which structure performs each action. Activity Diagrams model Control Flow and Object Flow between activities, including decision and merge, as well as fork and join logical operators.
33	⊟ Activity Diagrams General Rules	
34	Initial and Final Nodes	An 'Initial Node' and 'Flow Final Node' shall be included in each Activity Diagram.
35	Activity Names	Avoid putting multiple verbs into a single 'action'; use only one verb for each 'action'. e.g. convert "Locate and Image Command" into "Locate Command" and "Image Command." Also, activity names are always defined as verb-noun.
36	Pin Names	Pin names on Activity Diagrams should be labeled as such "in:Pin Name" and "out:PinName"
37	⊟ Activity Diagram Relationship Rules	
38	Swimlanes	'Swimlanes' shall be used to group related activities into one column or row.
39	Guards	'Guards' shall be used to display decisions made at each Decision Node. e.g. [Yes]
40	Input and Output Locations	Place input 'activity parameters' on the left side of diagram frame and output 'activity parameters' on right side of the diagram frame.
41	Merge Nodes	A 'merge node' must be used to show multiple controls or objects flowing into a single node.

Figure 2. SRA Modeling Rules

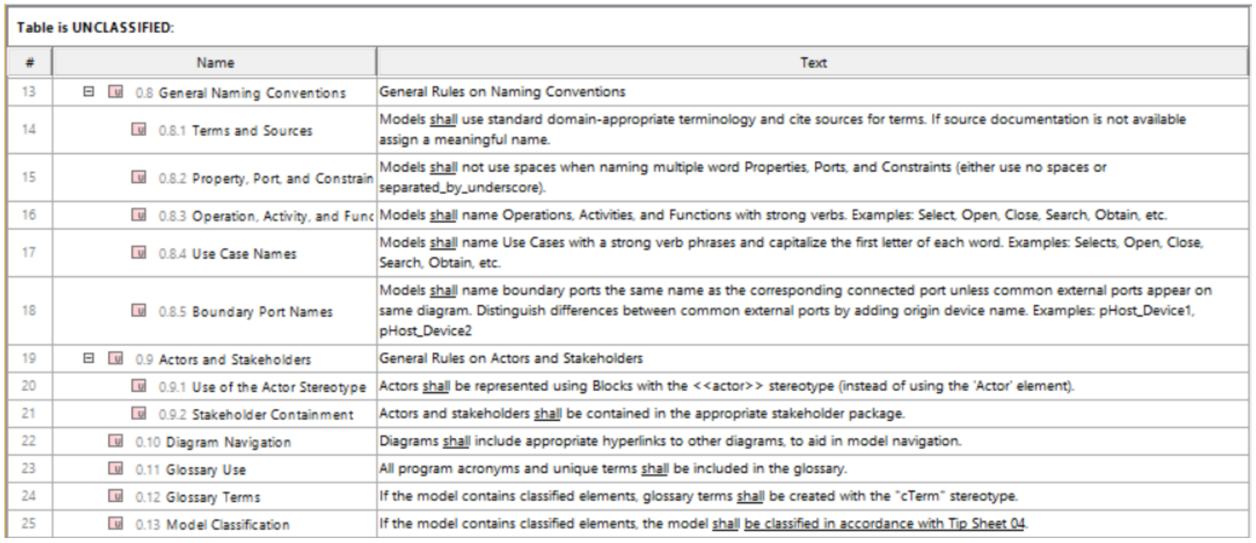


Table is UNCLASSIFIED:

#	Name	Text
13	⊟ 0.8 General Naming Conventions	General Rules on Naming Conventions
14	0.8.1 Terms and Sources	Models shall use standard domain-appropriate terminology and cite sources for terms. If source documentation is not available assign a meaningful name.
15	0.8.2 Property, Port, and Constrain	Models shall not use spaces when naming multiple word Properties, Ports, and Constraints (either use no spaces or separated_by_underscore).
16	0.8.3 Operation, Activity, and Func	Models shall name Operations, Activities, and Functions with strong verbs. Examples: Select, Open, Close, Search, Obtain, etc.
17	0.8.4 Use Case Names	Models shall name Use Cases with a strong verb phrases and capitalize the first letter of each word. Examples: Selects, Open, Close, Search, Obtain, etc.
18	0.8.5 Boundary Port Names	Models shall name boundary ports the same name as the corresponding connected port unless common external ports appear on same diagram. Distinguish differences between common external ports by adding origin device name. Examples: pHost_Device1, pHost_Device2
19	⊟ 0.9 Actors and Stakeholders	General Rules on Actors and Stakeholders
20	0.9.1 Use of the Actor Stereotype	Actors shall be represented using Blocks with the <<actor>> stereotype (instead of using the 'Actor' element).
21	0.9.2 Stakeholder Containment	Actors and stakeholders shall be contained in the appropriate stakeholder package.
22	0.10 Diagram Navigation	Diagrams shall include appropriate hyperlinks to other diagrams, to aid in model navigation.
23	0.11 Glossary Use	All program acronyms and unique terms shall be included in the glossary.
24	0.12 Glossary Terms	If the model contains classified elements, glossary terms shall be created with the "cTerm" stereotype.
25	0.13 Model Classification	If the model contains classified elements, the model shall be classified in accordance with Tip Sheet 04.

Figure 3. AFRL/STO Style Guide V1.2 Rules

APPLICATION

The SRA was an excellent starting point to build an AFRL/RV specific architecture. However, we believed that the starting point for AFRL/RV models should be broader than just a reference architecture. Figure 4 depicts the overall model management approach we use at AFRL/RV. The top layer of Reference Architecture, Component Library, and Style Guide is an adaptation of the SRA for use on space vehicles designed and operated by AFRL.

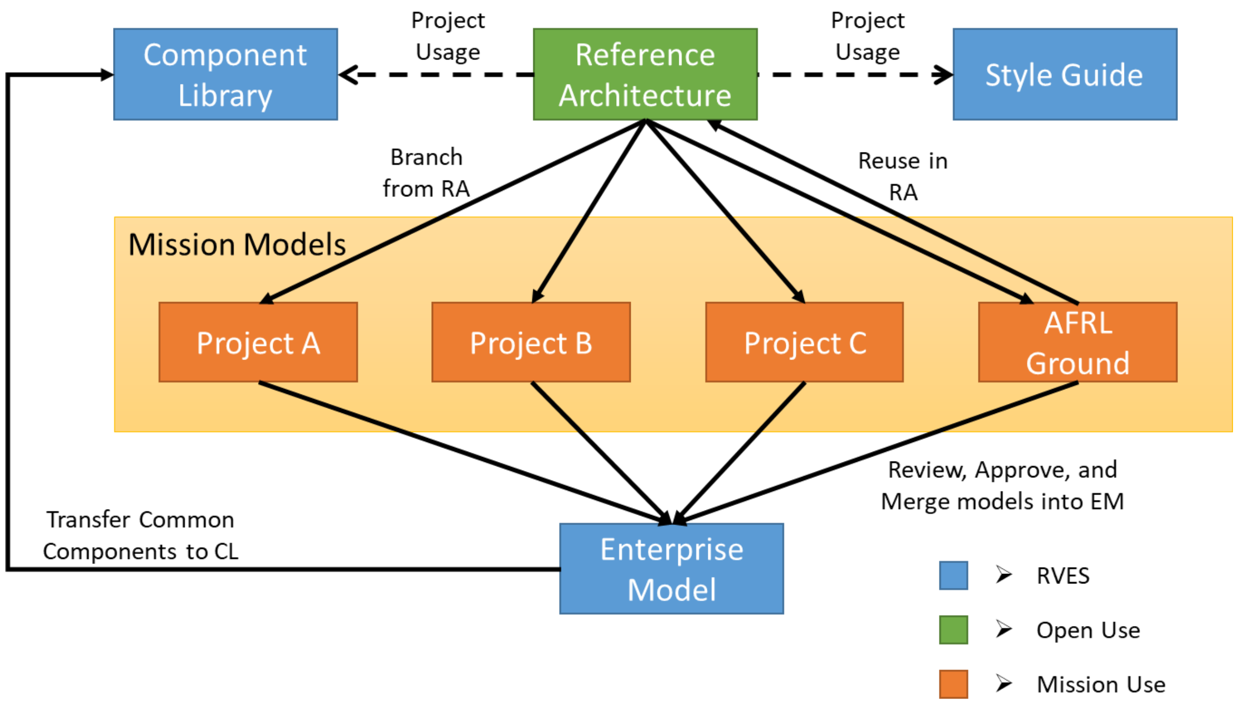
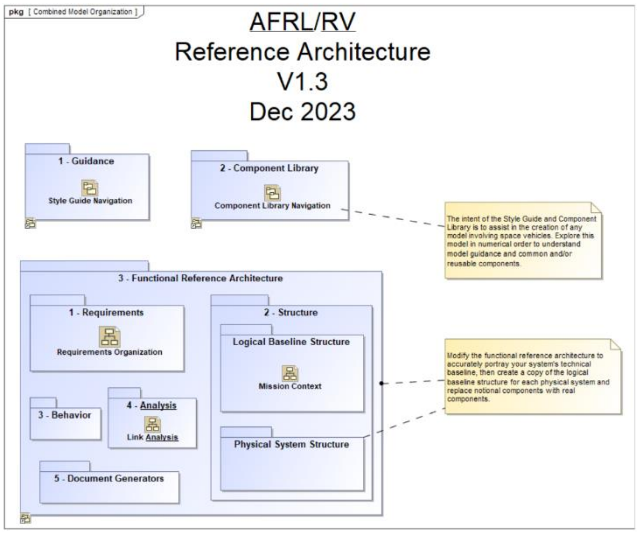


Figure 4. AFRL/RV Model Management Framework

A critical difference between the SRA and SVRA is the separation of models and the inclusion of "Project Usages" which is a modeling feature that creates read-only usage of data from other models. Currently, the SVRA has a project usage that pulls another "payload" project into the SVRA to include the Component Library and Style Guide. The advantage to this separation is the ability for mission models to update their "payload" model with the newest style guide and component library material without worrying about breaking their model. We have developed two different methods to accept these updates. For mission models inside of AFRL's digital engineering collaboration site, the Digital Laboratory Environment (DLE), the project usage can be simply updated by selecting the newest version of the "payload" model from the Teamwork Cloud within the mission model's project usage menu. For mission models in other networks, the "payload" project usage can be replaced by any static release of the SVRA payload model.

The SVRA

Upon opening the SVRA, users are greeted by a straightforward navigation diagram called "Combined Model Organization" as seen in Figure 5. The packages "1 - Guidance" and "2 - Component Library"

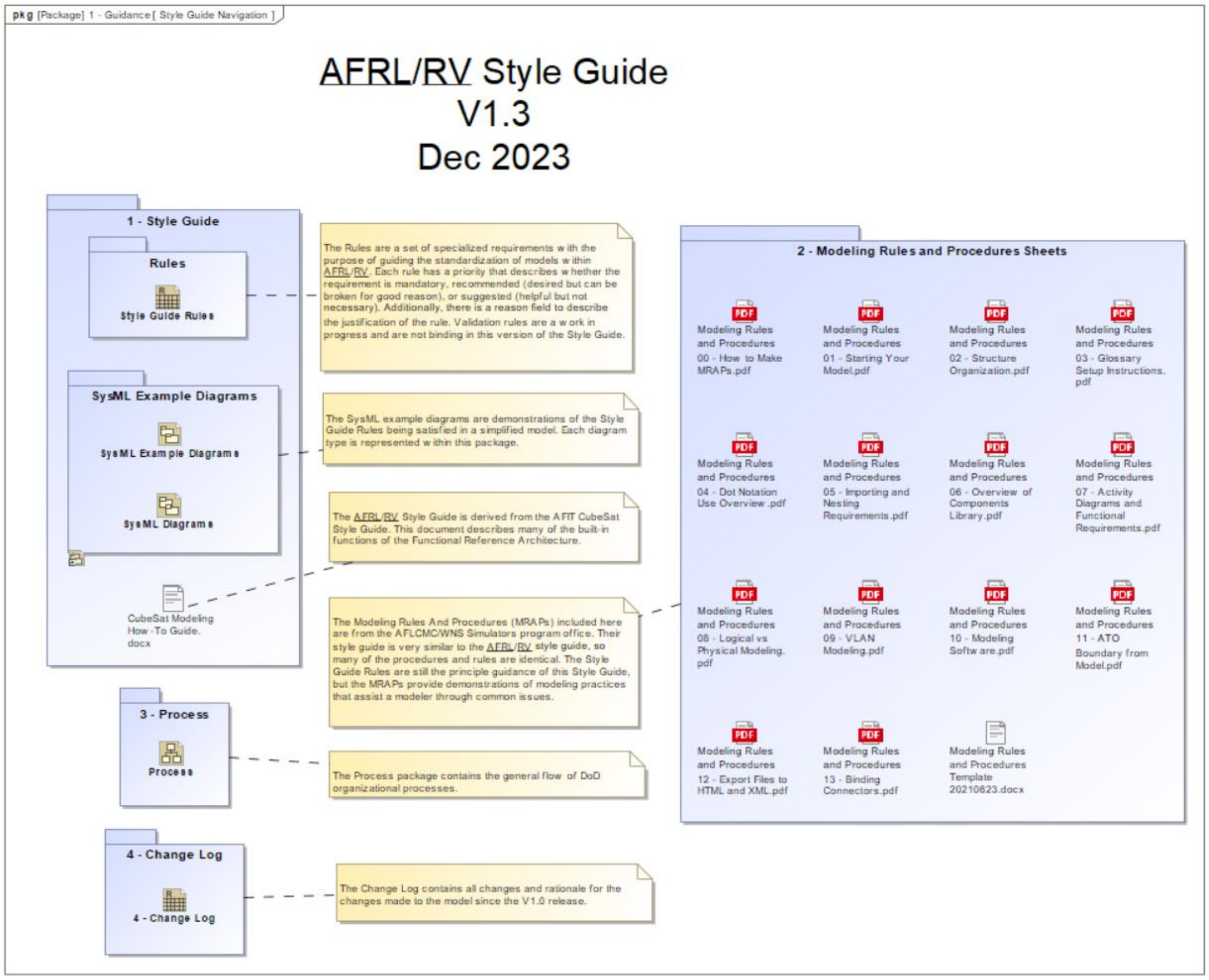


The intent of the Style Guide and Component Library is to assist in the creation of any model involving space vehicles. Explore this model in numerical order to understand model guidance and common and/or reusable components.

Modify the functional reference architecture to accurately portray your system's technical baseline, then create a copy of the logical baseline structure for each physical system and replace notional components with real components.

Figure 5. SVRA Combined Model Organization

directly correlate to the Style Guide and Component Library from Figure 4. Clicking anywhere in the "1 – Guidance" package will open a new diagram, Figure 6, which is the Style Guide navigation page. The Style Guide contains the same rules as seen in the SRA, with some additional SysML example diagrams that depict how the rules can be satisfied. Figure 7 shows how the rules are tied together with the example diagrams to aid modelers with quick navigation to see how the rules are meant to be used in practice. In comparison to the other rule tables in Figures 1-3, I have received feedback that the links to examples have been incredibly helpful from engineers that do not work with MBSE every day and need to check rules for refresher information. The AFRL/RV rules also include some new data with the priority of the rule and the reason for the rule. These pieces of data are frequently used in document-based style guides and were added as data fields in the "styleGuideRequirement" custom stereotype so the data will show up any time a style guide rule appears.


pkg [Package] 1 - Guidance [Style Guide Navigation]
AFRL/RV Style Guide
V1.3
Dec 2023
1 - Style Guide
Rules
Style Guide Rules
SysML Example Diagrams
SysML Example Diagrams
SysML Diagrams
CubeSat Modeling How-To Guide.docx
3 - Process
Process
4 - Change Log
4 - Change Log
2 - Modeling Rules and Procedures Sheets
Modeling Rules and Procedures 00 - How to Make MRAPs.pdf
Modeling Rules and Procedures 01 - Starting Your Model.pdf
Modeling Rules and Procedures 02 - Structure Organization.pdf
Modeling Rules and Procedures 03 - Glossary Setup Instructions.pdf
Modeling Rules and Procedures 04 - Dot Notation Use Overview .pdf
Modeling Rules and Procedures 05 - Importing and Nesting Requirements.pdf
Modeling Rules and Procedures 06 - Overview of Components Library.pdf
Modeling Rules and Procedures 07 - Activity Diagrams and Functional Requirements.pdf
Modeling Rules and Procedures 08 - Logical vs Physical Modeling.pdf
Modeling Rules and Procedures 09 - VLAN Modeling.pdf
Modeling Rules and Procedures 10 - Modeling Software.pdf
Modeling Rules and Procedures 11 - ATO Boundary from Model.pdf
Modeling Rules and Procedures 12 - Export Files to HTML and XML.pdf
Modeling Rules and Procedures 13 - Binding Connectors.pdf
Modeling Rules and Procedures Template 20210623.docx

The Rules are a set of specialized requirements with the purpose of guiding the standardization of models within AFRL/RV. Each rule has a priority that describes whether the requirement is mandatory, recommended (desired but can be broken for good reason), or suggested (helpful but not necessary). Additionally, there is a reason field to describe the justification of the rule. Validation rules are a work in progress and are not binding in this version of the Style Guide.

The SysML example diagrams are demonstrations of the Style Guide Rules being satisfied in a simplified model. Each diagram type is represented within this package.

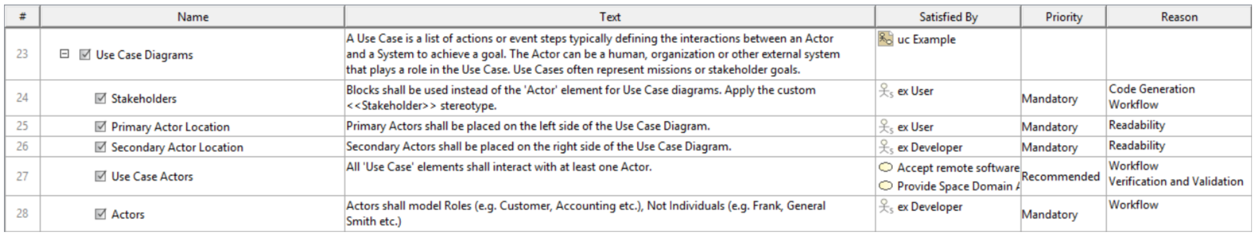
The AFRL/RV Style Guide is derived from the AFIT CubeSat Style Guide. This document describes many of the built-in functions of the Functional Reference Architecture.

The Modeling Rules And Procedures (MRAPs) included here are from the AFLCMC/WNS Simulators program office. Their style guide is very similar to the AFRL/RV style guide, so many of the procedures and rules are identical. The Style Guide Rules are still the principle guidance of this Style Guide, but the MRAPs provide demonstrations of modeling practices that assist a modeler through common issues.

The Process package contains the general flow of DoD organizational processes.

The Change Log contains all changes and rationale for the changes made to the model since the V1.0 release.

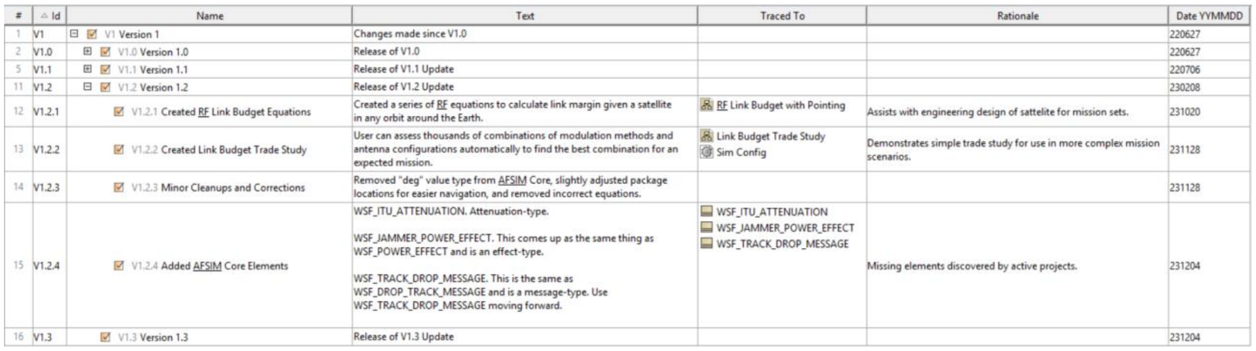
Figure 6. AFRL/RV Style Guide Navigation



#	Name	Text	Satisfied By	Priority	Reason
23	⊟ ☑ Use Case Diagrams	A Use Case is a list of actions or event steps typically defining the interactions between an Actor and a System to achieve a goal. The Actor can be a human, organization or other external system that plays a role in the Use Case. Use Cases often represent missions or stakeholder goals.	uc Example		
24	☑ Stakeholders	Blocks shall be used instead of the 'Actor' element for Use Case diagrams. Apply the custom <<Stakeholder>> stereotype.	ex User	Mandatory	Code Generation Workflow
25	☑ Primary Actor Location	Primary Actors shall be placed on the left side of the Use Case Diagram.	ex User	Mandatory	Readability
26	☑ Secondary Actor Location	Secondary Actors shall be placed on the right side of the Use Case Diagram.	ex Developer	Mandatory	Readability
27	☑ Use Case Actors	All 'Use Case' elements shall interact with at least one Actor.	Accept remote software Provide Space Domain A	Recommended	Workflow Verification and Validation
28	☑ Actors	Actors shall model Roles (e.g. Customer, Accounting etc.), Not Individuals (e.g. Frank, General Smith etc.)	ex Developer	Mandatory	Workflow

Figure 7. AFRL/RV Style Guide Rules

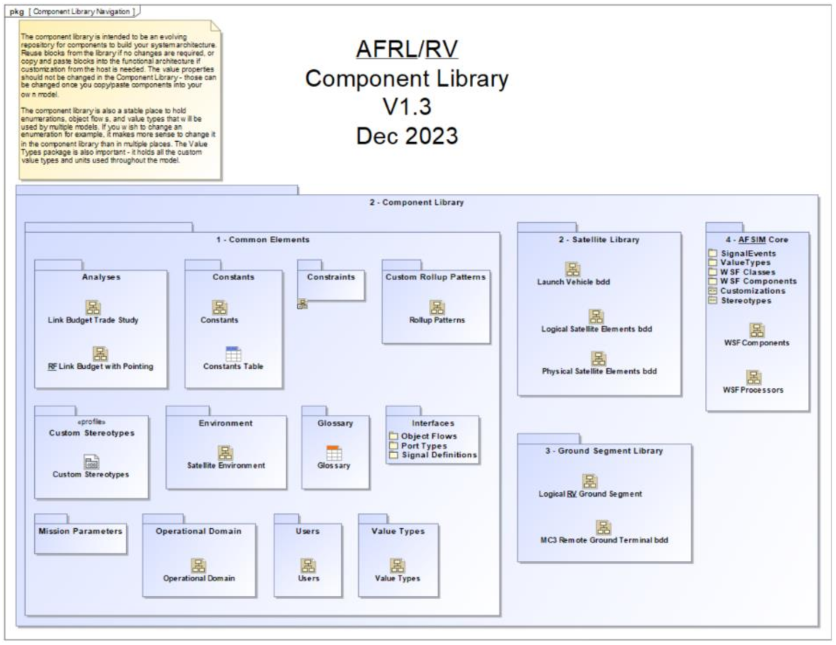
Package 2 in Figure 6 is a collection of step-by-step instructions for complex modeling tasks that I reused from the AFLCMC/WNS Style Guide. These Modeling Rules and Procedures Sheets (MRAPs) have positive feedback from various modelers, where the MRAP has saved hours of work planning how to model or complete the complex task. Package 3 is a general process diagram that explains how to create and track DoD requirements through the model. Finally, package 4 contains the change log for every change made in the SVRA from V1.0. As shown in Figure 8, the change log continues the excellent traceability from the rules by directly linking the new additions for the user to view.



#	Id	Name	Text	Traced To	Rationale	Date YYMMDD
1	V1	⊟ ✓ V1 Version 1	Changes made since V1.0			220627
2	V1.0	⊞ ✓ V1.0 Version 1.0	Release of V1.0			220627
5	V1.1	⊞ ✓ V1.1 Version 1.1	Release of V1.1 Update			220706
11	V1.2	⊟ ✓ V1.2 Version 1.2	Release of V1.2 Update			230208
12	V1.2.1	✓ V1.2.1 Created RF Link Budget Equations	Created a series of RF equations to calculate link margin given a satellite in any orbit around the Earth.	RF Link Budget with Pointing	Assists with engineering design of sattelite for mission sets.	231020
13	V1.2.2	✓ V1.2.2 Created Link Budget Trade Study	User can assess thousands of combinations of modulation methods and antenna configurations automatically to find the best combination for an expected mission.	Link Budget Trade Study Sim Config	Demonstrates simple trade study for use in more complex mission scenarios.	231128
14	V1.2.3	✓ V1.2.3 Minor Cleanups and Corrections	Removed "deg" value type from AFSIM Core, slightly adjusted package locations for easier navigation, and removed incorrect equations.			231128
15	V1.2.4	✓ V1.2.4 Added AFSIM Core Elements	WSF_ITU_ATTENUATION. Attenuation-type. WSF_JAMMER_POWER_EFFECT. This comes up as the same thing as WSF_POWER_EFFECT and is an effect-type. WSF_TRACK_DROP_MESSAGE. This is the same as WSF_DROP_TRACK_MESSAGE and is a message-type. Use WSF_TRACK_DROP_MESSAGE moving forward.	WSF_ITU_ATTENUATION WSF_JAMMER_POWER_EFFECT WSF_TRACK_DROP_MESSAGE	Missing elements discovered by active projects.	231204
16	V1.3	✓ V1.3 Version 1.3	Release of V1.3 Update			231204

Figure 8. AFRL/RV Style Guide Change Log

Returning to Figure 5, the next package to explore would be "2 - Component Library." The component library is the collection of every reusable component across the enterprise of AFRL/RV missions. Figure 9 shows that the component library contains a set of common elements, including: analyses, mathematical constants, value types, equations, custom stereotypes, a glossary of terms, and other reusable elements. Additionally, the component library has ever-expanding stores of satellite components, ground components, and Advanced Framework for Simulation, Integration and Modeling (AFSIM) representative components.



The component library is intended to be an evolving repository for components to build your system architecture. Reuse blocks from the library if no changes are required, or copy and paste blocks into the functional architecture if customization from the host is needed. The value properties should not be changed in the Component Library - those can be changed once you copy/paste components into your ow n model.

The component library is also a stable place to hold enumerations, object flow s, and value types that w ill be used by multiple models. If you w ish to change an enumeration for example, it makes more sense to change it in the component library than in multiple places. The Value Types package is also important - it holds all the custom value types and units used throughout the model.

Figure 9. AFRL/RV Component Library

Returning to Figure 5 again, the final package is the SVRA itself. The SVRA is a single package model of a generic satellite system using the Object-Oriented Systems Engineering Methodology (OOSEM). The tailorable, pre-populated info as shown in Figure 10 is limited to common DoD requirement organization, a generic satellite mission structure, and an example usage of the link budget trade study from the component library. The SVRA is the start for any MBSE model involving AFRL/RV space missions, which cover a vast variety of domains. Importantly, the SVRA is a tailorable starting point for models; a modeler can change anything in the SVRA for any reason to fit their mission needs.

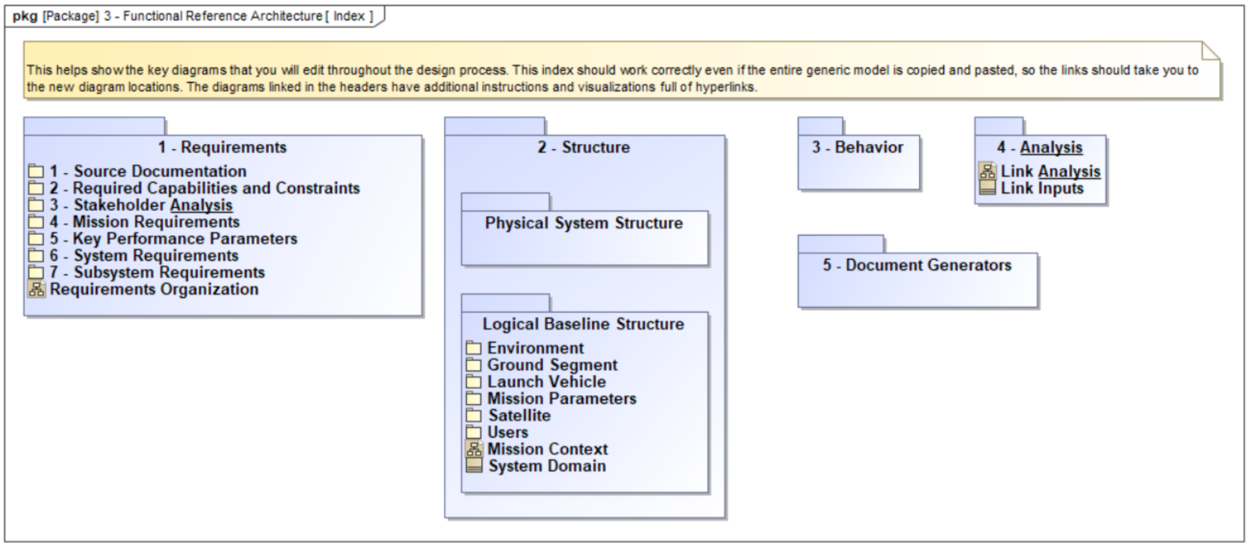

pkg [Package] 3 - Functional Reference Architecture [Index]
This helps show the key diagrams that you will edit throughout the design process. This index should work correctly even if the entire generic model is copied and pasted, so the links should take you to the new diagram locations. The diagrams linked in the headers have additional instructions and visualizations full of hyperlinks.
1 - Requirements
1 - Source Documentation
2 - Required Capabilities and Constraints
3 - Stakeholder Analysis
4 - Mission Requirements
5 - Key Performance Parameters
6 - System Requirements
7 - Subsystem Requirements
Requirements Organization
2 - Structure
Physical System Structure
Logical Baseline Structure
Environment
Ground Segment
Launch Vehicle
Mission Parameters
Satellite
Users
Mission Context
System Domain
3 - Behavior
4 - Analysis
Link Analysis
Link Inputs
5 - Document Generators

Figure 10. SVRA Functional Package Structure

Mission Models

The models we use for spaceflight experiments are not publicly releasable; however, I teach hands-on MBSE classes to a broad variety of audiences and use the SVRA as the starting point for generating class models. This paper will present 3 of the 10 class models as an analog for actual mission models because the approach is similar, just with 4 days of fidelity on an imaginary mission rather than the months needed to complete actual mission models. One major advantage of my class is that each class's model is a new, entirely independent concept that they find interesting from an engineering perspective. This approach results in a large collection of unique projects containing at least one example of each SysML diagram type and the inter-relations between diagrams.

The Battle Station

In a February 2023 class, one student asked if we could model a big, moon-like battle station from his favorite sci-fi series. The class agreed it would be interesting to model, so we designed the Mega Moon Battle Station (MMBS). The model ended up being straight-forward to build from the SVRA because the battle station was essentially a massively scaled-up space vehicle with an added energy cannon. The class had excellent examples from the sci-fi film to help populate behaviors (Figure 11), requirements (Figure 12), structure (Figure 13), and mathematical analysis (Figure 14) to fit with the class curriculum. The most significant challenge was creating the equations for free-space path loss to determine the range of the energy cannon. It turns out, for the absurd requirement of destroying a planet from 10 AU distance, the MMBS would require an equally absurd 715 dBW of power. For perspective, 715 dBW of power is approximately 10^{15} times the annual power output of the sun.

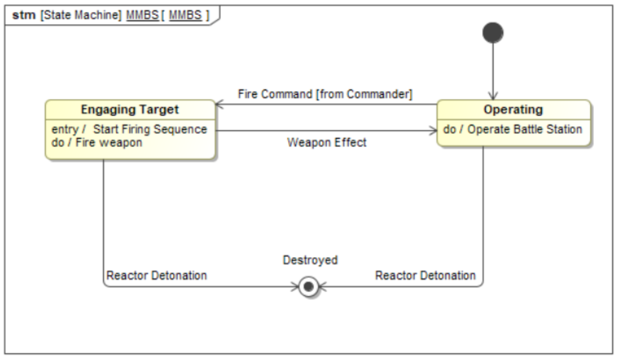

stm [State Machine] MMBS [MMBS]
Fire Command [from Commander]
Engaging Target
entry / Start Firing Sequence
do / Fire weapon
Weapon Effect
Operating
do / Operate Battle Station
Reactor Detonation
Destroyed
Reactor Detonation

Figure 11. MMBS Simple State Machine

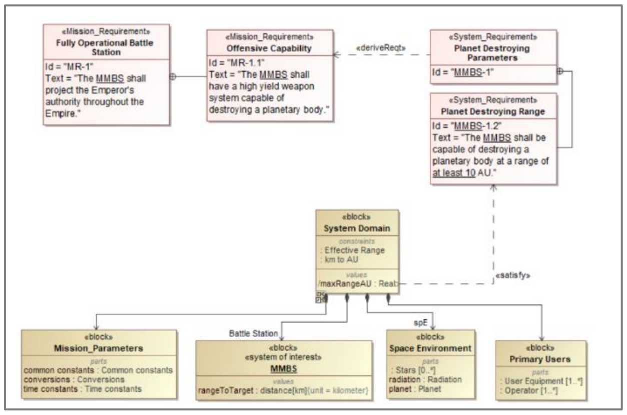


Figure 12. MMBS Requirement Flow

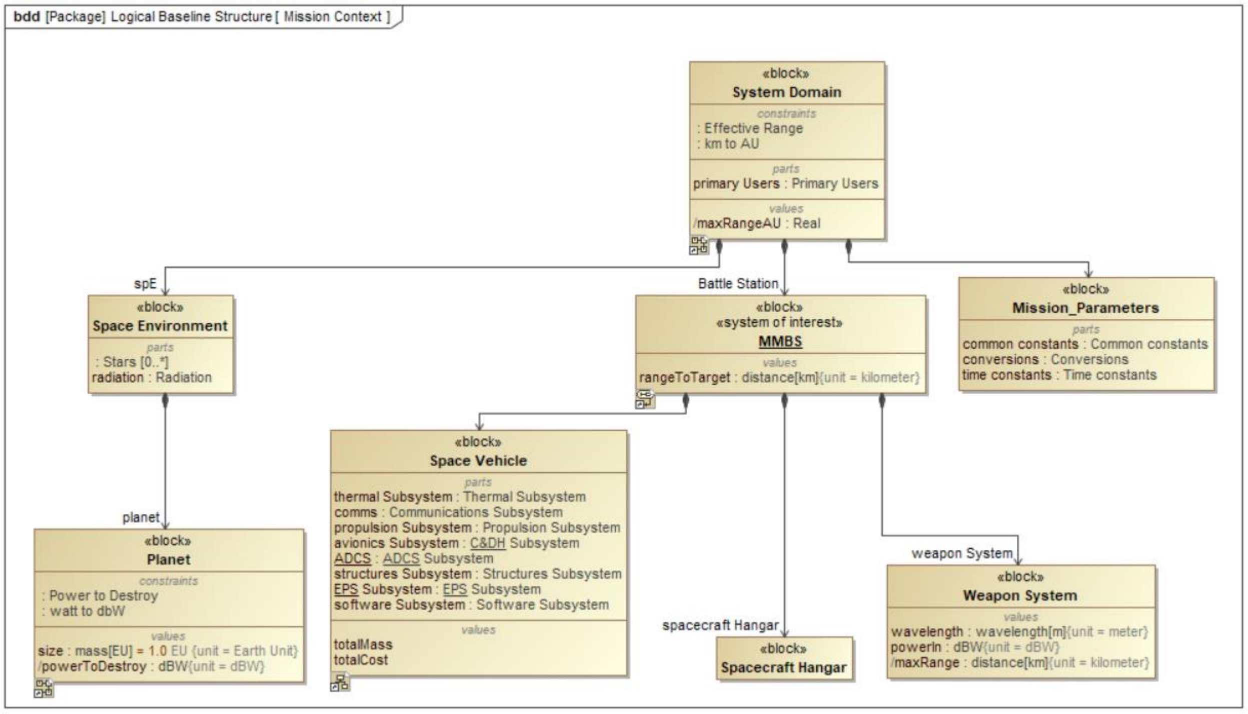
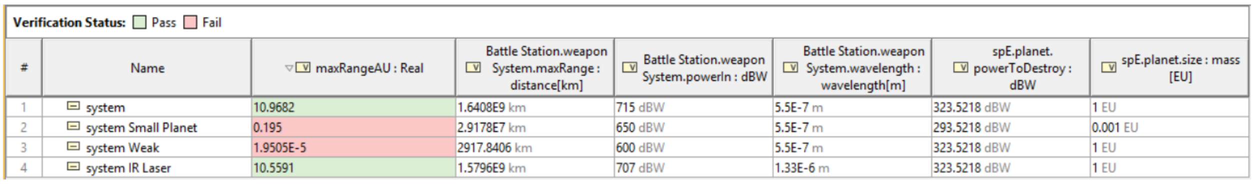

bdd [Package] Logical Baseline Structure [Mission Context]
«block» System Domain
constraints
: Effective Range
: km to AU
parts
primary Users : Primary Users
values
/maxRangeAU : Real
spE
«block» Space Environment
parts
: Stars [0..*]
radiation : Radiation
Battle Station
«block» «system of interest» MMBS
values
rangeToTarget : distance[km]{unit = kilometer}
«block» Mission_Parameters
parts
common constants : Common constants
conversions : Conversions
time constants : Time constants
«block» Space Vehicle
parts
thermal Subsystem : Thermal Subsystem
comms : Communications Subsystem
propulsion Subsystem : Propulsion Subsystem
avionics Subsystem : C&DH Subsystem
ADCS : ADCS Subsystem
structures Subsystem : Structures Subsystem
EPS Subsystem : EPS Subsystem
software Subsystem : Software Subsystem
values
totalMass
totalCost
planet
«block» Planet
constraints
: Power to Destroy
: watt to dbW
values
size : mass[EU] = 1.0 EU {unit = Earth Unit}
/powerToDestroy : dBW{unit = dBW}
spacecraft Hangar
«block» Spacecraft Hangar
weapon System
«block» Weapon System
values
wavelength : wavelength[m]{unit = meter}
powerIn : dBW{unit = dBW}
/maxRange : distance[km]{unit = kilometer}

Figure 13. MMBS Structural Decomposition



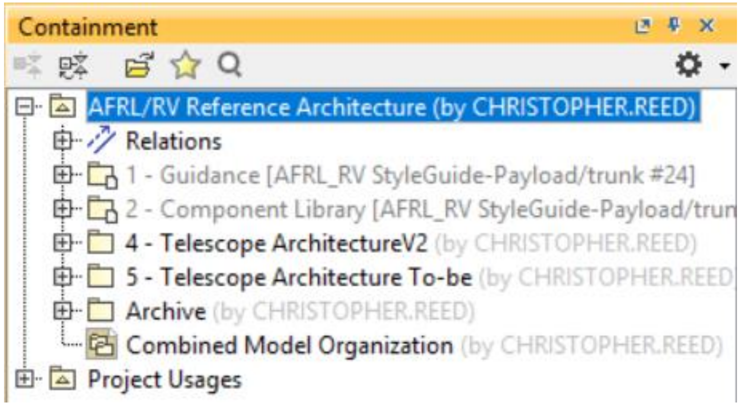
Verification Status: 🟩 Pass 🟥 Fail

#	Name	maxRangeAU : Real	Battle Station.weapon System.maxRange : distance[km]	Battle Station.weapon System.powerIn : dBW	Battle Station.weapon System.wavelength : wavelength[m]	spE.planet. powerToDestroy : dBW	spE.planet.size : mass [EU]
1	system	10.9682	1.6408E9 km	715 dBW	5.5E-7 m	323.5218 dBW	1 EU
2	system Small Planet	0.195	2.9178E7 km	650 dBW	5.5E-7 m	293.5218 dBW	0.001 EU
3	system Weak	1.9505E-5	2917.8406 km	600 dBW	5.5E-7 m	323.5218 dBW	1 EU
4	system IR Laser	10.5591	1.5796E9 km	707 dBW	1.33E-6 m	323.5218 dBW	1 EU

Figure 14. MMBS Requirement Verification Analysis

The Telescopes

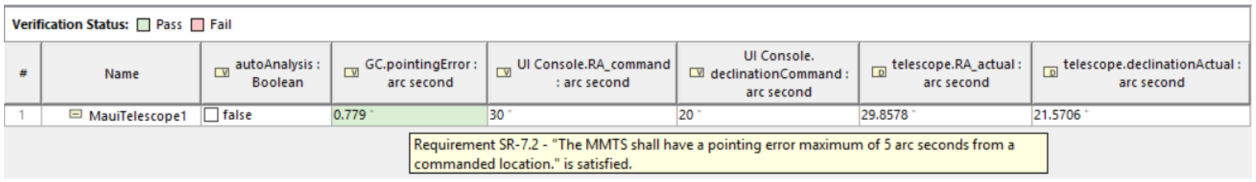
An October 2023 class was inspired by the intricate MMBS model and wanted to build a completely different concept of at least equal complexity. The class decided we should model a large telescope called the Mega Moon Telescope System (MMTS). The MMTS model also had behaviors, requirements, structure, and mathematical analyses; however, this class was much more interested in how to reuse this model for enterprise sustainment. OOSEM prescribes an object management style of as-is and to-be designs, like an operational mode and development mode for software. The as-is design is the current baseline(s) of the system and their physical implementations; when the to-be design is approved and built, it becomes the as-is design, and the old design is archived upon retirement. The example of the containment tree from this class is shown in Figure 15. Each physical design for this class was a SysML instance that showed design specifications against requirements like in Figure 16. Remarkably, this MMTS model is still being used by the group that took the class for their own telescope reference architecture.



Containment

- AFRL/RV Reference Architecture (by CHRISTOPHER.REED)
  - Relations
  - 1 - Guidance [AFRL_RV StyleGuide-Payload/trunk #24]
  - 2 - Component Library [AFRL_RV StyleGuide-Payload/trun
  - 4 - Telescope ArchitectureV2 (by CHRISTOPHER.REED)
  - 5 - Telescope Architecture To-be (by CHRISTOPHER.REED
  - Archive (by CHRISTOPHER.REED)
  - Combined Model Organization (by CHRISTOPHER.REED)
- Project Usages

Figure 15. MMTS Containment Tree



Verification Status: 🟩 Pass 🟥 Fail

#	Name	autoAnalysis : Boolean	GC.pointingError : arc second	UI Console.RA_command : arc second	UI Console. declinationCommand : arc second	telescope.RA_actual : arc second	telescope.declinationActual : arc second
1	MauiTelescope1	☐ false	0.779 ″	30 ″	20 ″	29.8578 ″	21.5706 ″

Requirement SR-7.2 - "The MMTS shall have a pointing error maximum of 5 arc seconds from a commanded location." is satisfied.

Figure 16. MMTS Telescope Instance

The SADSat

The most recent class, in June 2024, decided to use a more conventional concept for their model. This class wanted to model a satellite that could orbit the earth and perform maneuvers. The name of the design was SADSat, and it was able to heavily utilize the component library due to the modest mission description. For starters, the class wanted to design a CubeSat sized vehicle which enabled reuse of many components found in the original SRA. Additionally, this design was able to utilize the Component Library V1.3 features with orbit determination analyses to satisfy design requirements as seen in Figure 17.

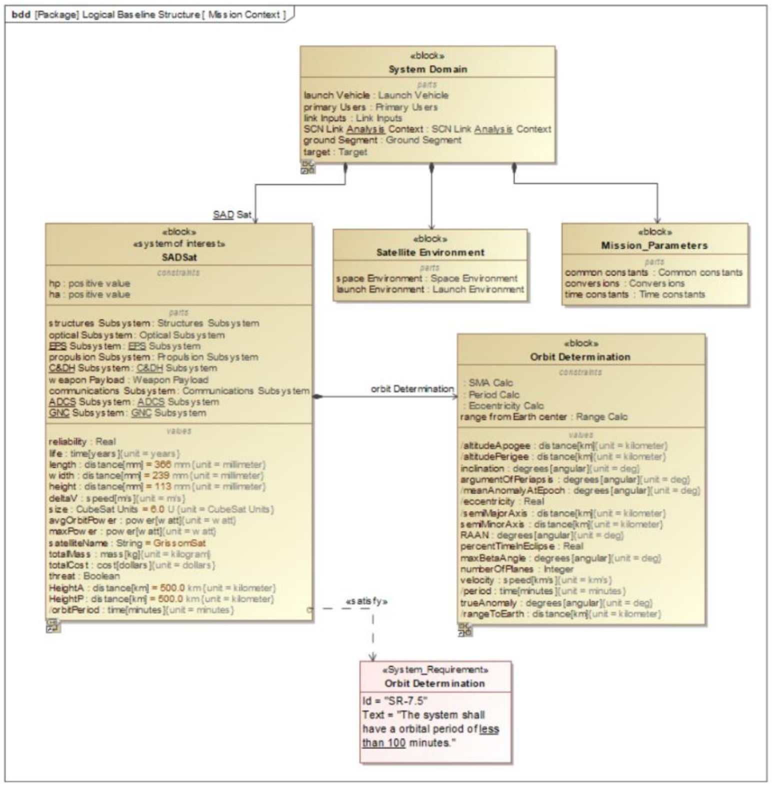

bdd [Package] Logical Baseline Structure [Mission Context]
«block» System Domain
parts
launch Vehicle : Launch Vehicle
primary Users : Primary Users
link Inputs : Link Inputs
SCN Link Analysis Context : SCN Link Analysis Context
ground Segment : Ground Segment
target : Target
SAD Sat
«block» «system of interest» SADSat
constraints
hp : positive value
ha : positive value
parts
structures Subsystem : Structures Subsystem
optical Subsystem : Optical Subsystem
EPS Subsystem : EPS Subsystem
propulsion Subsystem : Propulsion Subsystem
C&DH Subsystem : C&DH Subsystem
weapon Payload : Weapon Payload
communications Subsystem : Communications Subsystem
ADCS Subsystem : ADCS Subsystem
GNC Subsystem : GNC Subsystem
values
reliability : Real
life : time[years]{unit = years}
length : distance[mm] = 366 mm {unit = millimeter}
width : distance[mm] = 239 mm {unit = millimeter}
height : distance[mm] = 113 mm {unit = millimeter}
deltaV : speed[m/s]{unit = m/s}
size : CubeSat Units = 6.0 U {unit = CubeSat Units}
avgOrbitPower : power[watt]{unit = watt}
maxPower : power[watt]{unit = watt}
satelliteName : String = GrissomSat
totalMass : mass[kg]{unit = kilogram}
totalCost : cost[dollars]{unit = dollars}
threat : Boolean
HeightA : distance[km] = 500.0 km {unit = kilometer}
HeightP : distance[km] = 500.0 km {unit = kilometer}
/orbitPeriod : time[minutes]{unit = minutes}
«block» Satellite Environment
parts
space Environment : Space Environment
launch Environment : Launch Environment
«block» Mission_Parameters
parts
common constants : Common constants
conversions : Conversions
time constants : Time constants
orbit Determination
«block» Orbit Determination
constraints
: SMA Calc
: Period Calc
: Eccentricity Calc
range from Earth center : Range Calc
values
/altitudeApogee : distance[km]{unit = kilometer}
/altitudePerigee : distance[km]{unit = kilometer}
inclination : degrees[angular]{unit = deg}
argumentOfPeriapsis : degrees[angular]{unit = deg}
/meanAnomalyAtEpoch : degrees[angular]{unit = deg}
/eccentricity : Real
/semiMajorAxis : distance[km]{unit = kilometer}
semiMinorAxis : distance[km]{unit = kilometer}
RAAN : degrees[angular]{unit = deg}
percentTimeInEclipse : Real
maxBetaAngle : degrees[angular]{unit = deg}
numberOfPlanes : Integer
velocity : speed[km/s]{unit = km/s}
/period : time[minutes]{unit = minutes}
trueAnomaly : degrees[angular]{unit = deg}
/rangeToEarth : distance[km]{unit = kilometer}
«satisfy»
«System_Requirement» Orbit Determination
Id = "SR-7.5"
Text = "The system shall have a orbital period of less than 100 minutes."

Figure 17. SADSat Orbit Determination

The most impressive story from this class was the demonstration of new instructors taking over the class with seamless model handoff using DLE and the SVRA. On day 3, a new instructor that had never seen the model was able to access the instructor SADSat model through the Teamwork Cloud in DLE and learn about the mission we were modeling within 5 minutes during lunch. He did this by exploring the standardized organization of behavior, requirement, and structure diagrams as described by the SVRA. Then, he was able to lead the class in modeling additional structural features, including internal block diagrams. Finally, he was able to commit the model changes to the Teamwork Cloud for the next instructor. The next instructor was able to follow up on day 4 to learn about the model the same way, then lead the class in adding an orbital analysis to the model. The students greatly appreciated seeing the teamwork environment in action and wanted to implement the same methods for their projects after returning from the class.

Enterprise Model

The AFRL/RV Enterprise Model has a very similar purpose to the Operational Training Systems Enterprise Systems Model from Reed, 2019. The AFRL/RV Enterprise Model is a collection of all the mission models in one for enterprise level analyses, large scale source of truth baseline information of missions, and a means for translating common components for reuse in the Component Library. The enterprise analyses are primarily used to find capability gaps and test capability models for future spaceflight experiments. Imagine it as the sandbox to test communications, orbital dynamics, and payload performance in comparison to past and present AFRL space systems. Additionally, it is a source of information about previously modeled AFRL missions for use in decision making. If a specific wiring connector was causing minor issues across multiple missions, the data trend would be accessible in one place, then engineering replacements could be assessed across a broad range of designs and noted for future use in the Component Library. Finally, any non-proprietary model elements from any model are assessed in the Enterprise Model based on reusability, commonality, and modeled quality for inclusion into the Component Library. Each component added can then be accessed by updating the “payload” project usage in the SVRA.

CONCLUSION

The SVRA acts as the hub of integration and updates across the enterprise of AFRL/RV mission models thanks to the ever-evolving Component Library of model elements and the SysML Style Guide. Various AFRL/RV program models, originating from the SVRA, are reviewed and adopted into the Enterprise System Model where engineers run analyses on the collection of verified models, and common components can be moved to the Component Library. These Component Library updates are then immediately usable, if the update is locally accepted, in any model built on the SVRA. DoD professionals gain extra benefit from the SVRA in training, where personnel learn how to use MBSE and SysML through models that they want to build. AFRL has also built an excellent engineering ecosystem in DLE that enables seamless use of MBSE and the Teamwork Cloud across disparate teams. The best example of the benefits of Teamwork Cloud were observed during the construction of the SADSat model when instructors could learn, build, and hand off the model without issue.

The MBSE processes and methods utilized in the AFRL/RV MBSE implementation are the result of many pathfinder efforts across the Department of Defense (DoD) which have led to much faster execution and a clear path forward, as demonstrated through this report. The snowballing effect of model elements being created, then shared and reused through the SVRA in the Teamwork Cloud results in more detailed and useful models to aid in development of space missions in a shorter timeline than the traditional, document-based design approach.

REFERENCES

Ayers, G. et al. (2020). Model Based Systems Engineering for Simulator Sustainment. https://www.xcdsystem.com/iitsec/proceedings/index.cfm?Year=2020&AbID=79409&CID=572

Farrell, L. (2020). A Reference Architecture for CubeSat Development. https://scholar.afit.edu/cgi/viewcontent.cgi?article=4233&context=etd

Kelly, S. (2021). A Reference Architecture for Rapid CubeSat Development. https://scholar.afit.edu/cgi/viewcontent.cgi?article=5951&context=etd

Reed, C. (2019). The Foothold in the War of Cognition: The Operational Training Infrastructure Enterprise System Model. https://www.xcdsystem.com/iitsec/proceedings/index.cfm?Year=2019&AbID=27714&CID=48