Dr. Strangemodel: Assessing MBSE in the US Air Force Simulator Common Architecture Requirements and Standards (SCARS) Initiative

HOW I LEARNED TO STOP WORRYING AND START LOVING THE MODEL

William C. Riggs
SAIC
Wright Patterson AFB, OH
William.Riggs.5.ctr@us.af.mil

George Ayers
United States Air Force
Wright-Patterson AFB, OH
George.Ayers@us.af.mil

Austin Abraham,
J. Aaron Doak,
Scott M. Bailey
Tangram Flex, Inc. and CACI,
Dayton, OH
Austin.Abraham.4.ctr@us.af.mil,
Joseph.Doak.2.ctr@us.af.mil,
Scott.Bailey.31.ctr@us.af.mil

ABOUT THE AUTHORS

William C. Riggs is a Systems Engineering Manager with Science Applications International Corporation (SAIC) in support of SCARS. Prior to his current position, Mr. Riggs has been actively involved in a range of modeling and simulation systems engineering efforts over the past thirty two years, involving analysis of system architectures, system of system test and integration, and Live Virtual Constructive (LVC) simulation technology, Mr. Riggs holds a Master of Science in Technology Management from the Johns Hopkins Whiting School of Engineering, as well as Master's and Bachelor's degrees from Georgetown and Ohio State University.

George Ayers is an engineer working for the Air Force Materiel Command, Life Cycle Management Center (AFLCMC), Agile Combat Support Directorate, Simulators Program Office (WNS). He came to the Simulators Program Office after 13 years in the Air Force test community where he was a senior rocket sled test project manager. Mr. Ayers is the WNS lead for Model Based Systems Engineering (MBSE) in support of the Simulator Common Architecture Requirements and Standards (SCARS) Sustainment Initiative. Mr. Ayers holds degrees from Arizona State University, Texas A&M University, and Air University.

Joseph Aaron Doak is the contract team lead for MBSE on the Air Force Modeling and Simulation Support Services (AFMS3) 2.0 in support of the Simulators Program Office. Mr. Doak has a Master's degree in Industrial & Human Factors Engineering from Wright State University. He has seven years of systems engineering experience within the automotive industry, has attained a green belt in Six Sigma, and is a member of INCOSE.

Austin Abraham is a senior engineer on the AFMS3 2.0 MBSE team in support of AFLCMC/WNS. Mr. Abraham has a Bachelor's degree in Aerospace Engineering from the University of Colorado in Boulder. He is a member of INCOSE whose professional interests lie in aeronautical design and systems engineering.

Dr. Scott M. Bailey is a SCARS Engineer assigned to the SCARS Integration Support Integrated Product Team (IPT) engineering team. Mr. Bailey earned his Doctor of Philosophy in Information Technology from Capella University. He earned a Master of Military Art and Science degree from the United States Army Command and General Staff College. He earned a Master of Science in Information Technology Management from Trident University International. He earned a Bachelor of General Studies degree from Ball State University. He served for 20 years in the United States Army Signal Corps and now works for CACI supporting the SCARS initiative.

ABSTRACT

The US Air Force Simulator Common Architecture Requirements and Standards (SCARS) Initiative is spearheading the employment of digital engineering principles for the design, development, sustainment, and upgrade of Air Force training systems. The SCARS digital engineering strategy includes the creation of an Operational Training and Test Infrastructure Enterprise System Model (OTTI-ESM) and a Government Reference Architecture (GRA) for simulators that describes required structures, interfaces, and behaviors in compliance with the WNS Model Based Systems Engineering (MBSE) style guide, based on the Systems Modeling Language (SysML). This style guide prescribes general rules for SysML models as well as specific conventions for engineering analysis, requirements tables, structure, and behavior diagrams.

Because multiple government, industry, and military stakeholders contribute to the OTTI-ESM through construction of training system models used by SCARS, a consistent approach to evaluating SysML products is essential to the achievement of consistent modularity, openness, and usability of MBSE artifacts. This paper discusses various methodologies used to assess MBSE models in a system of systems context, SCARS evaluation methodologies, and potential improvements in architectural metrics as applicable to evolving simulation architectures in a cyber-secure, cloud-based environment. Both qualitative and quantitative methodologies are discussed, including the use of the peer review processes employed by the SCARS Engineering Capabilities Board and the AFLCMC/WNS MBSE team in conjunction with the SCARS Prime contractor, Affiliates (organizations that develop or sustain training systems), and Partners (organizations that develop and sustain training systems' networks). The applicability of attribute-driven design principles to SCARS Reference Architecture development is also discussed in conjunction with accessible tools and metrics. This paper provides examples of the methodologies used, results achieved to date, lessons learned, and proposed enhancements for the future.

INTRODUCTION

The implementation of digital engineering (DE) represents both a set of challenges and opportunities for the Air Force training and simulation portfolio, together with the real-world systems represented in live, virtual, and constructive simulation environments. Among these challenges are the evolution of closed, proprietary training architectures into modular open systems that can reuse common system components, the implementation of virtualized simulation designs in local and distributed cloud environments, and comprehensive system modeling to support cybersecurity processes and standards. These challenges must be addressed in the context of a rapidly changing operational and technical environment. (Brown Jr., 2020) As captured in the Department of Defense Digital Engineering Strategy, "this digital engineering transformation is necessary to meet new threats, maintain overmatch, and leverage technology advancements." (Office of the Deputy Assistant Secretary of Defense for Systems Engineering, 2018, p. v)

The Department of Defense Digital Engineering Strategy articulates five goals: (1) formalize the development, integration, and use of models to inform program and enterprise decision making, (2) provide an enduring, authoritative source of truth, (3) incorporate technological innovation to improve the engineering practice, (4) establish a collaborative environment across stakeholders to perform activities, collaborate, and communicate across stakeholders, and (5) transform the culture and infrastructure to adopt and support digital engineering across the lifecycle. (Office of the Deputy Assistant Secretary of Defense for Systems Engineering, 2018, p. 4)

The onset of digital engineering involves a learning curve for many training systems engineers, both within government and industry. The architecture and design of many legacy training systems, like the aircraft platforms they represent, predates the evolution of Model Based Systems Engineering (MBSE) and the modeling languages that support it. Frequently, existing training system support contracts specify legacy formats for simulation system and software specifications and requirements, architecture and design documents, and cybersecurity artifacts. By contrast, the US Air Force Operational Training Infrastructure (OTI) 2035 Flight Plan defines an Open Systems Architecture as "a standard that describes the layered hierarchical structure, configuration, or model of a communications or distributed data processing system that (1) enables system description, design, development, installation, operation, improvement, and maintenance to be performed at a given layer or layers in the hierarchical structure, (2) allows each layer to provide a set of accessible functions that can be controlled and used by the functions in the layer above it, (3) enables each layer to be implemented without affecting the implementation of other layers, and (4) allows the alteration of system performance by the modification of one or more layers without altering the existing equipment, procedures, and protocols at the remaining layers." (Goldfein, 2017)

The complexity of trainer designs can impose significant costs for MBSE implementation; at the same time, the use of proprietary hardware and software components may limit the system modeler's ability to represent the system at an adequate level of detail. Given the cost constraints associated with implementing MBSE, it becomes ever more important to define requirements for a good systems model that is fit for purpose and meets user needs in a digital engineering environment. Faced with budgetary realities, engineering managers are compelled to assess the value of MBSE to the enterprise, weighing benefits appreciated against costs incurred. The engineering challenge is to establish a mechanism not just to specify system models to be developed and integrated at the system, system of systems, and enterprise levels, but to characterize what makes for a “good” system model that fulfills the range of its intended uses. Experienced simulation developers may appreciate this modeling challenge, since it parallels the problem of creating valid and verifiable training simulations whose quality is likewise measurable for the purpose of assessing their value proposition against alternative approaches.

PREVIOUS EFFORTS

Many attempts to characterize the quality of MBSE models leverage existing software quality standards, including ISO/IEC 9126, ISO/ISE 25010, and ISO/IEC 25023. While ISO/IEC 25010 describes a hierarchy of quality characteristics (and subordinate characteristics), and ISO/IEC 25023 provides a methodology for quality measures and functions, there is no single standard for MBSE model quality measurement. Various frameworks have been proposed to bridge the gap between relatively well-defined software engineering practices and system modeling. Few if any of these have been implemented in the context of simulation and training system architecture and engineering. (Abran A., 2008)

While not all the quality characteristics described in ISO/IEC 25023 are relevant to the assessment of MBSE models, the following attributes and associated metrics have been utilized in similar assessments of open system models and their referents, including manned-unmanned teaming architectures (Garrett, 2019):

- **Usability** answers the question “Is the model or system usable?” Quality metrics relating to usability include the ability to recognize appropriateness of a product, learnability, user interface characteristics, and user error protection. (ISO/IEC 25023, 2016, pp. 14-19)
- **Modularity** answers the question “Can I change the model or system without redoing it?” Measures of modularity describe “the degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.” (ISO/IEC 25023, 2016, p. 25)
- **Openness** answers the question “Is the model or system accessible to the user and can it be shared?” Measures of openness include measures of interoperability, portability, and coexistence, including measures of the ability to exchange data, the adequacy of external interfaces and the ability to adapt system hardware and software to different environments. (ISO/IEC 25023, 2016, pp. 13-14, 28-29)
- **Functionality** answers the question, “Is the model fit for purpose, is it complete, and does it describe the system adequately?” Measures relating to functionality include functional completeness, correctness, and appropriateness. (ISO/IEC 25023, 2016, pp. 8-9)
- **Security** answers the question, “Does the model support ATO processes, and is the data in the model secure?” Measures relating to security include access controllability, data integrity, and authenticity. (ISO/IEC 25023, 2016, pp. 22-25)

One commonly used methodology for the implementation of measurable quality characteristics in system models is Attribute Driven Design (ADD), developed by the Carnegie Mellon University Software Engineering Institute (CMU/SEI). In ADD, the design process is driven by a prioritized set of quality attributes, which apply architectural tactics and patterns to model the system. CMU/SEI uses a combination of qualitative and quantitative methods to prioritize quality characteristics through the conduct of Quality Attribute Workshops to support the Architecture Tradeoff Analysis Method® (ATAM®). (Wojcik, et al., 2006)

ISO/IEC 25023 provides a set of metrics for the quality characteristics and sub-characteristics it describes. These metrics may be generic or specific. While generic metrics may be used in a wide range of situations, specific metrics are only applicable to a narrower set of users and needs. Metrics may be derived objectively or subjectively. While calculations for objective measures are repeatable, subjective measures are statistically non-repeatable, since they are derived from questionnaires and surveys, reflecting user attitudes and beliefs that may vary over time. A study of 122 ISO/IEC 25023 quality metrics by the University of Latvia indicated that seven (7) measures were entirely subjective. Twenty (20) of these measures were classified as objective, but may vary by the sample population of end users. Twenty-nine (29) measures were assessed as objective, but useful only for relative comparisons. Thus, 46 percent of the ISO/IEC 25023 quality metrics were either assessed as subjective, or required some degree of tailoring to be used effectively in software quality assessments. In terms of the quality characteristics described above, metrics relating to Security and Modularity were rated as “good.” The sub-characteristics associated with Openness ranged from “almost good” (Co-existence) to “good” (Interoperability). Two out of three sub-characteristics associated with Functionality were rated as “good, with Functional Completeness rated as “almost good.” Usability metrics demonstrated the greatest variation, with Appropriateness Recognizability rated as “moderate” and User Interface Aesthetics rated as “weak.” (Arnicane, Borzovs, & Nesaule-Erina, 2022) While this assessment is generally applicable to the measurement of MBSE model quality, assessment of MBSE models will necessitate a higher degree of tailoring to develop and implement useful measures.

WNS DIGITAL ENGINEERING OVERVIEW

The Air Force Materiel Command, Life Cycle Management Center, Agile Combat Support Directorate, Simulators Program Office (AFLCMC/WNS or simply WNS) is responsible for the acquisition, operation, and sustainment of thousands of simulators, of varying degrees of realism, distributed around the world. To further complicate matters, simulators are often acquired and used independently of aircraft. Consequently, while the Air Force acquires new simulators every year, some training systems are 40 years old. While simulators utilize constantly improving technology, not all systems are digital. In response to the need to manage the simulator portfolio more efficiently, WNS has decided to use Digital Engineering (specifically MBSE) to guide the creation of an Operational Test and Training Infrastructure (OTTI) Enterprise System Model (ESM).

OPERATIONAL TEST AND TRAINING INFRASTRUCTURE ENTERPRISE SYSTEM MODEL

The purpose of the OTTI-ESM is to manage the enterprise, permitting WNS to take advantage of economies of scale by identifying commonly used components. Information within the OTTI-ESM enables an enhanced cybersecurity posture through more rigorous compliance with risk management controls. Relevant information can be accessed by information security organizations and authorizing officials as requested to assess security risks. The OTTI-ESM supports efforts to conduct live-virtual-constructive training by identifying necessary interfaces and information. Additionally, it will be used to identify simulator architectural elements, supporting efforts to answer enterprise level questions in a matter of hours that take weeks to answer using legacy systems engineering artifacts.

Given the scale of the enterprise, the OTTI-ESM is unlikely to be completed in a timely manner if it is modeled by one organization. To offset this issue, the system models populating the OTTI-ESM are created by the individual training simulator programs. These program models are periodically delivered and integrated into the OTTI-ESM per individual program schedules. Models delivered to the government are snapshots of the state of the systems at the time of delivery. The intent is for individual programs within WNS to use the program model for their management and planning activities. It is expected that WNS contractors will use the models as systems engineering tools to enable the management of disparate parts of their program as well as to document the current state of the systems for which they are responsible. While the early emphasis for model construction is flight training devices, the intent is for all training systems managed through WNS to be modeled and included in the OTTI-ESM.

GOVERNMENT REFERENCE ARCHITECTURE

The Government Reference Architecture (GRA) created under the auspices of the SCARS Initiative is a high-level definition of generic training systems architectures. It is envisioned primarily as a means of identifying common architectural elements of training systems, particularly flight simulators. It may be used as a reference model for new simulator acquisition and as a tool to plan migration of common software functions from the training device to new architecture. The GRA development follows the familiar systems engineering decomposition process, starting with simulator requirements and functions. These are further decomposed to systems and subsystems before actual components are identified. This modeling effort incorporates SCARS standards and requirements and associates them with appropriate parts of the model. Model content is determined by the SCARS Engineering Capability Board (SECB). Subject matter experts from various fields identify candidate subsystems and requirements for major systems. As appropriate, relevant requirements are included for each subsystem and a notional interface diagram is constructed. As each major system architecture is brought to a useful state, the entire SECB reviews and makes comments on the proposed system architecture. Once a system is approved, it is integrated into an overall reference architecture model constructed and maintained by the SCARS Prime contractor. The Government SCARS team continues to guide discussions regarding the GRA and its content.

Of particular interest to the SCARS initiative is the identification of the software required for these systems and components to function, especially those that can be used by different platform training devices. This is an implementation of the Modular Open Systems Approach (MOSA), where the standardized interfaces of a system are known and controlled so that the system may be replaced by alternative solutions as needs and technology change. This promotes adaptability of the larger system to circumstances as they change over time without being tied to specific sources, reducing sustainment costs. For example, if a particular targeting pod may be mounted on several platforms, the software emulation of that pod would be a candidate for inclusion in a common software library for use on simulators for those platforms. If this software can be containerized and used in several simulators, it is expected that the enterprise will see a decrease in costs associated with concurrency redevelopment each time the pod is upgraded.

The top-level systems for the GRA have been identified and members of the SECB are working to determine the subsystems and generalized components that belong to each. After the initial identification of GRA contents has been completed, the GRA will move to an iterative cycle of system-by-system modeling and review. It is expected that additional items that should be included will be identified as the system model goes through this process. Since it is unlikely that the GRA will never need to be modified to reflect current circumstances, model scope creep may become an issue. To manage this, model development uses an incremental development approach. This approach (1) defines a set of items to be included in each increment, (2) works to complete the model to address the requirements of a specified increment, then (3) continues to the next increment.

The GRA must be of sufficient quality to achieve its purpose. Proposed GRA model quality metrics include usability, modularity, functionality, and openness. To be suitable, a high-quality GRA must have sufficient detail and scope. An architecture model with too little detail will not be able to achieve the purpose of the GRA. Too much detail would be unnecessary and counterproductive. Too narrow a scope for the initial architecture would reduce the SECB's ability to identify common items and too broad a scope would make it difficult to complete in a reasonable timeframe. While flight training devices are of primary interest, other training systems within the Air Force simulators portfolio will be represented in the GRA. Constructing the GRA so that major subsystems can be added or modified without disrupting the rest of the model is necessary to achieve this goal. If the architecture is to be used to acquire new training systems, it needs to be easy for those unfamiliar with it to understand. As with all models created for AFLCMC/WNS, the GRA should comply with the WNS MBSE style guide.

THE WNS MBSE STYLE GUIDE

The style guide was created in order to converge all models for WNS to a common appearance and organization. Originally the style guide was written out as a plain text document and was just a list of "dos and don'ts" for the modelers within the WNS MBSE team. The team then found out that there was a need for something similar among other early adopters of MBSE around the DoD. It was at this time that the OTTI-ESM came to life. In order to allow the style guide to be directly accessible from it, it was turned into a model in and of itself with the rules being written out in requirement blocks. This allows for future development in the way of automatically validating models attached to the style guide.

The WNS MBSE team uses a comment resolution matrix that allows members of the team to prioritize, select, and resolve comments that have been made about style guide content, either by our team or other modeling teams. This process enables the continuous improvement of the style guide so it can maintain its place as the standard for stylizing MBSE models within AFLCMC and beyond. A host of supporting material is also being developed to be distributed along with the style guide to anyone who wishes to bid on a contract where the deliverables require models. This supporting material will be part of a bidders' library for use in contract actions and will be including items such as a global reference library for modeling, a program model outline template, and an example program model among other things.

Future Style Guide rules and examples are planned to include more complex modeling topics related to system sustainment. Outreach efforts for style guide suggestions will include all SCARS affiliates as they begin to model their programs. This should allow us to make a better style guide that is well suited for all WNS programs. Ways to extend the style guide for use within other portions of AFLCMC and the Air Force at large will be explored. The team will investigate adding automatic validation capabilities into the style guide for certain rules so that teams using it to model do not have to manually review models for style guide compliance. If teams can quickly validate their models against the style guide, their models will be more easily integrated into the OTTI-ESM and understanding between different program models will be a given.

EXAMPLE MODEL

An example model has been created to better showcase what a well-constructed, style guide compliant model would look like. The WNS MBSE team devised a fictional program for the sole purpose of creating this example model. Using a fictional program allowed the team to make design decisions that are solely based on creating a good example model, rather than creating a good training device. This strategy allows for more flexibility that expedited the modeling process and eliminated the need for cross-referencing documentation obtained via the Affiliates.

While the style guide contains every rule to be followed and is where programs should go to understand how they should be building out their models, the example model shows how those rules should be implemented in practice. Where the Style Guide has many rules in place that are meant to convey how specific things shall be modeled, the example model serves as a place to show off more complex modeling techniques that cannot be easily described and understood in a set of rules. Therefore, there is a need for both the style guide as well as its supporting materials such as the modeling rules and procedure sheets or the example model.

TEAM MODELING

The WNS MBSE Modeling team consists of seven members utilizing a collaborative environment that hosts Cameo Teamwork Cloud (TWC). This allows the entire team to access and modify different sections of the model simultaneously. The team has often practiced paired modeling, similar to paired programing, where it has been found that modeling in small groups of two or three can be more efficient; especially when modeling a technically difficult scenario where creative solutions are still being identified. Two main methods of separating the modeling efforts were explored: section-based and task-based. Section-based modeling was first attempted, where one section of the model was assigned to one modeler or a small team of modelers. This method creates clear lanes to help avoid workflow congestion, but it also isolates the work in a way that reduces modeling style cohesiveness. The team switched to task-based modeling, where all modeling tasks were documented and tracked in an excel sheet. As a task is completed, each modeler or small team of modelers claims the next task on the list to be modeled. This has increased model cohesiveness, as the style for modeling a certain item remains common regardless of what section of the model it is located. However, it has also caused additional difficulties due to multiple modelers sometimes needing to access the same areas of the model for different tasks. One would often plan to log into the collaborative environment with plans to work on a Task B, only to discover that someone had either already started modeling Task B or started Task A that required them to lock elements needed to model Task B. To alleviate these inefficiencies, a simple team chat has been created in Teams to help communicate when a task has been worked on and what model elements have been locked.

One of the main lessons learned from creating the example model with a team of several people is that good communication is essential. Cameo Teamwork Cloud has a plugin called Cameo Collaborator that manages model review comments, via an external HTML web view, for both modelers and model reviewers. However, for those that do not have that plugin, or are using a different tool, it is vital to set up additional communication channels, both internal and external to the modeling tool. An example of utilizing elements provided by Cameo includes «problem» comments (that is, a comment that is stereotyped as “problem”), which has been found to be much more effective than standard notes in Word or OneNote. During a model review, «problem» comments are created on the diagram and linked to the relevant item(s) in the model. A table of each training device’s «problem» comments was created in the WNS MBSE Team’s viewpoint package within the Analysis section of the model. This allowed any modeler or model reviewer a quick overview of the outstanding items to be fixed for the model.

SCARS MODEL USE

The SCARS initiative uses models to communicate reference architectures to bidders and to evaluate their proposed technical solutions. WNS engineers develop SCARS infrastructure requirements, and a reference model is constructed. During this process, deficiencies and questions that need SME clarification are identified, requirements are revised, and the model is modified in an iterative fashion. This process also helps to refine the Performance Work Statement. The reference architecture model is provided as SysML models and DODAF views with other materials to offerors.

In turn, the offerors submit models that reflect their solution. The models are evaluated and, in cases where a model is found to have issues in a certain area, the team can quickly review other models to see if they have issues in the same area.

After contract award, the SCARS Prime contractor submits a systems model documenting its design, tracing requirements to subsystems, components, and behavioral diagrams. The system-subsystem specification and other requirements are imported from the contractor's requirements management software into a table within the model. Reviewers use this table to assess how these requirements were satisfied, and whether the solution is acceptable. Interfaces between different systems as well as components are specified and the concept of how capabilities are distributed across multiple systems is implemented. Government engineers can quickly evaluate the design against the requirements, and provide approval in time for devices to be procured and installed.

Appropriately constructed system models will be able to communicate many technical details and can be used as a basis for analysis. The SCARS Prime contractor currently combines two contract deliverables with the system model (system/subsystem design document and interface description document), and it is anticipated that more will be transitioned from the traditional document submission process to models (for example, the requirements traceability and verification matrix or the software version description). Work is underway to identify how this can be done for other programs within WNS and across the USAF.

LESSONS LEARNED

REVIEWING MODELS

Self-Guided Tour

When first receiving a new model to review, it is often useful to peruse the model to gain an understanding of how it was constructed prior to attempting any actual evaluation of the model. This can often be a pleasant and enlightening experience, where the model reviewers gain valuable insights into the minds of fellow modelers. In other instances, this can prove confusing and frustrating as the model reviewers try to make sense of the maze of thought and half-implemented modeling practices of a poorly managed modeling team. Either way it is often useful to get an initial understanding of the model.

Reviewing Requirements - Line-By-Line.

When evaluating a model to ensure that it meets requirements, there are only so many options available; unfortunately, they all involve reading through and evaluating each requirement. While this is often a mundane and tedious task, it can be less painful if the model is properly developed (with requirements mapped to system elements) and easy to navigate.

Assisted Walkthrough

Whether using a "Self-Guided Tour" of a model or going through "Line-By-Line," having one of the creators of the model, or at least someone that is familiar with it to assist in the review is extremely helpful. This is similar to the difference between taking a self-guided tour of the model, which will allow you to see some pretty pictures, and having a tour guide will reveal an entirely new level of understanding and appreciation of the artifacts.

Relation Maps

When assessing the architecture of a system using MBSE there are a few main metrics that can be tracked to assist in this assessment. One of the easiest ways to see if a model is accomplishing its job is by creating a relation map of the entire model and then doing a model gap analysis using this map. This will allow the reviewer to get a high-level view of how all the elements within the system relate to each other in different ways. A good model will have the proper relationships flowing from each class of elements down from the most abstract needs of the stakeholders. If a relation map lines up with what one would expect from a system engineering perspective, then the system has most likely been developed correctly.

Understanding the model navigation

Good model navigation is a critical factor to the usability of the model. If a model is difficult to navigate, there will be an exorbitant amount of time wasted trying to find relevant information. If a model is too difficult to navigate then it becomes detrimental to the systems engineering efforts and will not be used. For these reasons, it is extremely important that careful consideration be given to the organization of the model from the beginning of the model creation throughout its development. Mapping out the containment tree with the Relation Map Diagrams is also a good way to get a good understanding of how easy a model's containment tree is to comprehend. An example of a Relation Map Diagram is depicted in Figure 1.

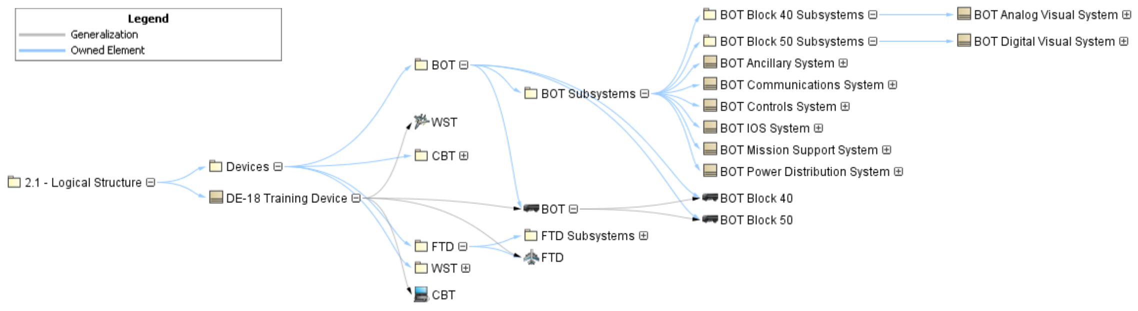

Figure 1: Example Relation Map Diagram

Beyond having a clean and easy to understand containment tree, being able to navigate using diagrams is important when there will be model users involved that are not necessarily SysML experts. This is especially true when the organization is standing up their MBSE process as a new initiative, when no one is familiar with MBSE models, and when the organization has a high turnover rate which will result in frequently bringing in new members that are unfamiliar with MBSE. For these reasons, model organization should follow a decomposition structure similar to the actual system(s). At the top of this organization is a Top-Level Title Page Package Diagram that essentially acts as a table of contents for the model with additional relevant program details included as needed, such as Security Classification and Distribution Statements. From this top level throughout the entire model, each package, block, and part property should link to a relevant diagram. In many cases, there may be multiple diagrams that would make sense for a single element to navigate to, in which case diagram icons should be grouped in a consistent manner to facilitate model navigation. For example, a Stakeholder package contains several diagrams to satisfy that stakeholder's needs; an additional package diagram could be created in this instance that is used for displaying shortcuts to all the stakeholder's diagrams. Alternatively, a single diagram may be the main navigation point from a diagram element, but a section of the diagram can be reserved specifically for displaying links to other relevant diagrams. In addition to being able to navigate from the top of the model to the lowest detailed diagrams, users will also need to navigate back. The consistent employment of diagram navigation links in every diagram allows users to get back to previous sections of the model.

Comment Resolution Matrix (CRM) Tracking

To best accommodate model reviews, ideally both the model creators and model reviewers will have access to the same collaborative environment. The model reviewers either utilize comments directly in the model, or access a special linked export of the model. This allows real-time oversight of model development, minimizing potential confusion which reduces technical difficulties that can be associated with regular deliveries of a model.

If getting the model reviewers into the same collaborative environment with the model creators is not an option, then a separate CRM can be helpful to track problems and resolutions when reviewing a model. This simple table can be passed back and forth between the reviewer and the modelers, utilizing version numbers between each cycle to trace the shortfalls of the model requirements. However, there are some potential disadvantages to this method. If there are several iterations of CRMs, then tracing an original change request and how the relevant portion of the model and comment history has evolved over time can become cumbersome.

MEASUREMENTS OF MODEL GOODNESS

The main concern of a systems engineer usually comes down to the satisfaction of the system requirements. This can be partially seen from building out the relation map discussed in the previous paragraph, but in order to be sure to see the whole picture, one would build out a requirements table or a satisfaction matrix. These two diagrams are a quick way to see if all system requirements have been met of if there are still requirements that need attention. This allows one to get quick idea of how successful the design of the system has been. The requirements table provides even more information than a matrix can because it gives the satisfaction as well as the context to that satisfaction and supporting information such as verification methods and relevant elements that can also be traced from the requirement. Ultimately, this proves the requirements table to be a very useful diagram for assessing a system's design and implementation.

Another metric of interest for model reviewers is that of model completeness. This topic has been investigated by many organizations over the last few years. Notably, the International Counsel on Systems Engineering (INCOSE) has collaborated with international modeling organizations to create a Model-Based Capabilities Matrix. (Hale & Hoheb, 2020) As the title implies, this matrix is made to assess an organization's model-based capabilities. The matrix resides in an Excel spreadsheet, and is comprised of rows separated into different groups of high-level goals, which are then broken down into specific capabilities of MBSE or DE. These capabilities are then further specified into different stages across the separate columns of the spreadsheet. Another section within the Excel workbook breaks down the capabilities in a role-based manner. The specific matrix was built mostly around the needs of DoD customers and it addresses those needs reasonably well, but the same format of matrix could be used for any program and could be specified more on a systems model basis and less on digital engineering at large. That would provide a program with a good, pseudo-quantitative, high-level view of how far the model has progressed with respect to the initial program goals. If a program sees that the stages and capabilities within the matrix are slowly being checked off then it can safely assume that its model is heading toward being "more complete."

In contrast to this approach to assessing model completeness, the WNS MBSE team developed the "WNS Stage Definition" document, which is a bulleted checklist of the different stages of model development for Air Force Simulator systems in sustainment. This checklist organizes five stages that go into detail about what is expected at each step in the model development process. The first stage focuses on general model organization and building an overall package structure. The second stage is centered on the building a logical structure down to the systems and subsystems associated with each training device, and the physical systems down to their locations, sites, and training devices. The third stage describes the bulk of the work in building out detailed logical connections along with the rest of the physical structure and real-world system detail. The fourth stage involves adding functionality to the structure by satisfying requirements, building out use cases and activities, and creating viewpoints for all stakeholders to the model. Finally, the fifth stage centers on sustaining and updating the model itself to keep it current with the real-world system, adding more functionality, and using it to create standardized reports for the system. The need for this stage definition document was recognized after multiple discussions about the needs of the different teams within WNS that will be creating and using the models. The definition of what constitutes a complete model is not the same for all WNS programs. This makes defining a uniform standard for model completeness difficult because what one team may see as complete would be barely useable for another team. In the end, the metrics by which one calls a model mature or complete are heavily dependent on the perspective of the stakeholder that will be using the model.

CONCLUSION AND PATH FORWARD

While substantial progress has been made developing and implementing a methodology and framework for assessing MBSE models, more work needs to be done to formalize metrics and implement MBSE model assessments that leverage both qualitative and quantitative measures. Recognized metrics defined in ISO/IEC 25023 provide descriptive measures that support the evaluation of system models. Experience with the Cameo model comparison tool has helped support model assessments, enabling model users to assess the maturity of MBSE models as well as model stability. The implementation of requirements satisfaction matrices and tables provides a useful tool that enables the model user to assess key quality attributes.

In the short term, the attribute driven design methodology should be more closely integrated with MBSE model development, enabling the definition of measurable quality characteristics for each model during requirements definition. Standardization efforts should be pursued at the working group level. The SCARS Engineering Capability

Board (SECB) provides a venue for both Government Reference Architecture standards and applicable quality measures to be defined across the broader community. The Simulation Interoperability Standards Organization (SISO) offers a useful mechanism for exchanging lessons learned and emerging techniques for digital engineering of M&S systems as these efforts mature.

Numerous questions remain as to the scope and purpose of MBSE modeling in support of the WNS portfolio. Reverse engineering of legacy simulation systems remains a daunting challenge, and the cost of comprehensive implementation of MBSE across the WNS portfolio remains an active concern. As these models are developed, both the government and contractors are developing a better understanding MBSE implementation in support of the WNS program portfolio.

The implementation of the WNS Style Guide has provided a firm basis for standardizing MBSE model content consistent with intended usage, as well as enabling useful dialogue between model producers and users. While the WNS Style Guide has been implemented with good effect, demonstrating the path forward with AFLCMC, its use may be expanded to include other Air Force organizations involved in the design, development, acquisition, and sustainment of Air Force training simulators. This would require formalization of the WNS Style Guide as a comprehensive standard for MBSE models. Future versions of the WNS Style Guide may be extended to support this broader community of practice.

REFERENCES

Arnicane, V., Borzovs, J., & Nesaule-Erina, A. (2022, May 12). Do We Really Know How to Measure Software Quality?

Brown Jr., C. Q. (2020, August). Accellerate Change or Lose. Washington D.C.: Office of the Chief of Staff, United States Air Force.

Drake, D., Hinton, M., Lutz, R., Morse, K., & Saunders, R. (2021, June 4). (U/FOUO) OSA Metrics Summary. *Johns Hopkins Applied Physics Laboratory*.

Garrett, C. (2019). Using Open Architectures in an Agile Acquisition Process: Integration Architectures and Models.

Goldfein, G. D. (2017, September 5). Air Force Operational Training Infrastructure 2035 Flight Plan.

Hale, J., & Hoheb, A. (2020, Jan 6). INCOSE Model-Based Capabilities Matrix and User's Guide. (Version 1). INCOSE.

ISO/IEC 25023. (2016, June 15). Systems and software engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — Measurement of System and Software Product Quality. *International Standards Organization ISO/IEC 25023*, pp. 8-29.

Office of the Deputy Assistant Secretary of Defense for Systems Engineering. (2018). *Digital Engineering Strategy.* Washington, D.C.

Office of the DoD CIO. (2010). *DoD Reference Architecture Description.* Washington D.C.: Office of the Assistant Secretary of Defense, Networks and Information Infrastructure.

Roper, W. (2020, October 7). *Dicula Nulla Est.*

Wojcik, R., Bachmann, f., Bass, L., Clements, P., Merson, P., Nord, R., & Wood, B. (2006). Attribute-Driven Design (ADD), Version 2.0. *Software Engineering Institute*, p. 31.