Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations | 1. Thesis and Dissertation Collection, all items

2022-09

ASSESSMENT OF MODEL CONVERSION FROM GENESYS TO MAGIC SYSTEM OF SYSTEMS ARCHITECT FOR MODEL-BASED SYSTEMS ENGINEERING INTEROPERABILITY

Donovan, Michael C.

Monterey, CA; Naval Postgraduate School

http://hdl.handle.net/10945/71058

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ASSESSMENT OF MODEL CONVERSION FROM GENESYS TO MAGIC SYSTEM OF SYSTEMS ARCHITECT FOR MODEL-BASED SYSTEMS ENGINEERING INTEROPERABILITY

by

Michael C. Donovan

September 2022

Co-Advisors: Paul T. Beery
Ronald E. Giachetti

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2022	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
ASSESSMENT OF MODEL CONVERSION FROM GENESYS TO MAGIC SYSTEM OF SYSTEMS ARCHITECT FOR MODEL-BASED SYSTEMS ENGINEERING INTEROPERABILITY

5. FUNDING NUMBERS

6. AUTHOR(S) Michael C. Donovan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

This thesis investigates whether the information contained in a Vitech Genesys model can retain its informational accuracy after conversion into a Dassault Systemes' Magic System of Systems Architect (MSOSA) model. The thesis uses a sample system model in Vitech that implements the system definition language (SDL) and converts it to MSOSA, which uses the systems modeling language (SysML). The study reviewed conversion methods available to the user and converted a Genesys model to an MSOSA model using the only available method, Excel. The study then assessed the converted model and outlined any post-migration remediation.

The results of this thesis demonstrate that the currently available methods are feasible but inefficient, as only 34% of the entities and 9% of the relationships transferred successfully during the experiment. Genesys can output tabular data that represents system model entities and relationships; however, the MSOSA import function was unable to correctly import entities that had one-to-many relationships with other entities. Consequently, the user must perform manual manipulation during the conversion process. Furthermore, ontological differences between the tools prevented the complete import of behavioral data, since many SDL entities map to more than one SysML entity.

Based on the results, this thesis recommends pursuing an extensible markup language–based software solution for Genesys and MSOSA and developing a formal Navy and Marine Corps ontology.

14. SUBJECT TERMS
MBSE, models, GENESYS, Vitech, Cameo, Magic Systems of Systems Architect, Core, modeling, model interoperability, semantics, ontology, SysML, SDL, conversion, modeling tools, tool interoperability, model transformation, D/SET

15. NUMBER OF PAGES
93

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

ASSESSMENT OF MODEL CONVERSION FROM GENESYS TO MAGIC SYSTEM OF SYSTEMS ARCHITECT FOR MODEL-BASED SYSTEMS ENGINEERING INTEROPERABILITY

Michael C. Donovan
Civilian, Department of the Navy
BSME, Drexel University, 2017

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 2022

Approved by: Paul T. Beery
Co-Advisor

Ronald E. Giachetti
Co-Advisor

Oleg A. Yakimenko
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis investigates whether the information contained in a Vitech Genesys model can retain its informational accuracy after conversion into a Dassault Systemes' Magic System of Systems Architect (MSOSA) model. The thesis uses a sample system model in Vitech that implements the system definition language (SDL) and converts it to MSOSA, which uses the systems modeling language (SysML). The study reviewed conversion methods available to the user and converted a Genesys model to an MSOSA model using the only available method, Excel. The study then assessed the converted model and outlined any post-migration remediation.

The results of this thesis demonstrate that the currently available methods are feasible but inefficient, as only 34% of the entities and 9% of the relationships transferred successfully during the experiment. Genesys can output tabular data that represents system model entities and relationships; however, the MSOSA import function was unable to correctly import entities that had one-to-many relationships with other entities. Consequently, the user must perform manual manipulation during the conversion process. Furthermore, ontological differences between the tools prevented the complete import of behavioral data, since many SDL entities map to more than one SysML entity.

Based on the results, this thesis recommends pursuing an extensible markup language–based software solution for Genesys and MSOSA and developing a formal Navy and Marine Corps ontology.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

BDD	block definition diagram
BPMN	business process model and notation
D/SET	Digital/Systems Engineering Transformation
DISA	Defense Information Systems Agency
DM2	DoDAF MetaModel
DOD	Department of Defense
DoDAF	Department of Defense Architecture Framework
DON	Department of the Navy
eMASS	Enterprise Mission Assurance Support Service
EMF	eclipse modeling framework
ERA	entity-relationship-attribute
FMU	functional mock-up unit
GOTS	government off-the-shelf
IME	Integrated Modeling Environment
INCOSE	International Council on Systems Engineering
IoIF	Interoperability and Integration Framework
LML	Life Cycle Modeling Language
MBSAP	Model-based System Architecture Process
MBSE	model-based systems engineering
MoDAF	British Ministry of Defence Architecture Framework
MOF	Meta-Object Facility
MSOSA	Magic Systems of Systems Architect
NAVAIR	Naval Air Systems Command
NAVWAR	Naval Information Systems Command
NAWCADLKE	Naval Air Warfare Center Aircraft Division Lakehurst
NSWC Crane	Naval Surface Warfare Center - Crane Division
OMG	Object Management Group
OOSEM	Object Oriented Systems Engineering Method
OWL	web ontology language
SCXML	state chart XML

SDL	System Definition Language
SE	Systems Engineering
SET	Systems Engineering Transformation
SoI	system of interest
SYSCOM	systems command
SysML	Systems Modeling Language
UAF	Unified Architecture Framework
UML	Unified Modeling Language
WG	working group
XMI	XML metadata interchange
XML	Extensible Markup Language
XPDL	XML for Process Definition Language

EXECUTIVE SUMMARY

The releases of the DOD's *Digital Engineering Strategy* (2018) and the *United States Navy and Marine Corps Digital Systems Engineering Transformation Strategy* (2020) outline the strategies to formally incorporate MBSE and digital engineering approaches in acquisition processes for the DOD and Navy and Marine Corps, respectively. Two of the objectives detailed in the *United States Navy and Marine Corps Digital Systems Engineering Transformation Strategy* (2020) are to "formalize the development, integration, and use of models" and to "provide an enduring authoritative knowledge source." The Naval Digital/Systems Engineering Transformation (D/SET) Working Group (WG) is the Department of the Navy's (DON) cross-systems command (SYSCOM) action team that was established with its primary purpose being "to accelerate implementation of Digital/Systems Engineering Transformation … and increase digital engineering collaboration across SYSCOMs" (Johnson 2020). The D/SET WG has begun implementing the Naval Integrated Modeling Environment (IME), a modeling environment that employs SysML and NoMagic's Cameo Systems Modeler that is available to all Navy and Marine Corps systems commands.

Throughout the Navy, different commands are likely using different tools. Yet, they must share information. In pursuit of enabling model interoperability, the purpose of this thesis is to explore user-available methods for converting a Vitech Genesys system model to a Dassault Systemes Magic System of Systems Architect (MSOSA) model, and assess if a model created in Genesys can retain its correctness and semantic content after the conversion process. Various commands employ Vitech's products to support their programs, while the Naval IME employs Cameo Systems Modeler, which this thesis considers to be analogous to MSOSA. These modeling tools employ different modeling languages; Genesys uses the system definition language (SDL) with support for SysML diagram types, while Cameo Systems Modeler and MSOSA use SysML without any pre-defined underlying ontology. The tools also employ different ontologies, which is how a system model conveys meaning through its semantic content.

To assess the conversion process, this thesis first identified the accepted file formats for data import and export in each tool, as well as any built-in program extensions supported by each tool. Second, an attempt was made to convert a system model originating in Genesys (using SDL) to a comparable system model in MSOSA (using SysML). The conversion methodology leveraged in the second step was informed and guided by the accepted file formats identified in the first step. Any obstacles encountered during the conversion process were also recorded, and finally, the informational content of the resultant MSOSA model was assessed.

The thesis limits the conversion approach to what is currently available to the typical user of MBSE tools. Hence, the thesis does not write any computer programs in an attempt to automate the conversion process. Since Genesys provides a means to export tabular data via Vitech's Excel Connector tool, and MSOSA supports the import of tabular data, this thesis explored the use of Microsoft Excel as an intermediary to exchange information between the two tools.

Figure 1 defines the methodology used in this study's attempt to convert a system model from Genesys to MSOSA via Microsoft Excel. Figure 1 aligns to each of the three SysML pillars in scope: physical, functional, and requirements. Within Figure 1, Steps 1–3 convert the physical entities from Genesys into MSOSA. Steps 4–6 then convert the functional entities from Genesys into MSOSA, and finally, Steps 7–9 convert the requirements from Genesys into MSOSA. This study utilized the Genesys-provided Geospatial Library (GSL) sample model as the system of interest (SoI).

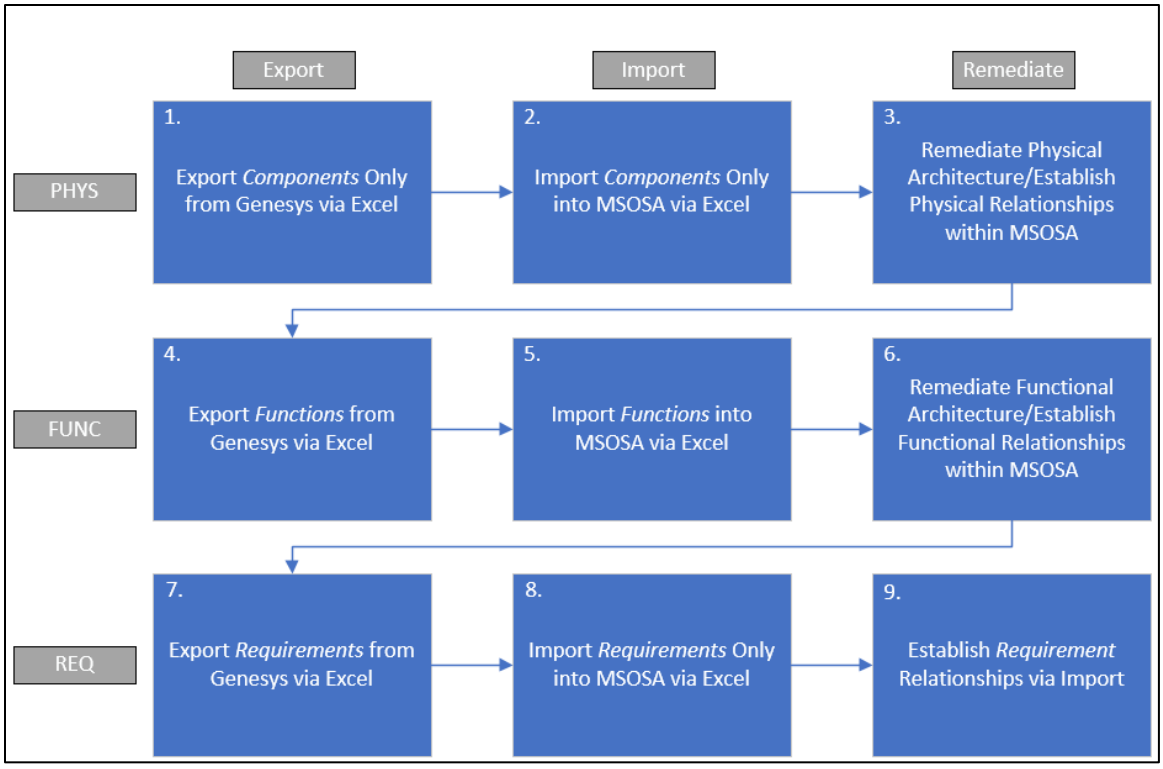


Figure 1. Process Summary for Genesys Export and MSOSA Import via MS Excel

The thesis was successful in replicating parts of the original Genesys GSL model in MSOSA, but it took a lot of user action to do the transfer and conversion. The experimentation involving the GSL system model revealed limitations in data transfer between the two tools. Table 1 summarizes the limitations as observed by this study.

Table 1. Limitations Exchanging Data from Genesys to MSOSA Using the Genesys Excel Connector and MSOSA Mapping Tool

Observation	Limitation	Description
Data Duplication or Blank Data	One-to-One Mapping	Unable to correctly transfer data when there are one-to-many relationships between elements. Either MSOSA creates multiple copies of the same element, or MSOSA creates blank entities. In either case, the user must go into the model and correct the mapping.
Multiple Imports Required	One-to-One Mapping	Multiple imports are required and in a particular order in order to correctly transfer relationships between entities.
Order of Import	One-to-One Mapping	The order of the import is of significance during the migration process. A user may incorrectly omit relationship information if both entities for any single relationship do not already exist in MSOSA.
Importing Behavioral Modeling Data	Ontological Differences	There is a major modeling difference between how SDL and SysML model items and the interfaces between components to capture behavior. SDL defines item elements used as inputs/outputs/triggers, whereas SysML uses object and control flows, in addition to various combinations of object nodes, central buffer nodes, pins, etc. (each of which also have specialized types). The ontological mapping between SDL and SysML for system behavior therefore proved difficult to reproduce in MSOSA.
Post Migration Remediation	Inefficiency	The overall process was inefficient and time intensive. As every level of the data exchange process required a degree of manual remediation in MSOSA, the primary advantage of the Excel method seems to be only in establishing new entities.

This experiment was successful in transferring only 144 of 423 entities and 116 of 1,304 relationships from Genesys to MSOSA. The simple GSL sample model contained less than 2,000 entities and relationships and took the user approximately seven hours to adequately transfer only 17% of the original model's overall contents. The experiment revealed how Genesys and MSOSA have limited data transfer capabilities through the Excel connector method. Transfer of requirement data was successful, and the Excel connector method facilitates transfer of entities well, but it does not perform well in exchanging relationship information. The primary obstacle in the Excel method,

demonstrated by Table 1, was the one-to-one interpretation implicit to the MSOSA mapping tool, as most models involving any degree of complexity will have one-to-many relationships. This effectively restricted the import method to the exchange of entities only. Imports of the physical and functional architectures were successful in MSOSA but required manual manipulation in the MSOSA tool after the import process to be reflective of the original model. This would require a modeling effort or project to recreate a representative GSL model in MSOSA, which severely undercuts the value of this method.

With these results and using the Levels of Conceptual Interoperability Model (LCIM) (Tolk and Muguira 2003), this thesis assesses the interoperability between Genesys and MSOSA at Level 0. Documentation exists independently for the two tools for exchange of data, but there does not exist documentation specific to the interface between the tools since one does not exist. Since the only feasible method identified by this thesis to transfer data is through Excel, and each tool handles this connection differently, there is no effective means for interoperability. Therefore, this thesis arrives at an LCIM Level of 0 for this use-case.

To support a digital thread with this study's approach, the Navy would have to invest in developing automated programs for extracting, transferring, and loading model data so it can piece together models from various sources. The methodology this study presents could theoretically be automated in a fashion like IBM Cloud's extract, transform, and load process, which "combines data from multiple data sources into a single, consistent data store that is loaded into a data warehouse or other target system," (IBM Cloud Education 2020) however this approach can be brittle as changes in either the source or target tool can break the automated process. It is likely more efficient at the extensible markup language (XML) data level, like Sandia's demonstration (Carroll et al. 2021), instead of using Excel as an intermediary and attempting to automate it.

Until the community establishes a universally available technical Genesys-MSOSA solution and standard Navy and Marine Corps ontology, commands utilizing Genesys will continue to diverge from the Naval IME modeling path that their sister SYSCOMS are adopting. This conclusion may force Navy commands, such as NAWCADLKE, to continue modeling outside of the Naval IME and will hinder any future model federation

goals. For these reasons, recommended future work includes the establishment of a SysML ontology for the Navy and Marine Corps to universally leverage and describe their systems, and the publication of a Genesys-MSOSA software solution that allows users to import XML data from Genesys to MSOSA. Since MSOSA was assumed to be synonymous with the Cameo Systems Modeler tool employed by the Naval IME, it is also recommended that the software solution be explored for specific use with Genesys and Cameo Systems Modeler.

References

Carroll, Ed, Carlos Tafoya, Jonathan Compton, Akinli Cengiz, and Jason Jarosz. 2021. "Retaining Systems Engineering Model Meaning Through Transformation: Demo 2." Technical Report SAND2021-2143. Albuquerque, NM: Sandia National Laboratories.

Department of Defense. 2018. Department of Defense Digital Engineering Strategy. Washington, DC: Office of the Deputy Assistant Secretary of Defense for Systems Engineering. https://ac.cto.mil/wp-content/uploads/2019/06/2018-Digital-Engineering-Strategy_Approved_PrintVersion.pdf.

Department of the Navy. 2020. United States Navy and Marine Corps Digital Engineering Transformation Strategy. Washington, DC: Department of the Navy. https://ac.cto.mil/wp-content/uploads/2019/06/2018-Digital-Engineering-Strategy_Approved_PrintVersion.pdf.

Genesys to Cameo Conversion Process. 2020. Crane, IN: NSWC Crane Division.

Johnson, Joan L. 2020. "Digital/Systems Engineering Transformation Working Group Charter." Official Memorandum. Washington, DC: Department of Defense.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my parents for their endless support and patience throughout my studies, as well as my many friends who have been more than understanding whenever I needed to focus on school in lieu of attending occasions. My family and friends will always have my gratitude for their persistent support and encouragement through my studies.

I would also like to thank my advisors, Professor Giachetti and Professor Beery, for their enduring guidance and tolerance during my thesis efforts.

Finally, I would like to also thank Carolyn Holguin for her recommendation that placed me in this NPS program and provided me the opportunity to author this thesis. I also want to extend my thanks to Michael Brazinski as well for his feedback while I brainstormed thesis topics.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This chapter establishes the motivation for this thesis. It also provides background information on model-based systems engineering (MBSE), the importance of semantics and ontologies, and how these topics are relevant to MBSE efforts. Furthermore, it summarizes the Dassault Systemes Magic Systems of Systems Architect (MSOSA) and Vitech Genesys modeling tools and their respective languages. Finally, this chapter details the research problem, the benefits of this research, and the overall organization of this document.

A. BACKGROUND

1. Model-Based Systems Engineering

The International Council on Systems Engineering (INCOSE) defines MBSE as "the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" (INCOSE 2007). Adoption of MBSE has become increasingly widespread in recent decades as industries attempt to develop and maintain systems of increasing complexity while adequately conveying this complexity in meaningful ways to their stakeholders (INCOSE 2007).

Giachetti and Vaneman (2021) identify the following six components required to implement MBSE successfully: a modeling language, a schema (i.e., ontology, meta-model), model-based processes, presentation framework, MBSE tools, and a knowledgeable and trained workforce. Many distinct variants in performing MBSE exist that can compose each of these MBSE elements. Table 1 illustrates some of the many options for each component. Table 1 omits the workforce training component as the organization must identify the other five components before any training may occur. Note that there is no intentional relationship between each column contained in Table 1.

Table 1. A Non-exhaustive List of MBSE Component Variations

Modeling Languages	Schemas	Model-based Processes	Presentation Frameworks	MBSE Tools
System Modeling Language (SysML)	User-defined (infinite variations by extension)	Object Oriented Systems Engineering Method (OOSEM)	Department of Defense Architecture Framework (DoDAF)	MagicDraw's Cameo/ Dassault Systemes' Magic System of Systems Architect (MSOSA)
System Definition Language (SDL)	Vitech's Core and Genesys Meta-Model	Vitech's STRATA	British Ministry of Defence Architecture Framework (MoDAF)	Vitech's Core and Genesys
Life cycle Modeling Language (LML)	DoDAF MetaModel (DM2)	Model-based System Architecture Process (MBSAP)	Zachman's Framework	IBM's Rhapsody
Unified Modeling Language (UML)	Meta-Object Facility (MOF)	NoMagic's Magic Grid	Unified Architecture Framework (UAF)	SPEC Innovations' Innoslate

While aspects of MBSE have progressed towards standardization, such as the MBSE languages, MBSE is still a growing field and continues to evolve (INCOSE 2007; 2022). Though some combinations would be infeasible (e.g., leveraging LML in Vitech Genesys), enumerating through Table 1 alone shows how a modeler could theoretically capture and present the same system in up to 16 different ways, yet all would represent the same information. The differences among these similar yet different theoretical models stress the importance of semantics and ontology and shift the focus from *what* to model to *how* to model.

2. Semantics, Ontologies, and Meta-Models

The Merriam-Webster dictionary formally defines the term semantics as "the study of meanings" (n.d). In the context of MBSE:

> There is a need for the semantics of the model to be formalized, which is accomplished with a metamodel. The reason for formal semantics is to avoid problems inherent in the communication of ambiguous representations, to enable computer interpretation of models, and to exchange models between systems engineering teams. The meta-model provides a standardized and consistent terminology resulting in a shared and precise interpretation of a model. (Giachetti 2015, 255).

Giachetti and Vaneman (2021) state that "semantics specify the interpretation of the constructs as well as what they mean when combined in the model." In other words, the semantics of a model convey the predefined language and meaning of the information captured by the model. For example, tool vendors often define a model's semantics in subject-predicate-object triples. Defining this semantic structure, including directionality, establishes the formal ontology that the system model uses to convey information (Carroll et al. 2021). That is, the modeling tool specifies or uses an ontology and the modeler uses the modeling tool's ontology, which includes standardized terms and their meaning to describe the system-of-interest in the model space (Vaneman 2022). Figure 1 is an example of a simple ontology that uses the subject-predicate-object triple construct.

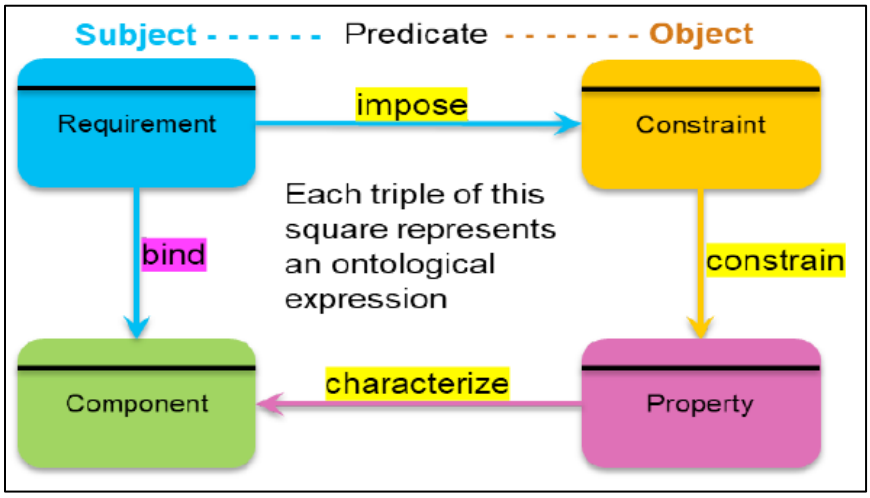


Figure 1. Sample Ontology. Source: Murdock and Carroll (2021).

Practitioners often use the term "ontology" interchangeably with the terms "meta-model" and "schema." Slight differences exist among these terms, which is also an exercise in semantics. The term "meta-model" further considers the syntax of the model, both abstract and concrete. The abstract syntax "specifies the concepts, relationships, and rules" governing a modeling language, whereas the concrete syntax "specifies the notation within the modeling language" (Giachetti and Vaneman 2021). Figure 2 illustrates the relationships among these terms with a system and its system model.

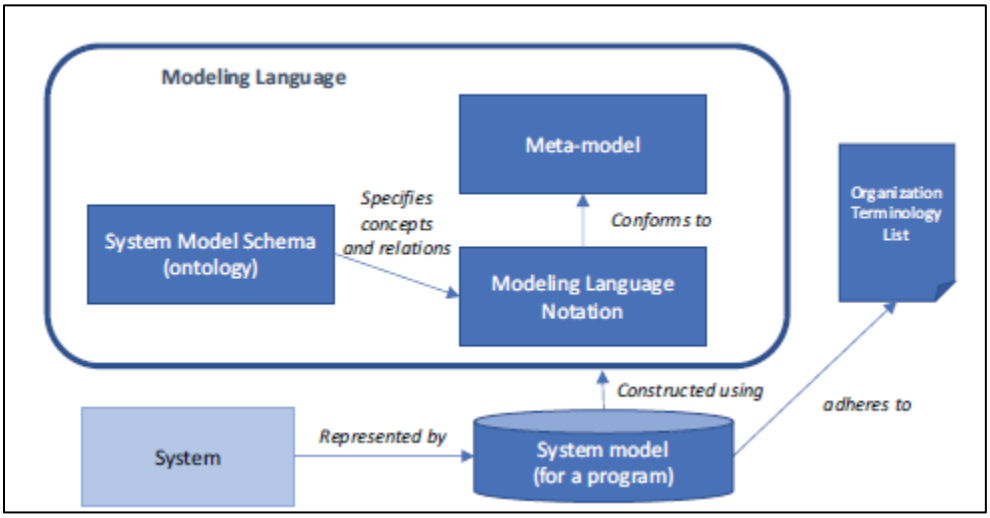


Figure 2. Concepts and their Relationships. Source: Giachetti and Vaneman (2021).

This thesis refers to the terms "ontology," "schema," and "meta-model" as interchangeable and synonymous to avoid confusion regarding these nuances.

3. Vitech Genesys Tool and Language

The Vitech Corporation, a Zuken company, developed and manages the Genesys tool. The Genesys tool, the successor to Vitech's Core modeling tool, implements the System Definition Language (SDL) modeling language. Vitech ("Key Concepts" n.d.) describes SDL as "a formal, structured language that avoids ambiguity inherent in using common English to define or specify a system." The SDL is a formal language with a robust taxonomy conveying consistent semantic meaning. Figure 3 shows the meta-model

of the SDL. Vitech's SDL ("Key Concepts" n.d.) uses an approach based on a subject-predicate-object triple, which they call an "entity-relationship-attribute (ERA)" language with obvious reference to the entity-relationship model of relational databases. A portion of the SDL in Figure 3 shows a *component* element *performs* a *function* element (an entity, a relationship, and an entity, i.e., the object, respectively). Each element then has attributes that characterize the element, such as a title, description, unique identifier, etc.

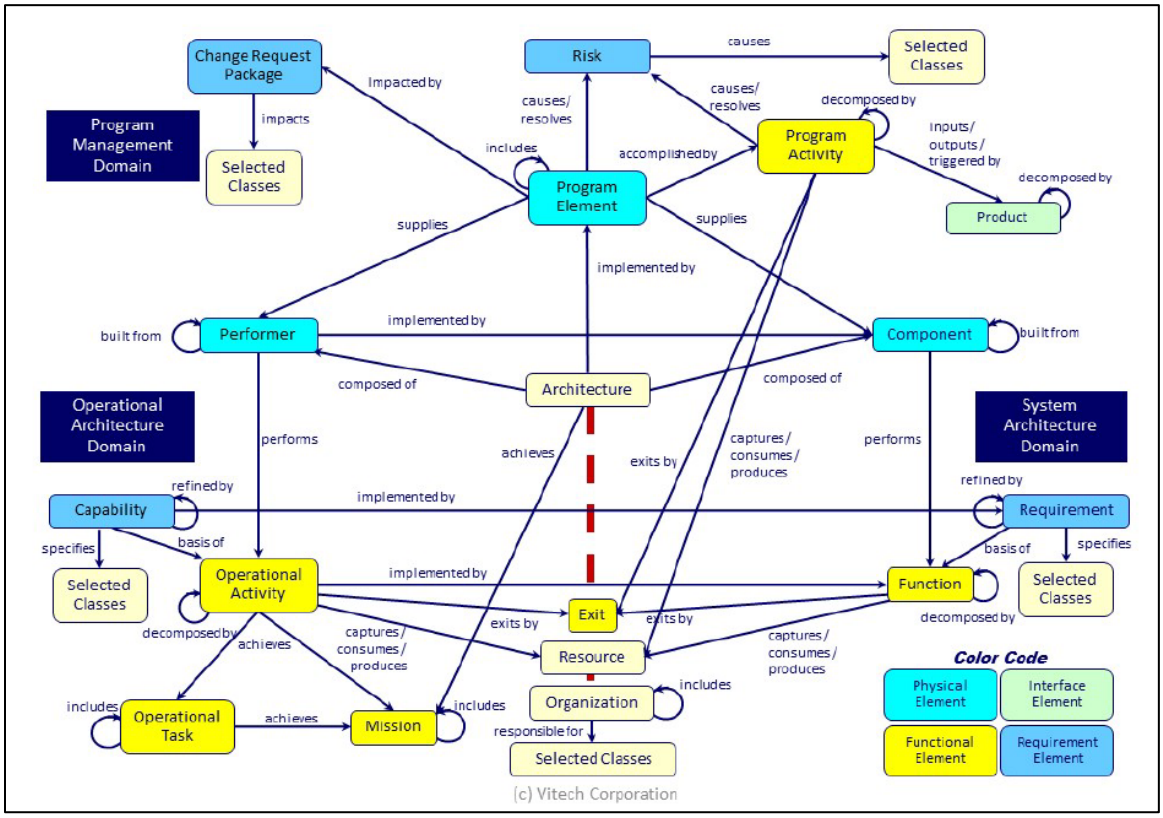


Figure 3. Vitech Genesys Schema. Source: "Genesys 4.1 Architecture Definition Guide" (2016).

Vitech's Genesys tool supports Systems Modeling Language (SysML) views, but it was not originally built around SysML. This report further discusses SysML in the Dassault Systemes Magic Systems of Systems Architect Tool and Language section.

4. Dassault Systemes Magic Systems of Systems Architect Tool and Language

The Magic Systems of Systems Architect (MSOSA) tool is a successor of the NoMagic Cameo suite of tools. Dassault Systemes acquired NoMagic and rebranded the tools in 2018. This thesis interchangeably refers to both NoMagic's Cameo Systems Modeler tool and the MSOSA tool because these tools have the same implementation of SysML and provide the same general functionality.

MSOSA uses multiple modeling languages based on the Unified Modeling Language (UML). MSOSA does not support Vitech's SDL. The focus of this thesis is on MSOSA's implementation of SysML. SysML is an extension of the UML. The Object Management Group (OMG) manages both UML and SysML. The OMG describes SysML as:

> A general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities. In particular, the language provides graphical representations with a semantic foundation for modeling system requirements, behavior, structure, and parametrics, which is used to integrate with other engineering analysis models. "What Is SysML?" (n.d.).

SysML is tailorable to any specific domain. SysML lacks a schema to define terms relevant to the systems engineering (SE) domain. The OMG has intentionally designed SysML to be extensible, which requires the user or organization to define a schema. For instance, the *block* is the most basic unit of structure in SysML. A *block* can be "stereotyped" as different classes of blocks as the user sees fit. A generic example is a user defining a *block* as a fluid that contains specific user-defined properties, such as density and viscosity. Other blocks can then be stereotyped as a fluid, such as water or oil, and these *blocks* will inherit the properties defined by a fluid with default or user-defined values (Holt and Perry 2019). For these reasons, the user-defined ontology created to support a given domain can differ from those in other domains, or even the same domain, despite having similar semantic meaning.

Figure 4 depicts the SysML diagram types, which fall into three main categories: behavior, requirement, and structure diagrams.

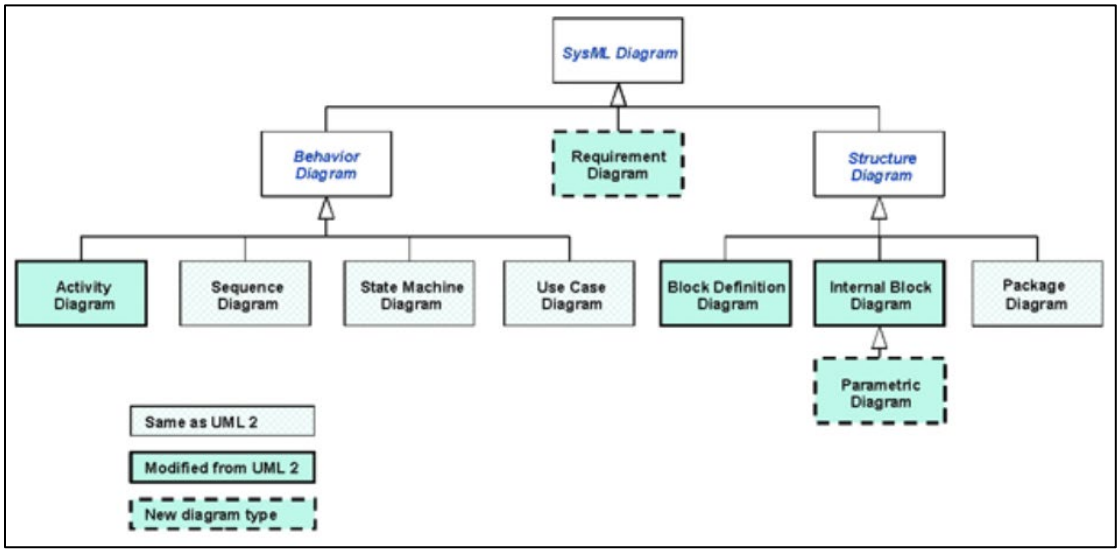


Figure 4. SysML Diagram Types. Source: What is SysML? (n.d.).

Behavior diagrams describe the system's functions and how the system performs under various conditions. Requirement diagrams describe the requirements the system must meet and their traceability. Structure diagrams describe how system elements are connected to form the system. Structure diagrams include both internal and external connections to the system. The parametric diagram also exists as a type of structure diagram, in which variables can be defined with relationships to support model simulation and analysis.

B. MOTIVATION

In years past, the Department of Defense (DOD) acquisition community has implemented some degree of MBSE and realized some of its benefits. However, "MBSE process and methods [have] generally [been] practiced in an ad hoc manner and not integrated into the overall systems engineering processes" (INCOSE 2007, 15). Moreover, many commands are pursuing different MBSE approaches using different tools, which may create challenges when those commands must work together and share model data.

Some of these challenges include redundant modeling efforts and obstacles in conducting simulation and analyses of system models. They occur when system models cannot connect or share data.

For instance, using a generic example of a car implies external interfaces with other systems, such as a gas pump for refueling. A modeler needs to define the car with sufficient detail in their modeling tool but cannot do so without having also defined the gas pump as well, to some extent. Despite the modeler not having control over the gas pump, they must still account for its interface, at the least. This is often simplified through interface standards (e.g., all gas pumps use the same length and diameter nozzle), but if no interface standard exists, then more details regarding the external system must be captured to sufficiently define the car model. This leads to model duplication, as the model element(s) representing the gas pump are captured by both the vehicle designer and the gas pump designer. This same scenario can occur in military systems as well. One organization, such as Boeing, may capture an aircraft's tailhook in their aircraft model, while another organization, such as NAWCAD, may capture the arresting cables on an aircraft carrier in their model. There exists a physical interface between the tailhook and arresting cable components during operations. Hence, to fully define their system, Boeing may capture details of the arresting cables in their model, and NAWCAD may capture details of the tailhook in their model. Neither organization has control over both components, yet a modeler needs to capture the interface, and therefore some details of the external system, to correctly design their own systems. In both examples, two different models contain redundant model elements, each of which took time to define and neither entity can exert direct control over the external elements. If the models could instead be connected and simply share data, this redundancy would be mitigated. The redundancy in model elements may also present risk, as modelers may incorrectly define the elements they do not control or may not actively manage them and could eventually lead to design error.

Another symptom includes the inhibition of simulation and analysis of system of systems or platform behavior due to discrete, unconnected models. A model's scope constrains any given simulation of that model. Therefore, a simulation that depends upon external systems must assume the interactions at the interfaces as ideal, which is not always

the case in the real world. Aggregating, or federating, the models of each system to generate the complete end-to-end behavior provides more detailed insight into operational risks and is more likely to elicit identification of emergent behavior. Maintaining discrete and unconnected models inhibits the simulation of the end-to-end functionality and deters its analysis.

To address these areas, the DOD has published the *Digital Engineering Strategy* (2018), and the Department of the Navy (DON) (2020) has published the *United States Navy and Marine Corps Digital Systems Engineering Transformation Strategy*. Two objectives of the Navy's Transformation Strategy are to "formalize the development, integration, and use of models" and to "provide an enduring authoritative knowledge source" (2020).

In response to the strategic guidance, some Navy systems commands (SYSCOMs) have recently adopted common modeling practices to assist MBSE interoperability. For example, the Naval Information Warfare Systems Command (NAVWAR) released NAVWARINST 5401.9 in October 2021. Among other things, NAVWARINST 5401.9 calls upon NAVWAR to identify and certify a standard tool suite and to maintain, update, and enforce a NAVWAR Data Schema and a NAVWAR Integrated Dictionary. It also calls upon NAVWAR to "develop and maintain MBSE models leveraging at a minimum the Systems Modeling Language and, if appropriate, the Unified Architecture Framework Profile" (Department of the Navy 2021). In essence, NAVWAR is recommending the entire Navy agrees upon and uses a single set of methods, tools, and languages. This is one approach to addressing MBSE interoperability concerns but may not be the best approach. This type of mandate would constrain the Navy to a single tool and language, reducing its flexibility to adapt to new methods, tools, or languages, and would dramatically increase tool vendor dependence.

Another response to the Navy's Transformation Strategy, NAVAIR stood up the Systems Engineering Transformation (SET) Team to promote MBSE and Digital Engineering, and the Navy stood up the Naval Digital/Systems Engineering Transformation (D/SET) Working Group (WG). The SET Team is NAVAIR's community of practice tasked with the implementation of the DOD and Navy digital strategies. It has

similar goals as the greater D/SET WG, whose purpose "is to accelerate implementation of Digital/Systems Engineering Transformation … and increase digital engineering collaboration across SYSCOMs" (Johnson 2020). In 2019, the SET Team selected the SysML as the standard modeling language and endorsed NoMagic's Cameo Systems Modeler as the preferred modeling tool (Moschler 2019, 18). The D/SET WG has also begun implementation of the Naval Integrated Modeling Environment (IME), a modeling environment available to all SYSCOMs that employs SysML and NoMagic's Cameo Systems Modeler.

Selection of a common modeling language and tool is an approach for the Navy to avoid MBSE interoperability problems. However, adopting the SysML and Cameo Systems Modeler challenges organizations and programs that already implement MBSE using different languages and/or tools in a different way. These organizations and programs may have models that: are created using different MBSE processes, conform to different schemas, utilize a different presentation framework, or are captured in different modeling languages or tools.

Some commands had previously embraced Vitech's modeling tools when MBSE gained momentum in the early 2000s. Other commands had settled on IBM's Rhapsody when that seemed like it would become a de facto standard tool in the Navy. With the Navy's selection of SysML and NoMagic's Cameo System Modeler for the IME in 2019, those commands are now on a divergent MBSE implementation path from the other Navy SYSCOMS. Therefore, commands are likely to continue to encounter interoperability hurdles when working with other commands and moving toward a single digital thread.

One may extend this scenario beyond the Navy into the DOD and even industry. While the Navy might settle on a single tool and language, other DOD components might choose something different causing interoperability issues with joint programs. Moreover, there is no guarantee that industry will use the same tools and languages. Finally, even if all the commands used the same tools and language, there are many variations which could cause other (but more likely minor) interoperability issues. MBSE tool interoperability is an issue the SE community must address in order to realize the greater vision of a digital thread.

Throughout the Navy, different commands are likely using different tools. Yet, they must share information. The various enumerations in and beyond those shown in Table 1 generate barriers to achieving interoperability among SE models. For instance, referring to Figure 3, Vitech uses *Operational Activities* to form the *basis of* a *capability*. However, MSOSA with SysML defines only generic *Activities*, which may represent any activity. It is unknown if one can distinguish these variations in SDL after translation to SysML (e.g., can a user distinguish an SDL *Operational Activity* from an SDL system *function* after translation to SysML?), especially in the absence of a previously defined ontology that would stereotype *activities* in SysML. Obstacles such as this present concerns for true interoperability among SE models. This research attempts to understand areas the SE community must address to enable such interoperability.

C. RESEARCH QUESTION

This research investigates whether a user can feasibly export the information contained in a Vitech model based on SDL and import the model into an MSOSA model based on SysML while retaining model correctness and semantic content. The thesis limits the approach to what is currently available to the typical user of MBSE tools. Hence, the thesis does not write any computer programs in an attempt to automate the conversion process.

D. BENEFITS OF STUDY

The results of this research will inform organizations, such as NAWCADLKE, of the interoperability concerns that may prevent the successful conversion of models between different MBSE tools and/or languages. Addressing MBSE interoperability concerns will contributes to the DON's (2020) pursuit of digital engineering.

E. ORGANIZATION

Chapter II of this thesis contains a literature review that identifies various works related to model interoperability and integration and the means to successfully implement an authoritative source of truth using a digital engineering thread. Chapter III of this thesis details the experimental setup and the methodology for model conversion, and the

experiment execution and results found using this methodology and experimental setup. Chapter IV of this thesis then presents the results of the experiment and reports on the conclusions drawn from those results.

II. LITERATURE REVIEW

This chapter contains a literature review on the interoperability of system models, especially in the context of MBSE. This thesis presents research on the current state of model interoperability and how it will impact the ultimate pursuit of a singular digital engineering thread. Two sections compose this chapter: how interoperability plays a role in the context of MBSE and related work regarding MBSE interoperability.

A. INTEROPERABILITY IN THE CONTEXT OF MBSE

1. Limiting Factors of Interoperability Among Models

Tolk et al. (2012) identify two fundamental observations with modeling and simulation (M&S) that inhibit interoperability: simplification and abstraction. Tolk et al. (2012) state that all models are inherent "simplifications and abstractions of reality in order to support a certain task."

Simplification occurs when a modeler omits various aspects of a real object's definition, which defines a model's scope. For example, suppose the system of interest (SoI) is an information system on a naval platform. In that case, a modeler will only include information up to the boundary of the SoI, including interfaces and first-order contextual definitions such as the external system(s) that make up an interface. The modeler does not model the entire naval platform, only areas that concern the SoI. Therefore, the modeler simplifies the SoI to support its specific tasks.

Abstraction occurs when models are created "with different structures and resolutions" (Tolk et al. 2012). For example, suppose again that the SoI is an information system. In this case, the SoI comprises hardware, software, and firmware. One modeler may explicitly capture all three as independent configuration items. Another modeler may abstract firmware as an attribute of hardware and only capture two configuration items. Both models would accurately represent the real system. However, the models would differ in resolution due to abstraction.

Furthermore, Tolk et al. (2012) also describe the impact of the cognitive aspect in M&S: "In order to conceptualize the observation, the observer needs to have an internal model he can map this observation to. A physician will see more in an X-ray than a layman. An educated mechanic sees more in an engine than a novice." This cognitive aspect means a logistician's and a mechanical engineer's models may represent the same SoI, but still be entirely different. Both models would be informationally accurate and represent the same SoI in the real world. Nevertheless, they could implement disparate schemas and have completely different structures, both of which inhibit the interoperability of the two models despite them representing the same SoI. Hence, the components of MBSE will be dependent upon the domain of both the user and the SoI.

2. Progress Towards a Solution

Academia has made progress toward a solution for the MBSE interoperability hurdle. In pursuit of enabling an authoritative source of truth in the digital engineering thread, Bone et al. (2018) introduced the Interoperability and Integration Framework (IoIF) to achieve model interoperability and integration that is tool, language, and process agnostic.

The IoIF implements a software-intensive five-step process that is not yet automated. It is "envisioned to have two main functions: (1) data acquisition and aggregation; and (2) semantic query and reasoning that allows for consistency and completeness checking of the data" (Bone et al. 2018). The process leverages a sequence of file conversions, graphical mappings, data parsing, and decision logic. As such, it requires a high degree of competence in software engineering that is not familiar to practicing systems engineers and inhibits its overall implementation. Figure 5 depicts the IoIF and its five-step process, using a model created in MagicDraw in SysML as an example.

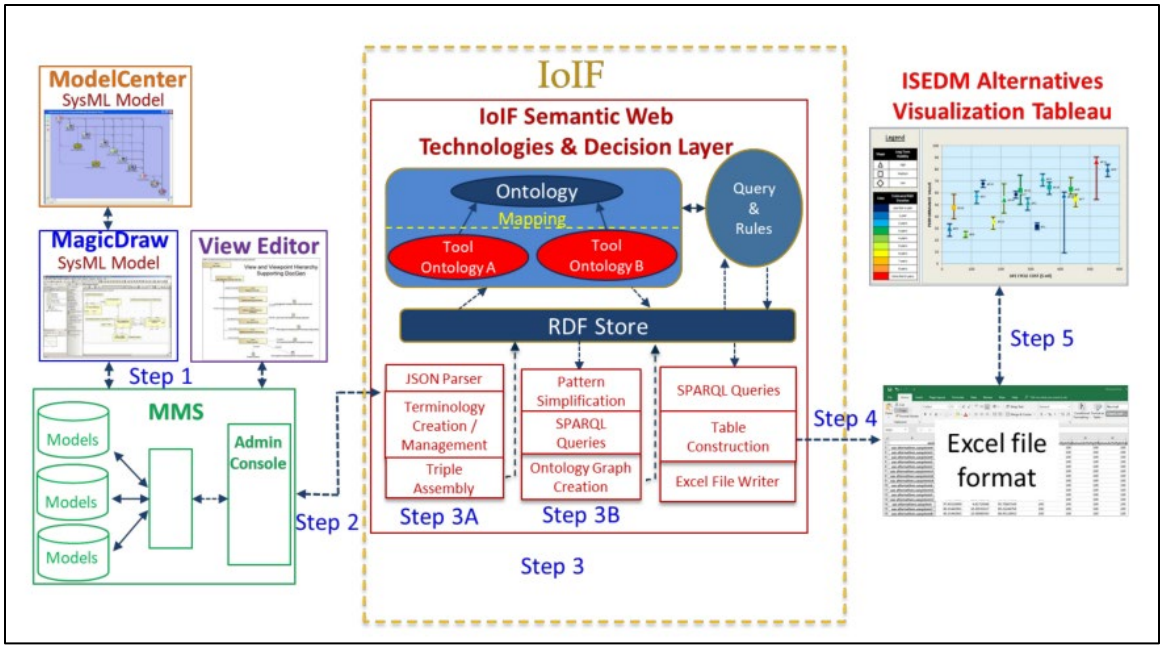


Figure 5. The Interoperability and Integration Framework (IoIF). Source: Bone et al. (2018).

3. Assessing MBSE Interoperability

The assessment of interoperability among MBSE models requires a methodical approach that is language and tool agnostic. The assessment must take account of, but not depend upon, the tool and language. This agnostic approach warrants an assessment at the conceptual level, for it is not interoperability between model instances in question but interoperability between their higher-level conceptual models and ontologies. The levels of conceptual interoperability model (LCIM) is introduced by Tolk and Muguira (2003), whom define five levels of conceptual interoperability as captured in Figure 6.

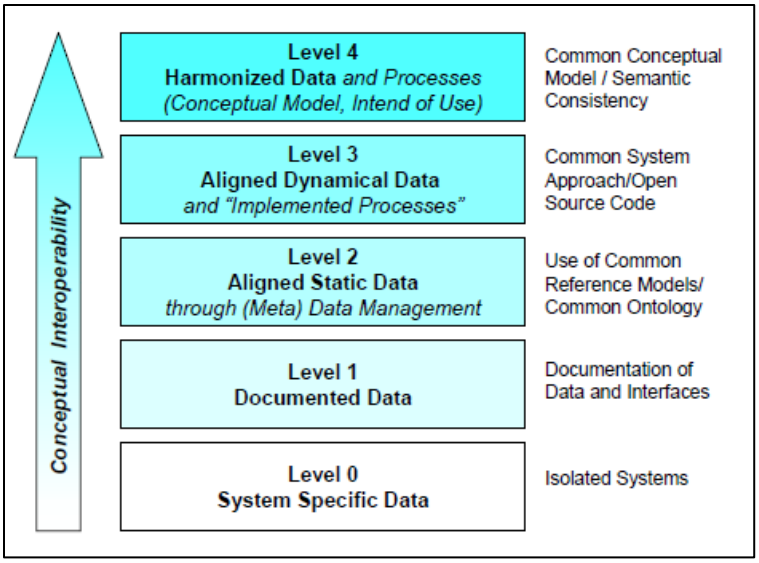


Figure 6. The Levels of Conceptual Interoperability. Source: Tolk and Muguira (2003).

The LCIM focuses on data to exchange and interface documentation to make a qualitative assessment. Tolk and Muguira (2003) summarize each of the five levels as the following:

- **Level 0 – System Specific Data:** No interoperability between two systems. Data is used within each system in a proprietary way with no sharing. The component (or application) is a black box.
- **Level 1 – Documented Data:** Data is documented using a common protocol… and is accessible via interfaces. The component is a black box with an interface.
- **Level 2 – Aligned Static Data:** Data is documented using a common reference model based on a common ontology, i.e., the meaning of the data is unambiguously described…The component is a black box with a standard interface.
- **Level 3 – Aligned Dynamic Data:** The use of the data within the federate/ component is well defined using standard software engineering methods such as UML. This shows the use of data within the otherwise unknown "black box behind the interface," also known as white box.
- **Level 4 – Harmonized Data:** Semantic connections between data that are not related concerning the execution code is made obvious by documenting the conceptual model underlying the component. It is not only a white box; because beyond the implemented parts of the concept the important relations that are NOT captured in the implementation are captured.

As one traverses towards higher levels of interoperability within the LCIM, model data becomes increasingly aligned, and documentation of that data is increasingly available and detailed. The LCIM (Tolk and Muguira 2003) defines "Harmonized Data" as being the most mature and occurs when two models have semantic and relational consistency. A simple example would be examining two Vitech Genesys models; Vitech predefines both model's ontologies and therefore achieves semantic and relational consistency at all levels. At the other end of the spectrum, the LCIM defines "System Specific Data" as being the least mature and occurs when there is no interoperability between two systems. The Defense Information Systems Agency's (DISA) Enterprise Mission Assurance Support Service (eMASS), which is a government off-the-shelf (GOTS) solution for integrated cybersecurity management, and Vitech's Genesys tool are an example of "System Specific Data," since neither tool can integrate with the other.

One may use the LCIM may to assess conceptual agreement between two models. It is analogous to a technology readiness level (TRL). Where a TRL can succinctly convey technology maturity, an LCIM assessment can succinctly convey interoperability maturity among two models.

B. RELATED WORK

There are numerous related works available regarding model interoperability. This literature review identifies three specific efforts highly relevant to this thesis. This section describes these efforts and the insights they provide. The first related work described in this section is a whitepaper from Naval Surface Warfare Center (NSWC) Crane on the Genesys to Cameo conversion process. The second is a demonstration and report from Sandia National Labs on retaining an SE model's meaning through transformation. The third and final related work described in this section is a software solution presented to the Naval D/SET community by the company SodiusWillert.

1. NSWC Crane — Genesys to Cameo Conversion Process

As of late 2021, NSWC Crane had led an effort to document the process of converting a model created in Genesys to a model in NoMagic Cameo System Modeler.

NSWC Crane developed a draft whitepaper to detail this process that was obtained as part of this research.

NSWC Crane's (*Genesys to Cameo Conversion Process* 2020) first step in converting Genesys to Cameo was to map element types and relationships from Genesys (the source tool) to the SysML language in Cameo System Modeler (the target tool). The mapping is a translation between the conceptual data models and ontologies of the two tools. This translation is one of the most significant steps since all subsequent modeling efforts will be predicated on this translation. NSWC Crane mapped common Genesys entities and relationships to SysML. However, NSWC Crane generalized (simplified) many Genesys elements to a generic *block* element in SysML, and over 50 Genesys relationships remained unmapped to a SysML counterpart.

Once NSWC Crane (*Genesys to Cameo Conversion Process* 2020) established the mapping, they used the only available export file format for the full Genesys model, a *.gsnx file, which is unique to the Genesys tool. They then renamed the file as a *.zip file and decompressed it. This step enables viewing of the folder structure, which separates model data from schema data, each containing files in the extensible markup language (XML) format. The XML file format is both human- and machine-readable.

The next step in NSWC Crane's (*Genesys to Cameo Conversion Process* 2020) process was to parse the XML files and generate data for element creation. To do so, NSWC Crane extended Cameo System Modeler by developing a plugin in the Eclipse Integrated Development Environment that enabled Cameo to read Genesys data. NSWC then used structured query language (SQL) statements to create SysML elements and their related data, all based on the previously defined mappings.

This conversion process elicits some valuable observations. First, omitting various element and relationship mappings implies that any originating data captured by those elements and relationships will have information lost during the conversion process. The target model will require further manual remediation to reestablish the applicable detail in SysML or remain incomplete. Second, an in-house development effort that yielded a Cameo plugin was the enabler for parsing the Genesys XML files. This plugin is

unavailable to the greater SE community; therefore, the SE community has yet to validate its scalable implementation. Until a plugin solution is available to the entire SE community, organizations with existing Genesys models must develop their own conversion methods. Finally, the white paper offers little-to-no documentation in validating the resultant model to assess informational consistency. The results focused on a user's navigational views and not the model's content, which leaves open the question "was something missed?"

2. Sandia National Laboratories Demonstration

In February 2021, Sandia National Laboratories (Carroll et al. 2021) published the report *Retaining Systems Engineering Model Meaning Through Transformation*. Sandia National Laboratories leveraged a similar yet different approach from NSWC Crane. Sandia's goal was to develop a "proof of concept that the meaning of a system model can be retained during transformation," which the authors assert is "the missing ingredient in effective systems model-to-model interoperability" (Carroll et al. 2021). Sandia's (Carroll et al. 2021) approach followed these steps in converting from Genesys to MagicDraw:

> 1. Developed a method for exporting and transforming an entire model from the GENESYS™ application into a file formatted according to the Resource Description Framework (RDF).
> 2. Transformed a systems model of interest into an RDF file format structure and dropped the file into a location known by the target system/target import application, MagicDraw®.
> 3. Mapped the RDF file classes from GENESYS™ to custom MagicDraw® profile elements.
> a. The RDF namespaces were leveraged to guarantee valid (complete) containment.
> b. Model package structure was constructed in parallel to profile class structure.
> c. Custom profile element types and standard profile stereotype superclass/ metaclass types were designated according to the GENESYS™ object type.
> i. Encapsulated in independent configuration data classes.
> ii. Maintained flexibility while identifying all added data.
> 4. Redeveloped (refactor, expand, improve) an in-house developed MagicDraw® plugin used in previous projects.
> a. The RDF file was loaded to an in-memory ontology model, using an RDF library.
> b. Added traceability presenting views on the data for error checking.
> c. Added model post-processing (consistency checks/cleanup).
> d. Added prebuilt structure to process model instances.

Sandia's approach relied heavily on resource description framework (RDF) files and an internally developed Genesys RDF translator created as part of prior Sandia (Carroll 2019) research. The Genesys RDF translator's function was "to translate (i.e., transform) a GENESYS™ project into an RDF graph" (Carroll et al. 2021). Their approach also relied on an RDF-to-MagicDraw plugin written in Java, another internally developed tool.

The results of the Sandia demonstration and report detail how "the GENESYS™ model transformed into a MagicDraw® model did not produce an identically matching model." However, Sandia (Carroll et al. 2021) reports that "100% of the GENESYS™ model data was transformed and… the model objects and relationship were transformed (retaining the meaning of the model)." One of the most significant observations made by the report states that "SE models can be integrated effectively when the underlying ontological structure of the model is maintained through transformation" (Carroll et al. 2021).

As with NSWC Crane, Sandia also utilized internally developed software tools. Thus, the same implications are associated with Sandia's effort, as the implemented methodology is not usable or available to the wider SE community. This proof of concept also differs from the NSWC Crane effort in terms of validation. Sandia offers insight into validation efforts within their report and demonstrates successful translation between Genesys and MagicDraw. However, not all element and relationship types were present in the two models used by Sandia, so while Sandia transformed 100% of the original model, their method may potentially still have gaps.

The most important observation regarding both efforts and successful translation and model interoperability is how retention of the semantic structure, and an understanding of the ontological differences and their implications, are key to successfully integrating SE models.

3. SodiusWillert's Publisher for Rhapsody

In August 2022, the company SodiusWillert presented their software solution, Publisher for Rhapsody, to the D/SET community via virtual presentation. Publisher for Rhapsody is a plugin for IBM's Rhapsody modeling tool that facilitates model

transformation from Rhapsody to Ma gicDraw/Cameo using automated techniques. The plugin also logs individual element transformations and alerts the user to any conflicts during the transformation process and validates the completeness of the resultant model. The company reports successful use of their solution on models that contain upwards of 900,000 elements and 7,000 diagrams, demonstrating the program's scalability (Pilato 2022).

Although this solution transfers data between two tools already employing the same modeling language (SysML), its purpose of transferring model data from one tool to another without information loss is identical to the purpose of this thesis and demonstrates success in doing so. Based on this evidence, a similar software solution for the use-case examined by this thesis (i.e., going from Genesys to MSOSA), is theoretically feasible.

THIS PAGE INTENTIONALLY LEFT BLANK

III. ANALYSIS OF CONVERSION

This section presents the experimental setup and methodology used to conduct the research. The data found by using the methodology and the experimental setup then follows.

A. METHODOLOGY

1. Experimental Setup

Table 2 shows the software tools used, their version, and notes on their configuration.

Table 2. Applications Used in Experiment

Application	**Properties to Note**
Genesys 5.0 Collaborative Edition	Baseline Schema: Base Schema v50 Base Set View: SysML
Excel 2016	Add-ins: Genesys 5.0 Excel Connecter Enabled
Magic System of Systems Architect 2021x	Environment: SysML (Expert)

The conversion experiment uses the Geospatial Library (GSL) sample model provided with the Genesys software. The GSL model is a Vitech-created system model used for demonstration and educational purposes. Vitech ("Genesys 2021 R2 Systems Engineering Guided Tour" 2021) defines the GSL model as a "demonstration system [that] accepts requests for imagery information, determines the best way for the system to respond to the request, and then provides the request information to the requestor."

The experiment addresses three of the four SysML pillars: the structure (components), behavior (functions), and requirements pillars. The author's experience with modeling at NAWCADLKE has been that most models only use these three pillars. Consequently, the fourth SysML pillar of parametrics was not in scope for the assessment. Parametrics are useful for model simulation and analysis, but are a degree deeper into an SoI's structure that is not necessary for this initial assessment.

a. GSL Model Structure

The GSL model has nine physical *components*, ninety-eight *functions*, thirty-five *requirements*, and additional entities such as *states*, *test activities*, *test configurations*, and *use cases*. The total number of entities amounts to 423 in the GSL model. Relationships among all entities in the GSL model total to 2,358. Accounting for relationship directionality (e.g., considering relationships such as *traces to* and *traced from* as one bi-directional relationship), the GSL model consists of 1,304 relationships. Figure 7 shows the composition of the GSL model using the Project Explorer view in Genesys.

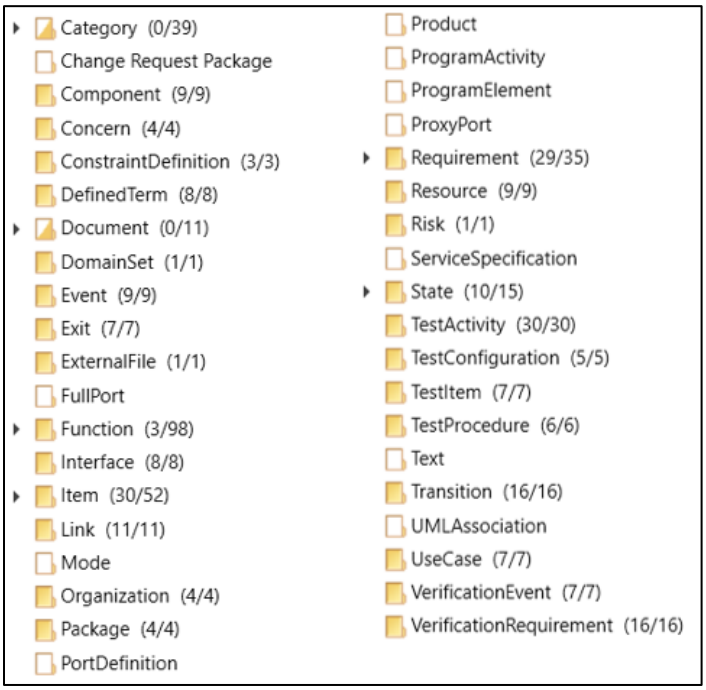


Figure 7. GSL Model Composition

Figure 8 shows the physical hierarchy of the GSL model with its larger system context. The GSL is composed of two subsystems: the Command Center and the Workstation. The GSL interacts with Customers, Collectors, and the Certification Authority, which are external to the GSL system.

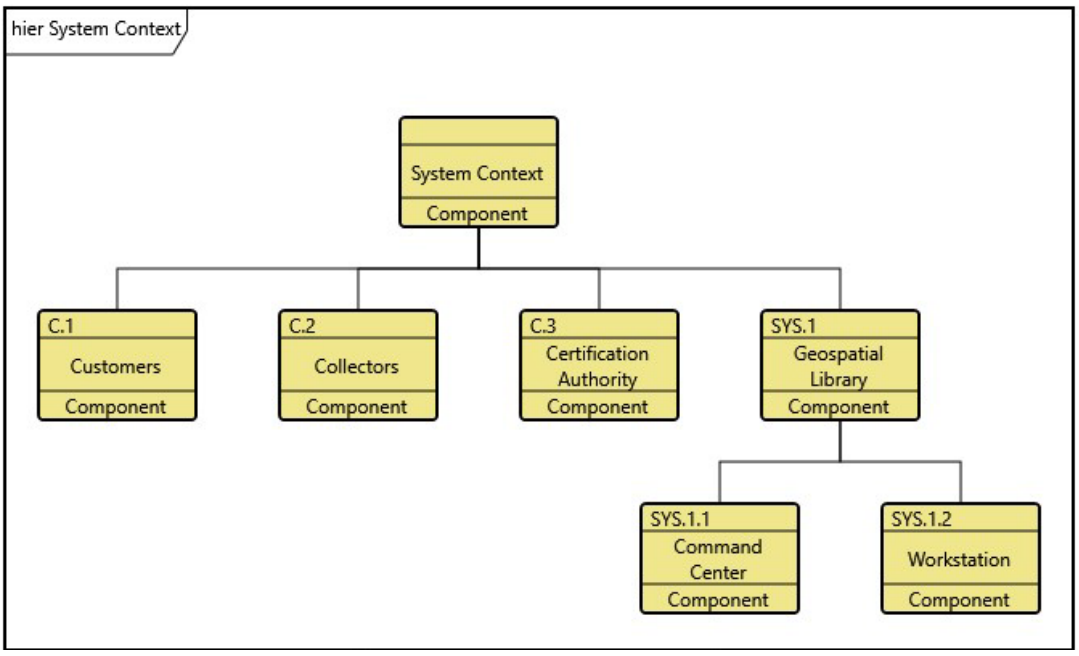


Figure 8. GSL Physical Hierarchy Diagram

Figure 9 shows the interfaces and links between each component via a physical block diagram, including the GSL's connections to the Customer and Collectors. In SDL, an *interface* represents some form of connection to another *component* and *includes* one or more *links*. *Links* represent more detailed connections to other *components* and show transfer of *items*, such as energy, mass, money, or more commonly, information. For example, the Workstation has three *interfaces* that include five *links*: the Workstation *interface* to the Command Center *includes* the single *GL Internal Link*, the Workstation *interface* to the Customer *includes* two *links* for disapproval notification and product request, and the final Workstation *interface* to the Certification Authority *includes* two *links* for certification request and certification response.

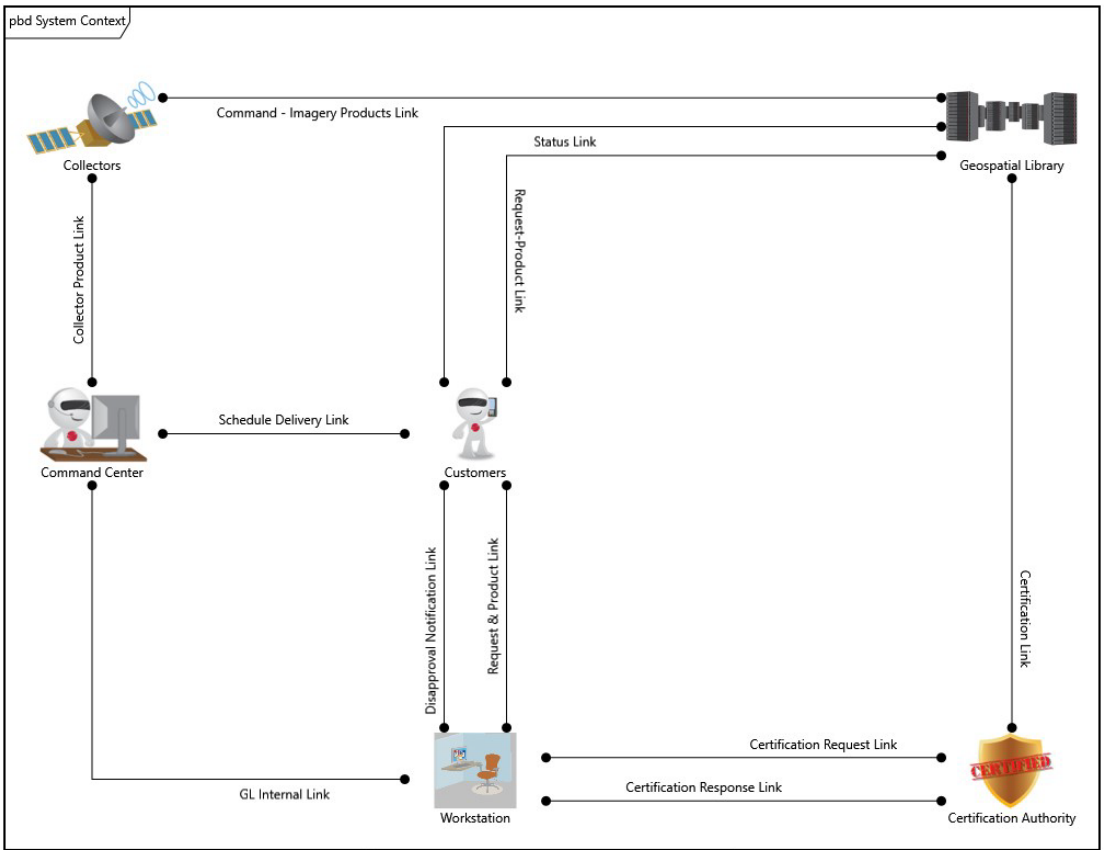


Figure 9. GSL Physical Block Diagram of System Context

b. GSL Model Behavior

Figure 10 depicts some of the system's behavior via a partial activity diagram, showing the various information *items* (represented by blue nodes) exchanged between the GSL and Certification Authority. For example, the GSL accepts and formats the request from the customer, provides the certification request to the Certification Authority, which then provides certification responses back to the GSL to check the certification response. The full activity diagram from the GSL sample model is available in APPENDIX A.

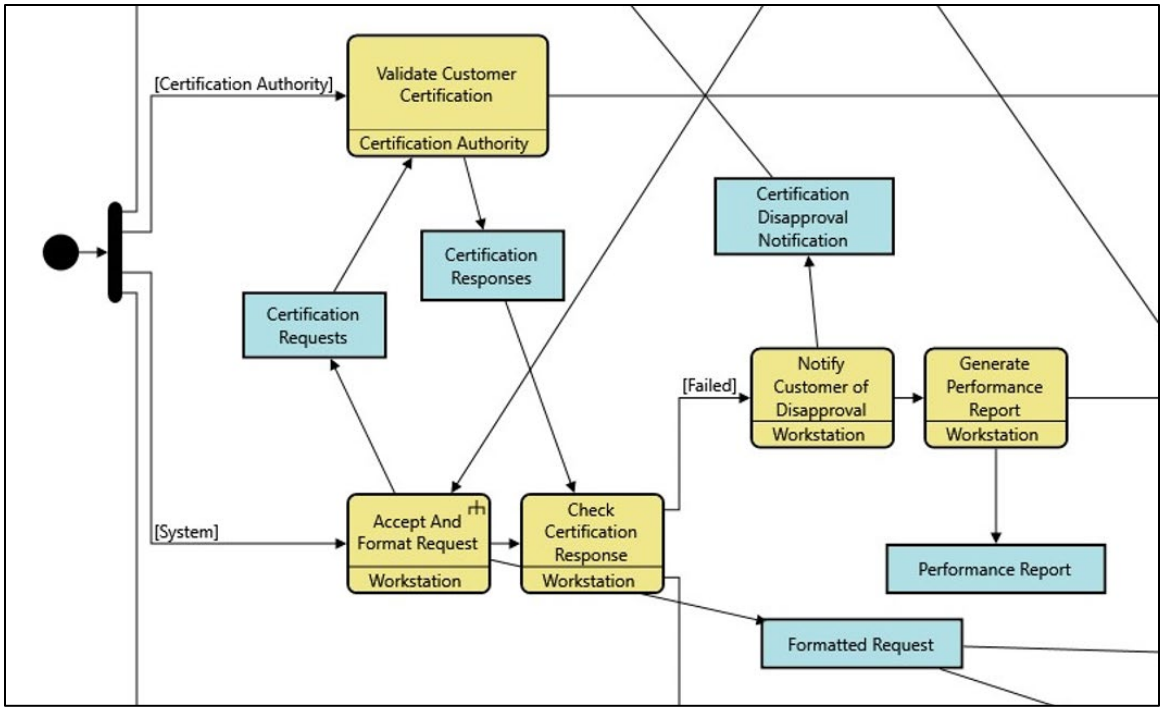


Figure 10. Partial GSL Activity Diagram

c. GSL Requirements Diagrams

Figure 11 depicts a requirements diagram of the GSL in Genesys and shows the "Continuous Support and Availability" *requirement* decomposition. A second requirements diagram depicts the "Specific Requirements" for the GSL in Genesys, and is available in APPENDIX B.

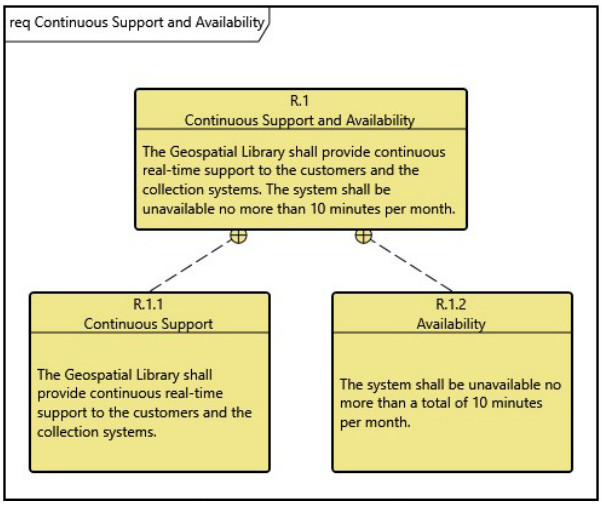


Figure 11. Partial GSL Requirements Diagram

2. Research Methodology

This research seeks to convert a system model from a source tool using one modeling language into a target tool using another modeling language. The research then assesses the converted model to determine and document informational inconsistencies and identify post-migration remediation efforts. When importing directly into a target tool proved infeasible, the data of the exported model from the source tool was examined as well. Figure 12 illustrates the analysis methodology as an Action Diagram created in SPEC Innovations' Innoslate software. With the Naval IME on the horizon for many Navy commands, this thesis assesses two tools with this methodology: Vitech's Genesys, predominantly used at NAWCADLKE and accessible to the author, and Dassault Systemes' MSOSA, a tool similar to the tool employed by the Naval IME, Cameo Systems Modeler.

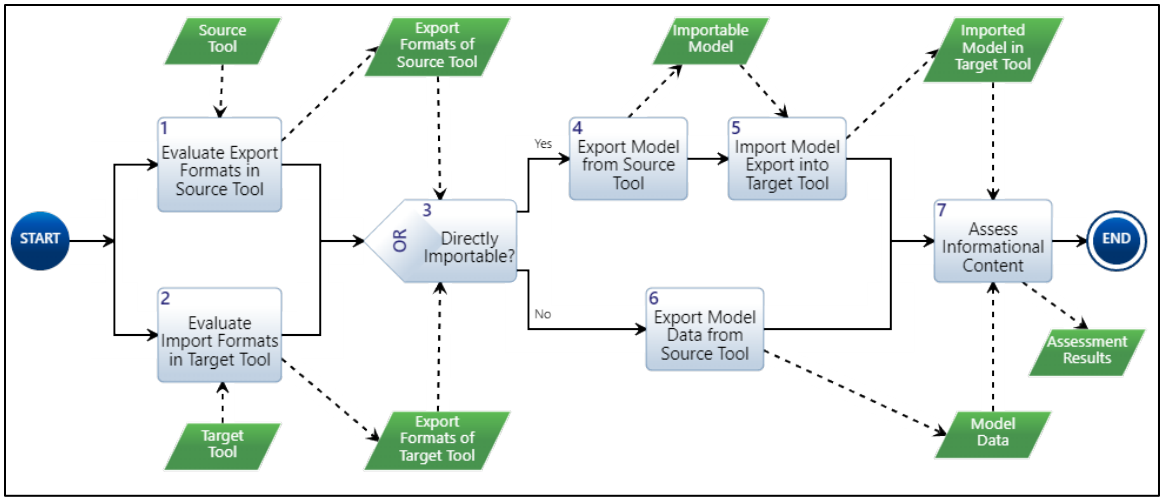


Figure 12. Research Methodology

a. Activity 1 – Evaluate Export Formats in Source Tool

Activity 1 in Figure 12 represents the initial evaluation of data export methods for the source tool. It is concerned with identifying the medium(s) available to the user for data transfer from the source tool. Its input is the selection of the source tool, and its output is a listing of file formats and plugins available to the user for data export in the source tool.

b. Activity 2 – Evaluate Import Formats in Target Tool

Like Activity 1, Activity 2 in Figure 12 represents the initial evaluation of data import methods for the target tool. It is concerned with identifying the medium(s) available to the user for data transfer to the target tool. Its input is the selection of the target tool, and its output is a listing of file formats and plugins available to the user for data import in the target tool.

c. Activity 3 – Directly Importable? (OR Gate)

Activity 3 in Figure 12 represents a decision based on the outputs of activities 1 and 2, the export and import formats for either tool. Being directly importable implies a software solution is available to the user for data transfer directly from the source tool to the target tool and is provided by one or both tools (i.e., no need exists for intermediate software provided by a third party, such as a custom plugin). With this information, a "yes

or no" assessment may be made for advancement through the workflow. If yes, the workflow advances to Activity 4. If no, the workflow advances to Activity 6.

d. Activity 4 – Export Model from Source Tool

Activity 4 in Figure 12 represents the export of a model from the source tool due to a "yes" decision input from Activity 3. In this activity, the user exports the model from the source tool using the identified file format detailed by Activity 1. An importable model is the output of this activity.

e. Activity 5 – Import Model Export into Target Tool

Activity 5 in Figure 12 represents the import of a model into the target tool using the output from Activity 4, the exported model. The imported model in the target tool is the output of this activity.

f. Activity 6 – Export Model Data from Source Tool

Activity 6 in Figure 12 represents the export of model data from the source tool due to a "no" decision input from Activity 3. In this activity, the user exports model data into a human-readable file or collection of files. This file or collection of files is the output of Activity 6.

g. Activity 7 – Assess Informational Content

Activity 7 in Figure 12 represents the final assessment of the resultant model or model data. Dependent on the path, the imported model in the target tool or the exported model data from the source tool are the input. The activity's output is the assessment results.

B. EXPERIMENT EXECUTION AND OBSERVATIONS

This section discusses the experiment and data this study collected using the methodology described in the previous section. Its organization aligns with Figure 12 and the three SysML pillars of interest to this research. Subsection 1 details the supported data formats for each of the two tools assessed, covering the initial evaluations of import and

export formats. Section 2 then supplies the data on tool interoperability, covering the "directly importable" assessment, the export the model and model data, and import of the model. It is followed by Chapter IV, which provides the results of the assessment.

1. Supported File Formats by Tool

MBSE tools do not support all file formats, nor will each tool necessarily support the same file formats. This section details the supported file formats for imports and exports in Genesys and MSOSA.

a. Vitech Genesys File Formats

Genesys can import Genesys and Core file formats (*.gsnx and *.xml, respectively). Many tools use the extensible markup language (XML) for data transfer, with Vitech's tools being among them. However, the structure of XML files can vary from tool to tool. Vitech's XML implementation structure appears to be unique, which limits importable XML files to those created by Vitech tools. Genesys exports models in a single format, a Genesys archive file (*.gsnx). Within this option, various selections are available to the user for the export of certain parts of the project, such as a project schema, a project template, or a full project backup.

Genesys also supports the import and export of data to and from IBM's Dynamic Object-Oriented Requirements System (DOORS) tool via the DOORS connector. However, the DOORS connector only imports and exports *requirement* elements. Genesys must be connected directly to the DOORS tool via this connector for its use ("Genesys 4.1 DOORS Connector Guide" 2016).

The tool also imports and exports data to and from Microsoft Excel via the Excel connector, a Vitech-created plugin for Microsoft Excel. Genesys requires Excel to be connected to Genesys to use the Excel connector, but in contrast to the DOORS connector, the Excel connector allows for a user to read the data before it is imported or after it is exported.

Table 3 summarizes the import and export file formats supported by Genesys.

Table 3. Vitech Genesys Export/Import Formats

Vitech Genesys			
Supported Import Formats	File Extension	Supported Export Formats	File Extension
Genesys Archive	.gsnx	Genesys Archive	.gsnx
Core Archive	.xml	*DOORS	N/A
*DOORS	N/A	**Excel File	.xlsx
**Excel File	.xlsx		

*Via connector only, and limited only to requirement entities

**Via connector only

b. Observations from Genesys Supported File Formats

The Vitech Genesys tool seems to only support file exchange formats for Vitech tools. The DOORS and Excel connectors are exceptions to this.

The DOORS connector demonstrates limited capability. A user cannot view the output from DOORS prior to import into Genesys, or vice versa. Data transfer to or from DOORS must occur first before it is visible to the user in the target tool. Furthermore, the DOORS connector allows for entities of only one type: *requirement* because DOORS only captures requirements.

The Excel connector is most useful when modifying existing entities in the Genesys model by exporting data into Excel, making necessary modifications, and importing the modified data back into the same Genesys model.

The observation regarding Vitech file exchange formats is especially important in terms of interoperability with other models and modeling tools. Vitech seems to have designed Genesys with only the Genesys user in mind, not the greater SE community that may already have information captured in another vendor's tool, or desire to interoperate with models captured in another vendor's tool. If an organization has existing models created outside of a Vitech tool, it drastically reduces the appeal of Genesys to those organizations as existing models cannot be imported into Genesys and would need to be re-created. This presents a barrier in the adoption of Vitech's Genesys. In the opposite

direction, it is also a significant limitation for organizations that have already adopted Genesys as it effectively confines those organizations to the tool and prohibits them interoperating with models generated outside of Genesys or Core.

Genesys has an application programming interface (API) that follows standard Microsoft.NET framework practices ("Genesys 6.0 Getting Started with the Genesys API" 2018). The API requires elevated privileges beyond the basic user license. Organizations using the API would have to fund a programming project to review the GENSYS file format and then write the code to export and/or import the model data. This is obviously more difficult and costly than if Vitech provided data interchange capabilities from the onset.

c. MSOSA File Formats

MSOSA can import data in multiple file formats. This includes table-based file formats such as Microsoft Excel (*.xlsx) and comma-separated values (*.csv), as well as MSOSA native XML file formats and various XML metadata interchange (XMI) file formats (*.xmi). It contrasts with Genesys in that it provides greater support for data interchange from source tools other than itself, such as IBM's Rhapsody and Rational Software Architect tools.

MSOSA exports data in multiple file formats. Though not the same as the supported import file formats, MSOSA allows for the export of data in file formats such as ReqIF (*.reqif, *.reqifz), Eclipse modeling framework (EMF) Ecore (*.ecore, *.ecore.xmi), and various other XMI file formats. It also supports the import of dynamic model file formats such as functional mock-up units (FMU) (*.fmu) and Simulink (*.slx).

The import and export file formats supported by MSOSA are summarized in Table 4 ("Magic Systems of Systems Architect 2021x User Manual" 2020).

Table 4. Dassault Systemes MSOSA Import/Export Formats

Dassault Systemes MSOSA			
Supported Import Formats	File Extension	Supported Export Formats	File Extension
Excel	.xlsx	UML 2.5 XMI	.xmi
Comma-separated values	.csv	Requirements Interchange Format (ReqIF)	.reqif, .reqifz
UML 2.1/2.5 XMI	.xmi	SCXML	.scxml
MSOSA Native XML	.xml	Eclipse UML2 XMI	.xmi, .uml2
MOF XMI	.xmi	Modelica	.mo, .moe
Requirements Interchange Format (ReqIF)	.reqif, .reqifz	Simulink	.slx
CA Erwin Data Modeler	.erwin	XPDL	.xpdl
OWL Ontology	.owx, .owl, .rdf	BPMN2	.bpmn
Eclipse UML2 XMI	.xmi, .uml2		
FMU	.fmu		
Simulink	.slx		
Modelica	.mo, .moe		
XPDL	.xpdl		
**Enterprise Architect UML 2.1 XMI 2.1	.xmi		
*System Architect DoDAF 2.0	.xml		
*Rhapsody SysML	.xml		
*Rational Software Architect	.xml		

*Via plugin

**Enterprise Architect does not export 100% standard UML 2.1 XMI, and this causes some data loss during the import

("Magic Systems of Systems Architect 2021x User Manual" 2020)

d. Observations from MSOSA Supported File Formats

Imports of table-based file formats require the implementation of what MSOSA terms a "mapping." MSOSA requires the user to resolve any ontological differences between the heterogeneous modeling languages and/or schemas. A user may save a

mapping as a template; however, mappings are user-defined and MSOSA does not provide any predefined mappings. So, consistency is stressed when creating mappings to import tabular data. Figure 13 shows an example of the mapping function in MSOSA, where a user defines various parameters of the import, such as the import type, target scope, and data location in Excel. The "Nested Properties to Map" field identifies the element properties in MSOSA to map from data in Excel. In this example, MSOSA data elements (left) map from columns in an excel spreadsheet (right).

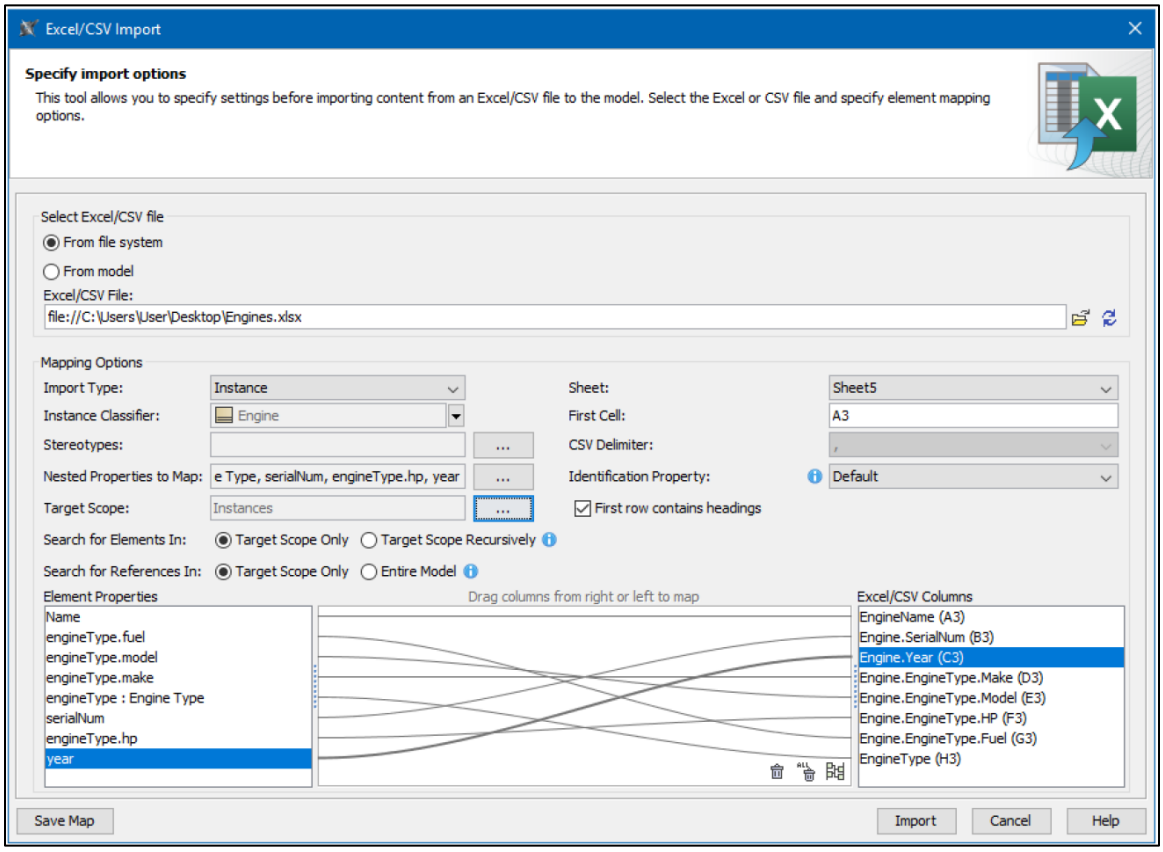


Figure 13. MSOSA Excel/CSV Import Mapping Example. Source: "Magic Systems of Systems Architect 2021x User Manual" (2020).

2. Migration Attempts

The experiments involved exports from Genesys (source tool) imported into MSOSA (target tool). This thesis then made a qualitative assessment of tool interoperability scenarios.

a. Genesys to MSOSA Migration Attempt

MSOSA cannot import .gsnx file formats, the export format of Genesys. Consequently, a direct import of a system model from the Genesys tool is infeasible. Following the research method presented in Figure 12, this thesis recorded the decision node for Activity 1.3 with Genesys "no," bringing this specific attempt to Activity 1.6. Despite failing this activity since Genesys XML files could not be directly imported into MSOSA, various Genesys XML files were reviewed to understand Vitech's implementation of XML constructs.

This research followed the steps detailed in NSWC Crane's *Genesys to Cameo Conversion Process* (2020), in which the .gsnx file can be renamed and decompressed as a .zip file, which enables viewing of the XML files that compose the .gsnx file. The .gsnx file consists of approximately 50 separate XML files that comprise the model's project data, schema data, and project metadata, such as change history and user accounts.

This research reviewed the XML files for three basic Genesys projects created for this thesis and one sample Genesys project: one with physical definition only, one with functional definition only, one with requirement definition only, and the sample GSL model provided by Vitech that contains all three constructs. Reviewing the multiple projects helped us understand Vitech's organization of the XML files.

The review revealed how Vitech implements XML to capture model data. Vitech uses two unique identification strings labeled "IdA" and "IdB" for each model element. Another data element, "EntityDefinitionID," defines the model entity type using a pre-defined library of entity names such as *component*, *function*, and *interface*. To manage relationships, Vitech uses "IdA" or "IdB," as "SourceEntityID" and "TargetEntityID" to establish which entities are to be related and the directionality of that relationship. Vitech then creates relationships between the entities via a similar means as the entity definition,

by using a “RelationDefinitionID” from a predefined library of relationship definitions, such as *decomposes*, *refines*, and *satisfies*. Some examples of the XML files generated by Genesys that contain GSL model data, such as the Entities.xml and Relationships.xml files, are depicted in Figure 14 and Figure 15, respectively. Some common tags in these XML files include: IdA, IdB, and data.

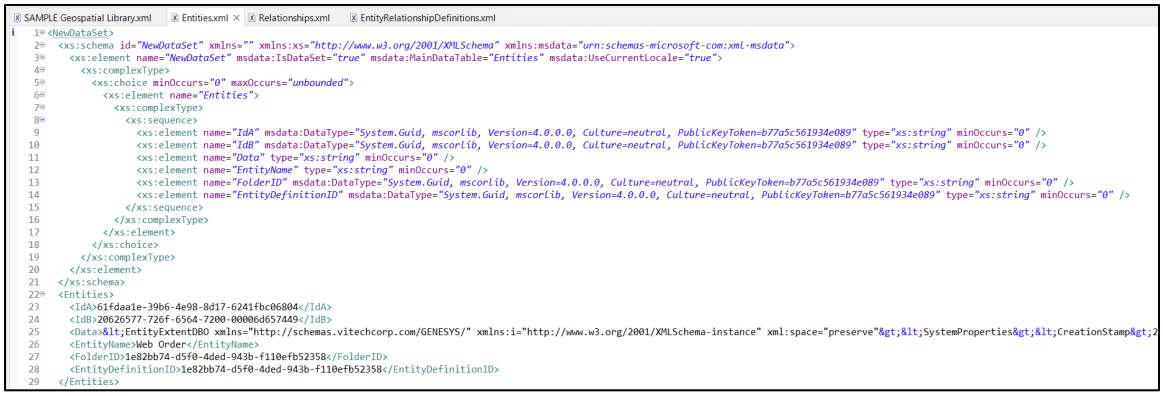


Figure 14. Genesys Entities.xml File Example from GSL Model

Figure 15. Genesys Relationships.xml File Example from GSL Model

Vitech captures the entity and relationship information with two of the many XML files within the .gsnx archive file, the “Entities.xml” file and the “Relationships.xml” file, respectively. The review of these XML files revealed that all project data is indeed captured and exported by the .gsnx archive file.

(1) Importing Model Structure into MSOSA

MSOSA accepts data in a tabular format via Excel or CSV files (i.e., *.xlsx or *.csv, respectively) via the use of a mapping tool shown in Figure 13. Custom plugins created by Sandia and NSWC Crane were not available to the author. Therefore, the mapping tool, in conjunction with the Excel Connector, is the only available way to import Genesys data into MSOSA without use of the API. As such, this study utilized the MSOSA mapping tool and Excel method in attempt to convert the model over to MSOSA.

When the Genesys's Excel connector outputs data to Excel, the output contains column headers that contain data labels (e.g., element name, element number, performs, etc.), and the body of the spreadsheet contains the content (e.g., Satellite, C.2.1, Perform Satellite Functions). The Genesys Excel connector requires a user to define the output columns (representing entities, relationships, and attributes to export) during the process, as shown in Figure 16.

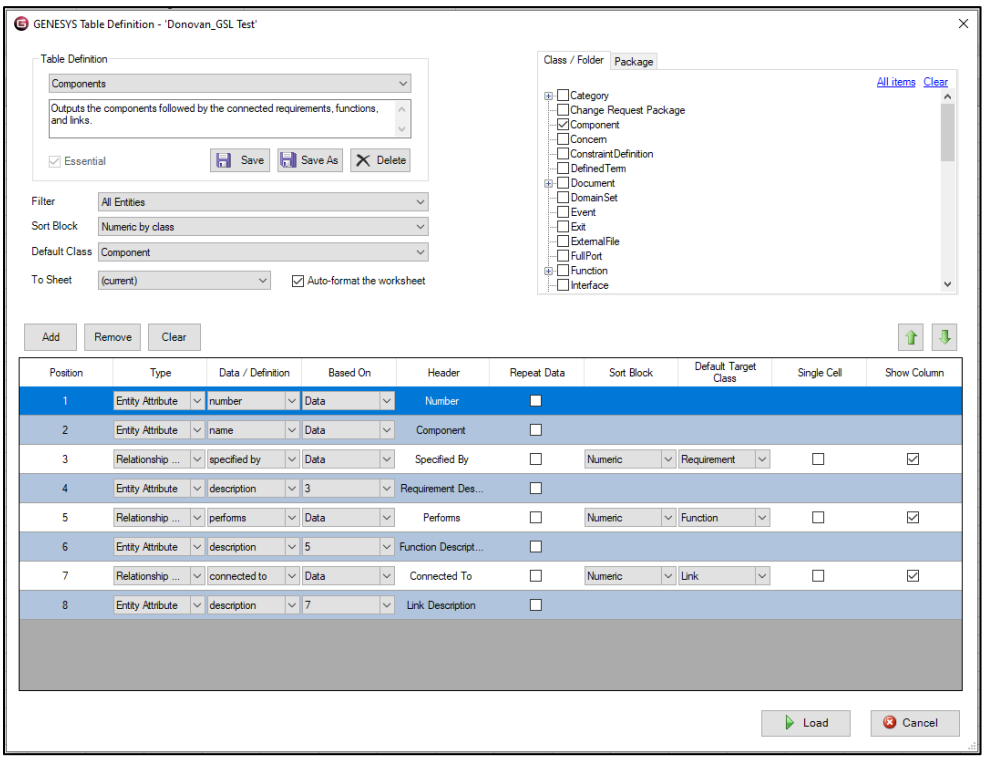


Figure 16. Table Definition of Genesys-Excel Connector in Microsoft Excel for Component Export

Figure 16 shows a default table definition called “Components,” which defines what model information the tool will export. “Position” identifies the numbered column the data will export to in the Excel spreadsheet. The “Type” provides a drop-down menu of data types (e.g., relationship, entity attribute). The “Data / Definition” drop-down menu selections are dynamically generated lists of attributes or relationships, depending on the “Type” selection (e.g., “name” for entity attribute, or “specified by” for relationship). The “Based On” field represents what entity the tool is using to query data and is useful when acquiring second-order data. In the “Based On” field, “Data” denotes the tool will pull information directly from the entity, whereas a numerical value denotes the tool will pull information based on the information in another numbered row, identified by the numerical value. For example, row 4 is “Based On” row 3 in Figure 16, which means row 4 will gather the entity attribute “description” from the entities found by row 3. In other words, row 3 will gather data on targets that the entity is *specified by*, and row 4 will gather the “description” of each entity found by row 3. The remaining fields are for formatting of the exported spreadsheet.

When using the mapping tool in MSOSA, one serious limitation becomes evident that severely inhibits tabular data as a scalable medium for information exchange between Genesys and MSOSA. This limitation occurs with how the mapping tool imports tabular data with a one-to-one mapping and obstructs complex data transfer.

Figure 17 shows how Genesys will output the data for a single element, in this case the Geospatial Library component, using multiple rows in the spreadsheet. The Geospatial Library *component performs* multiple *functions*, which are listed in column E. Referring again to Figure 16, rows 5 and 6 represent columns E and F in Figure 17, respectively. Column F, the *function* description, is based on the data in column E, the *functions* that the component Geospatial Library *perform*, as reflected in Figure 16.

A	B	C	D	E	F
Number	Component	Specified By	Requirement Description	Performs	Function Description
SYS.1	Geospatial Library	R.1.1 Continuous Support	The Geospatial Library shall provide continuous real-time support to the customers and the collection systems.	0 _Perform Geospatial Library Functions	The "root" function of the system. This function captures the behavior of the system.
		R.1.2 Availability	The system shall be unavailable no more than a total of 10 minutes per month.	t.2.4 t2.Accept & Format Request	The system shall accept the requests for information, verify that the requester is a valid customer of the system, and format the request into a form and media that the system can use.
		R.2.4 Maximum Staff	The system shall be staffed at a maximum of 30 personnel on any shift.	t.2.5 t2.Prioritize Request	The system shall organize customer orders using a priority-based approach. Priorities shall be assigned based upon delivery date, customer need, current collection assignments, and system resources.
				t.2.6 t2.Determine Collector Mix	The system shall evaluate the customer's order to establish which set of collectors shall best provide imagery products answering the customer's need.

Figure 17. Partial Genesys Excel/CSV Export of GSL Model

Figure 18 shows the mapping tool to get the Excel data into MSOSA. It shows that two columns from Excel are mapped to the properties of a block element. The Number column in Excel is mapped to the Element ID, and the component column in Excel is mapped to Name.

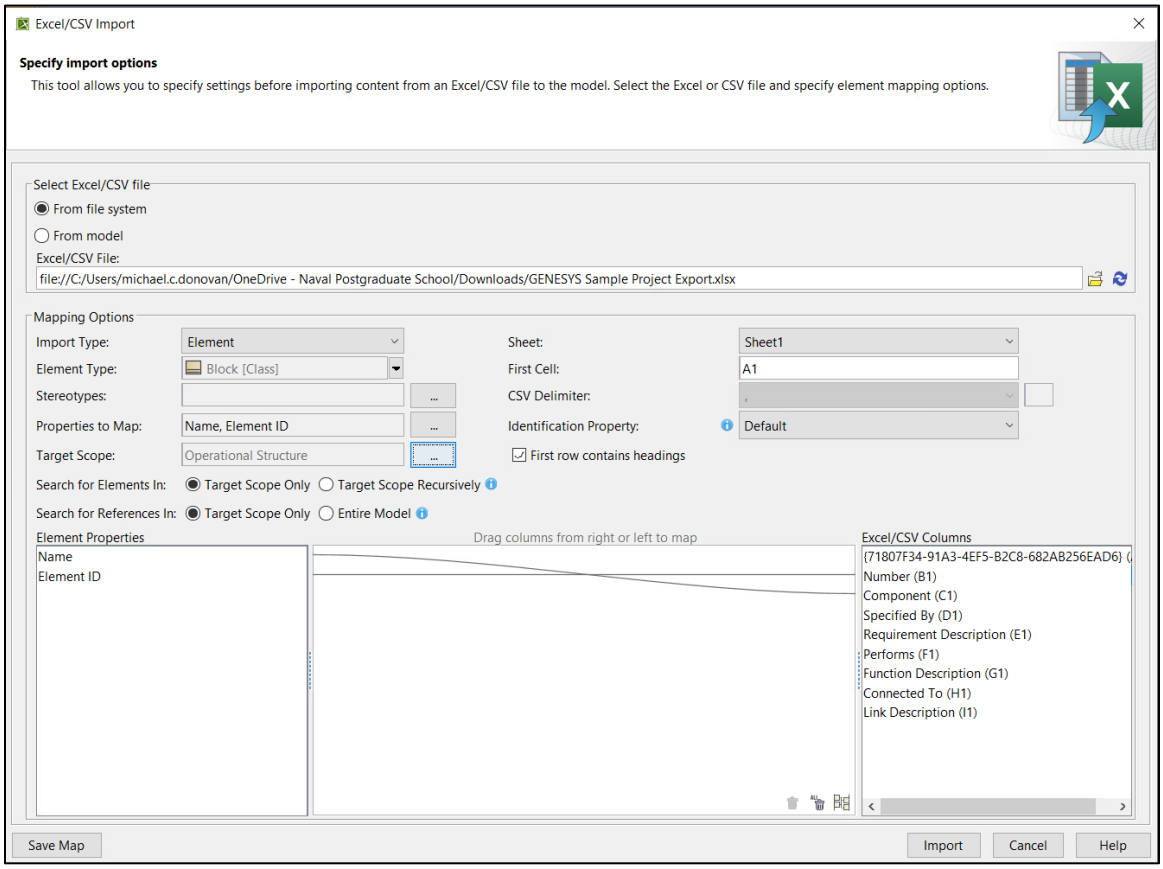


Figure 18. MSOSA Excel/CSV Import Mapping from Genesys GSL Components Export

A problem occurs because in the spreadsheet (Figure 17) there are blanks cells in the Component column. MSOSA does not know these cells are supposed to a continuation of the previous component (i.e., Geospatial Library). Instead, MSOSA creates separate elements based on these rows with empty component name. See Figure 19 where all the blank cells in the Component column are created as blocks in MSOSA.

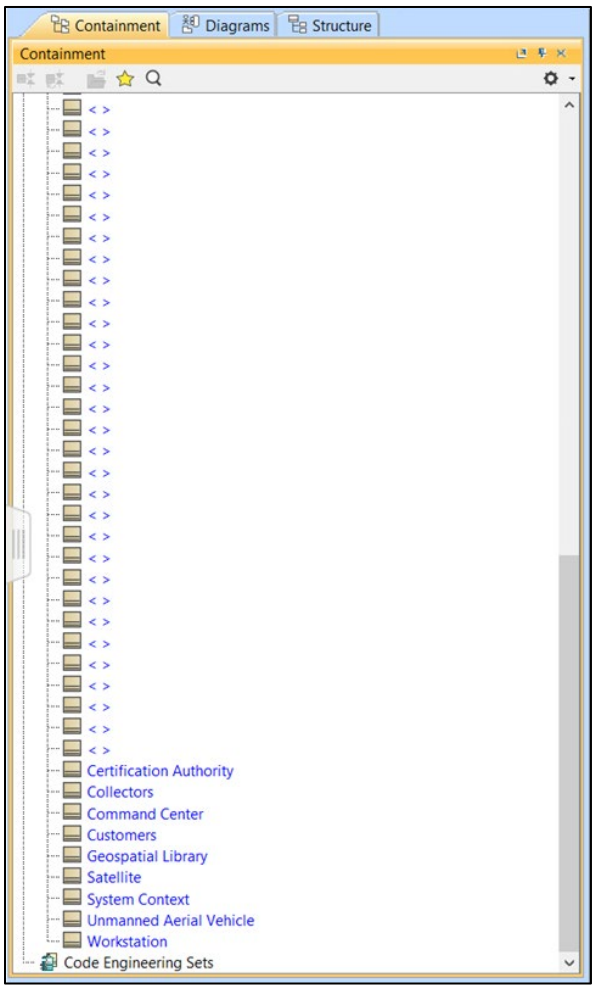


Figure 19. Genesys GSL Components Imported into MSOSA (Before Remediation)

This conversion error occurs whenever Excel data contains a Genesys element that has a one-to-many relationship with other elements and is imported into MSOSA. Genesys will output tabular data pertinent to a single entity on multiple rows if a one-to-many

relationship exists (e.g., the GSL *performs* multiple *functions*). Subsequently, MSOSA interprets this as separate entities in a one-to-one fashion, rather than multiple relationships to a single entity, and creates empty MSOSA entities during import. A user must therefore refine the Genesys export such that no blank cells are present before importing into MSOSA or delete MSOSA data after the import process. In this case, this study deleted the blank entities shown in Figure 19 after the import to yield Figure 20.

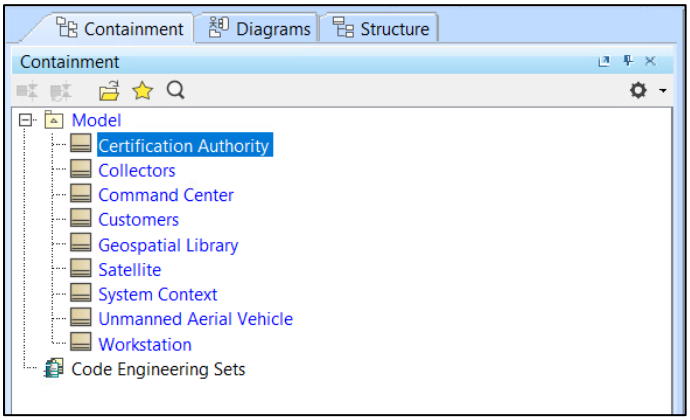


Figure 20. Genesys GSL Components Imported into MSOSA (After Remediation)

Figure 20 shows that using Excel for the conversion did not exchange any hierarchical information, which nested *blocks* in the containment view would illustrate. Instead, this method established all *blocks* at the same level. One would expect the System Context *block* to be composed of all other physical entities as it was in the original Genesys model. The MSOSA mapping tool is incapable of importing hierarchical relationships because MSOSA can import only one type of element at a time using the "Element Type" dropdown menu. Therefore, the user must first create both entities (source and target) before defining any relationship between those entities. This includes hierarchical relationships for functional and physical decompositions. In other words, you must create the entities that will be related before creating the relationship(s) among those entities.

This study makes two observations regarding this limitation: (1) the import mapping tool must be run multiple times on multiple data sets to capture all of a model's

information (even simple ones), and (2) the actual order of the import process is of importance; otherwise relationship information may be incorrectly omitted.

To recreate the physical architecture of the MSOSA model such that it matches the Genesys model, this research made multiple attempts to import the physical relationships into MSOSA from Genesys using the Genesys-Excel connector and MSOSA Excel import function. This proved difficult because SysML represents the physical hierarchy via an entity property, whereas SDL uses a relationship to create hierarchy. For example, **** provide an example of how SysML does hierarchy.

In the MSOSA Excel import function, there is no means to establish part properties for *blocks* already imported, MSOSA will simply create new *blocks* with these part properties that is not desired. To work around this, this study deviates from the ontological mapping provided by NWSC Crane's *Genesys to Cameo Conversion Process* (2020), and the *owner* SysML property is mapped to the Genesys SDL relationship, *built in*. User manipulation in the MSOSA model, with this deviation, generates a physical hierarchy reflective of the source model in the model view, shown in Figure 21.

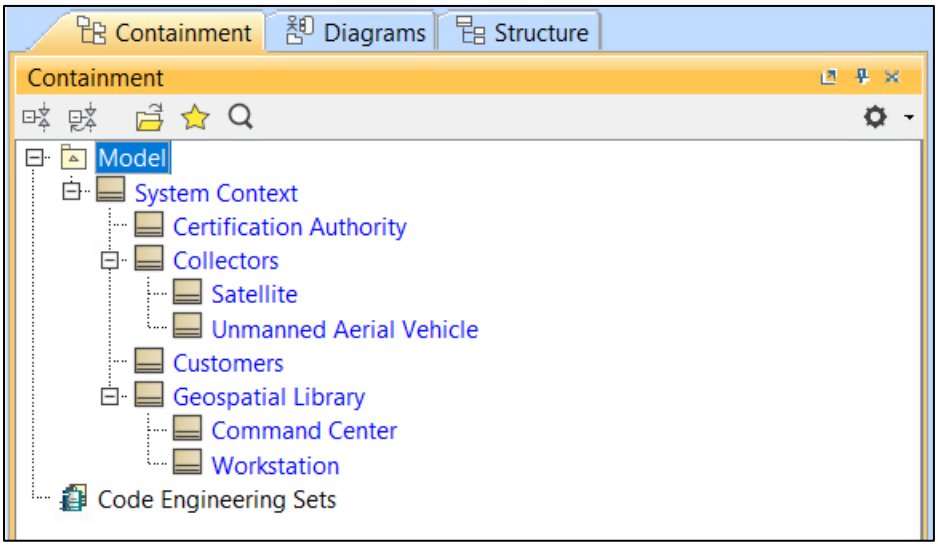


Figure 21. Physical Hierarchy in MSOSA Model Containment View

The use of the *owner* SysML property does not create the required relationships for the block definition diagrams (BDDs).

MSOSA is incapable of importing the physical architecture of a model from Genesys using the Genesys-Excel connector and MSOSA Excel import function alone. Instead, the user must go into MSOSA after importing the component elements and recreate the physical hierarchy by defining the owner property in the appropriate blocks. Then, using the "directed composition" selection within the MSOSA BDD successfully establishes the part properties and thereby creates the correct physical decomposition as shown in Figure 22, which matches Figure 9 from Genesys.

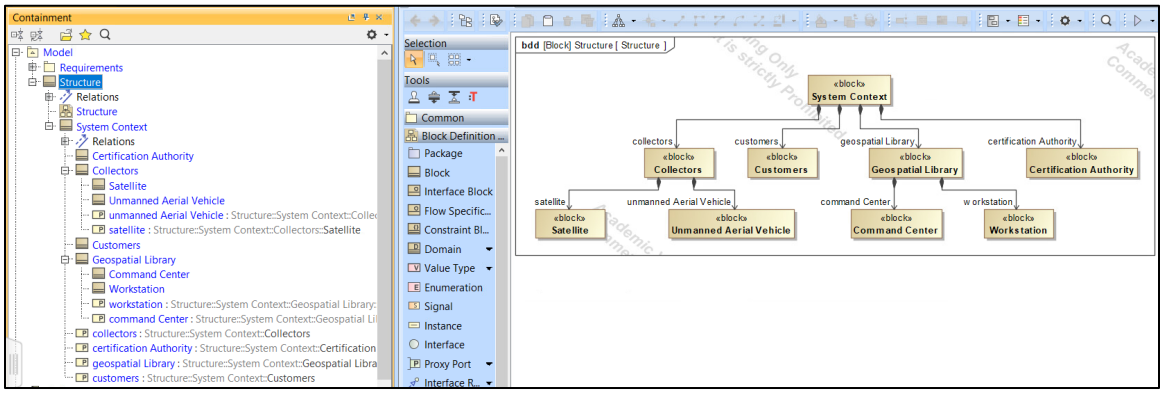


Figure 22. Block Definition Diagram (BDD) of GSL in MSOSA

(2) Importing Model Behavior into MSOSA

This study also attempted to recreate the functional architecture of the GSL in MSOSA. Figure 23 depicts a custom table definition used to export GSL behavior data into MSOSA, including the *function* names, descriptions, allocation relationships, and decomposition relationships from Genesys.

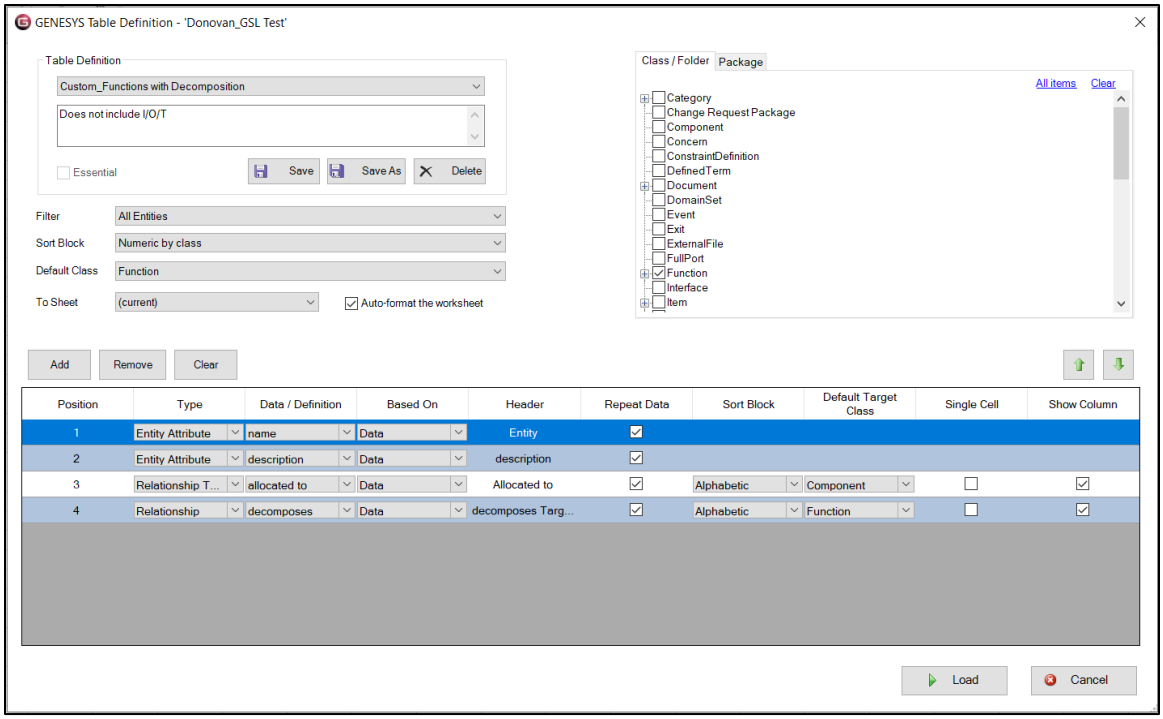


Figure 23. Custom Table Definition of Genesys-Excel Connector in Microsoft Excel for Function Export

Using the data generated in Figure 23, Figure 24 then depicts the mapping of that data into MSOSA. The element properties are the import fields to MSOSA, and the Excel/ CSV columns are the data from Genesys. The mapping abides by the conceptual mapping provided by NSWC Crane's *Genesys to Cameo Conversion Process* (2020), creating *CallBehaviorAction* entities in MSOSA as equivalents for *functions* in Genesys.

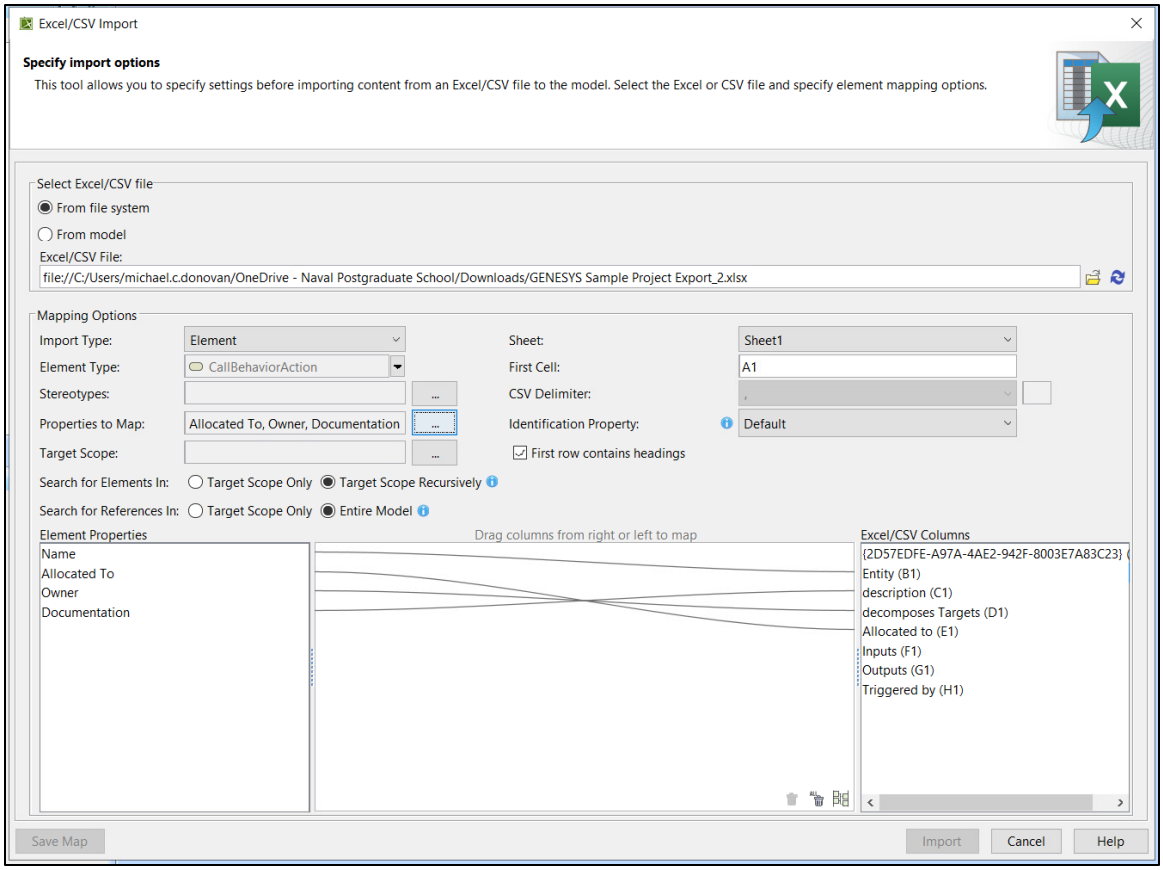
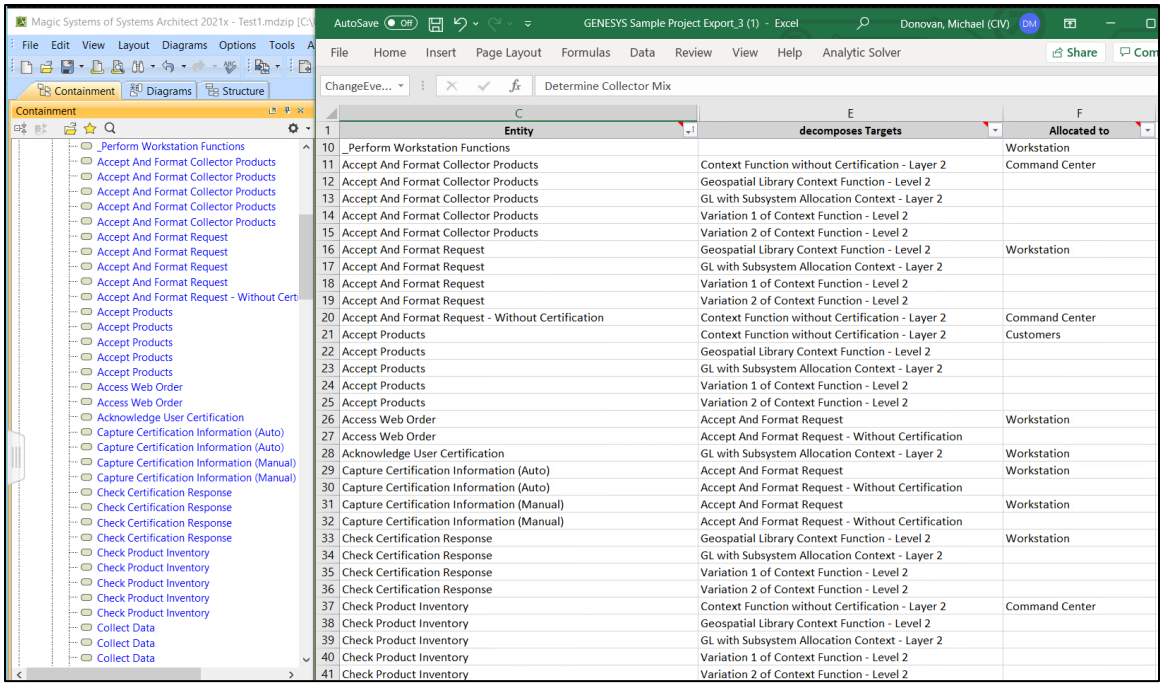


Figure 24. MSOSA Excel/CSV Import Mapping from Genesys GSL Functions Export

This successfully creates the Genesys functions as *CallBehaviorActions* in MSOSA, and correctly maps allocation relationships to their respective *blocks* (representing structure). This study observes two limitations during this import.

The first limitation concerns data duplication and is like the limitation where MSOSA imports empty entities due to blank cells in the imported spreadsheet. In Genesys, if an entity has multiple relationships with other entities, then when exported into Excel, each of those relationships is a different row in the spreadsheet. When imported into MSOSA, MSOSA incorrectly interprets each row as a different entity and creates a separate block. Figure 25 shows the *function* "Accept Products" occurs five times because it was in five separate rows in the spreadsheet.



	C	E	F
1	Entity	decomposes Targets	Allocated to
10	_Perform Workstation Functions		Workstation
11	Accept And Format Collector Products	Context Function without Certification - Layer 2	Command Center
12	Accept And Format Collector Products	Geospatial Library Context Function - Level 2	
13	Accept And Format Collector Products	GL with Subsystem Allocation Context - Layer 2	
14	Accept And Format Collector Products	Variation 1 of Context Function - Level 2	
15	Accept And Format Collector Products	Variation 2 of Context Function - Level 2	
16	Accept And Format Request	Geospatial Library Context Function - Level 2	Workstation
17	Accept And Format Request	GL with Subsystem Allocation Context - Layer 2	
18	Accept And Format Request	Variation 1 of Context Function - Level 2	
19	Accept And Format Request	Variation 2 of Context Function - Level 2	
20	Accept And Format Request - Without Certification	Context Function without Certification - Layer 2	Command Center
21	Accept Products	Context Function without Certification - Layer 2	Customers
22	Accept Products	Geospatial Library Context Function - Level 2	
23	Accept Products	GL with Subsystem Allocation Context - Layer 2	
24	Accept Products	Variation 1 of Context Function - Level 2	
25	Accept Products	Variation 2 of Context Function - Level 2	
26	Access Web Order	Accept And Format Request	Workstation
27	Access Web Order	Accept And Format Request - Without Certification	
28	Acknowledge User Certification	GL with Subsystem Allocation Context - Layer 2	Workstation
29	Capture Certification Information (Auto)	Accept And Format Request	Workstation
30	Capture Certification Information (Auto)	Accept And Format Request - Without Certification	
31	Capture Certification Information (Manual)	Accept And Format Request	Workstation
32	Capture Certification Information (Manual)	Accept And Format Request - Without Certification	
33	Check Certification Response	Geospatial Library Context Function - Level 2	Workstation
34	Check Certification Response	GL with Subsystem Allocation Context - Layer 2	
35	Check Certification Response	Variation 1 of Context Function - Level 2	
36	Check Certification Response	Variation 2 of Context Function - Level 2	
37	Check Product Inventory	Context Function without Certification - Layer 2	Command Center
38	Check Product Inventory	Geospatial Library Context Function - Level 2	
39	Check Product Inventory	GL with Subsystem Allocation Context - Layer 2	
40	Check Product Inventory	Variation 1 of Context Function - Level 2	
41	Check Product Inventory	Variation 2 of Context Function - Level 2	

Figure 25. Example of Entity Duplication in MSOSA

The second limitation this study observed is when transferring the behavior modeling data. This study was able to export and import *function* definitions, but it could not import the behavior modeling data associated with the *functions*, such as the inputs and outputs for each *function*. This is due to semantic and ontological differences between SDL and SysML combined with the one-to-one limitation of the Excel import method.

SDL uses the *function* entity type for all behaviors within the system architecture domain. *Functions output items* that are themselves entities and are *transferred* via *links* between two *components* in SDL. Furthermore, the SDL entities of type *item* can be an *input* to, an *output* of, or a *trigger* of a *function*.

In contrast, SysML defines two element types called *actions* and *activities* in which an activity is the higher level construct and can contain multiple actions, a lower-level construct (Delligatti 2014, 93). Furthermore, SysML employs the concept of *tokens,* which "are not model elements" and consist of two types: an *object token,* that represents an instance of matter, energy, or data that flows through an activity, and a *control token* that "simply indicates which action in an activity is currently enabled at a particular moment

during an execution of the activity" (Delligatti 2014, 71). *Object nodes* instantiate *object tokens* in SysML and are transferred via *edges*. There exist many other specialized types of nodes as well, which the *edges* can connect to, and *edges* themselves may also be specialized. SysML also employs *activity input* and *output pins*, which are *object nodes* as well but require less real estate on an activity diagram.

When mapping to SysML per NSWC Crane's *Genesys to Cameo Conversion Process* (2020), *activity input* and *output pins* as well as *central buffer nodes* were used. However, *activity input* and *output pins* are properties of *activities* and not individual entities themselves, with individual *pins* used for each *block* with a *central buffer node* between them. This does not align with the SDL ontology to capture behavior. These ontological differences, combined with the Excel import method's one-to-one mapping limitation, demonstrated that there are no means to import behavior data into MSOSA from Genesys. Instead, model behavior must be recreated by the user within MSOSA. This limits the usefulness of Excel imports to only the import of *functions* from Genesys as it does not capture the actual dynamic behavior it represents.

These two limitations imply that the import of behavior modeling data from Genesys to MSOSA via the Genesys-Excel connector and MSOSA mapping tool is feasible but insufficient.

(3) Importing Model Requirements into MSOSA

The MSOSA 2021x User Manual (2020) instructs users to manually include the additional columns "id" and "owner" when importing *requirements* via spreadsheets. The additional columns enable MSOSA to interpret the decomposition of those *requirements*. Consequently, after exporting requirements from Genesys, the Excel file is manipulated by adding those columns and the required data. Figure 26 provides an excerpt of the requirements spreadsheet after manipulation. The first *requirement*, "Continuous Support and Availability," is the parent of the second and third *requirements*, "Continuous Support" and "Availability" as shown because the owner of these requirements is "1." The fourth *requirement*, "Specific Requirements," is the parent of the "Accept Requests from Certified Customers" *requirement*, and so on.

	A	B	C	D	E
1	id	Requirement	Description	Owner	Parent Requirement
2	1	Continuous Support and Availability	The Geospatial Library shall provide continuous real-time support to the customers and the collection systems. The system shall be unavailable no more than 10 minutes per month.		
3	2	Continuous Support	The Geospatial Library shall provide continuous real-time support to the customers and the collection systems.	1	R.1 Continuous Support and Availability
4	3	Availability	The system shall be unavailable no more than a total of 10 minutes per month.	1	R.1 Continuous Support and Availability
	4	Specific Requirements	1. The system shall accept information requests from certified customers. 2. The system shall retain an inventory of previously collected images/products and provide them to users, if appropriate. 3. The system shall control multiple image collectors and multiple types of image collectors. 4. The system shall be staffed at a maximum of 30 personnel on any shift. 5. The system shall provide feedback on the customer's request within 24 hours. 6. The system shall provide a means of prioritizing the customer's requests. 7. The system shall monitor and assess its own performance.		
7	5	Accept Requests from Certified Customers	The system shall accept information requests from certified customers.	4	R.2 Specific Requirements
	6	Accept Requests	The system shall accept information requests.	5	R.2.1 Accept Requests from Certified Customers
	7	Accept Media of Requests	The system shall accept requests via any of the following media: 1) Hardcopy Forms;	6	R.2.1.1 Accept Requests

Figure 26. Modified GSL Requirements Table for MSOSA Import

Using this method, this study successfully imported the *requirements* from Genesys into MSOSA. Figure 27 is a requirements diagram from MSOSA that demonstrates how the nested *requirements* imported correctly. Note that Figure 27 reflects the content contained in the original Genesys requirements diagram, Figure 11.

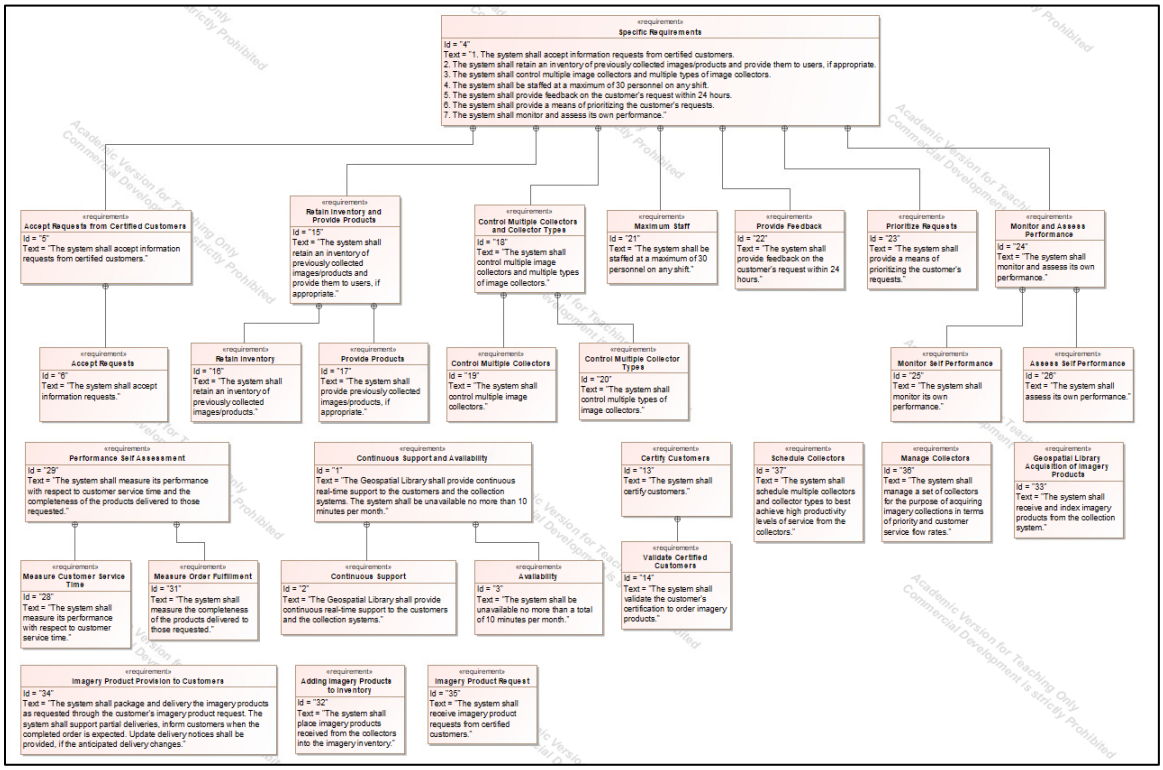


Figure 27. GSL Requirements Diagram In MSOSA

This study also attempted to transfer any Genesys relationships between *requirements* and elements in the physical and functional architectures. Referring again to Figure 3, *requirements* are the *basis of* a *function*, and a *function* may be *specified by requirements* in Genesys. *Components* may also be *specified by requirements*. Per NSWC Crane's *Genesys to Cameo Conversion Process* (2020), the SDL *basis of* and *specified by* relationships between *requirements* and *functions* are both mapped to the *refine* relationship between a *CallBehaviorAction* and *requirement* in SysML. Furthermore, the SDL relationships *specifies/specified by* between *requirements* and *components* map to the *satisfy* relationship between a *blocks* and *requirements*.

Importing these relationships using a single Excel sheet, like Figure 28, yielded an issue where MSOSA incorrectly placed the *refines/refined by* relationship onto *blocks*, and incorrectly placed the *satisfies/satisfied by* relationship onto *CallBehaviorActions*.

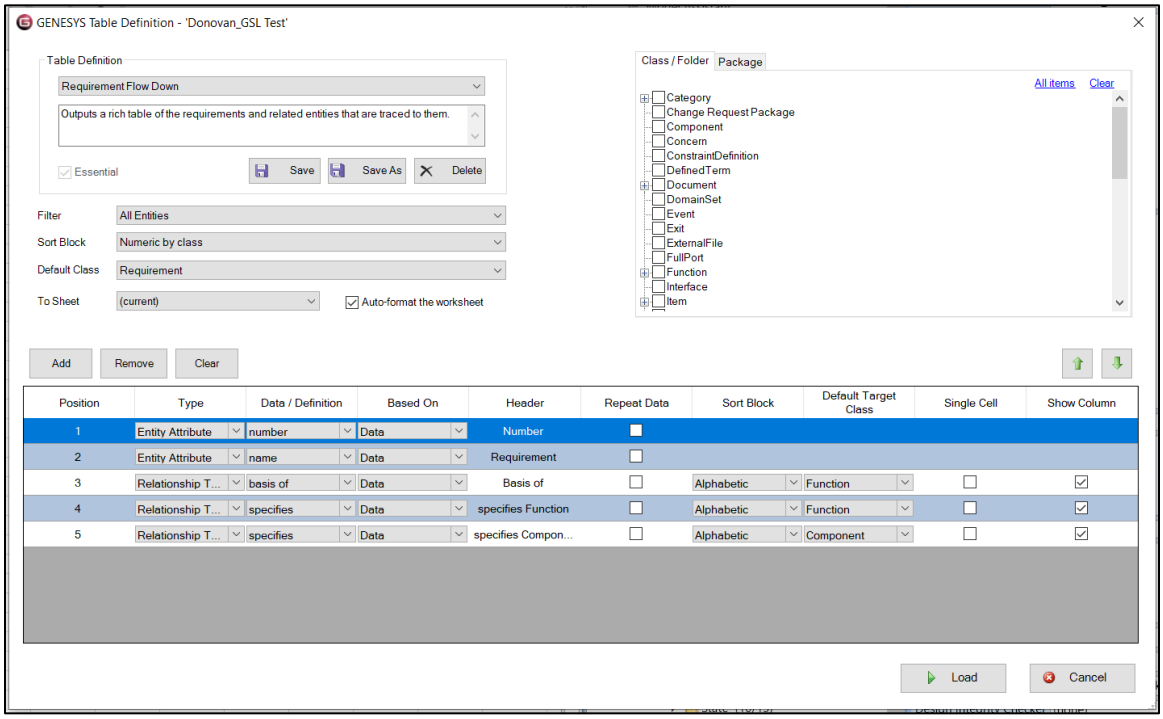


Figure 28. Genesys Requirement Relationships Export Table Definition (Too Rich)

This was due to the export from the Genesys Excel connector table definition and the MSOSA import function. The table definition captured all entities for each *requirement* where that *requirement*: was the *basis of* a *function*, *specifies* a *function*, and *specifies* a *component*. There were simply too many different types of relationships in the table for MSOSA to interpret correctly using a single set of import options. Instead, the user must constrain the output scope of the table definition. The GSL export was then obtained twice using these less-rich table definitions – once for relationships between *requirements* and *functions* (Figure 29), and again for relationships between *requirements* and *components* (Figure 30).

Position	Type	Data / Definition	Based On	Header	Repeat Data	Sort Block	Default Target Class	Single Cell	Show Column
1	Entity Attribute	number	Data	Number	☐				
2	Entity Attribute	name	Data	Requirement	☐				
3	Relationship T...	basis of	Data	Basis of	☐	Alphabetic	Function	☐	☑
4	Relationship T...	specifies	Data	specifies Function	☐	Alphabetic	Function	☐	☑

Figure 29. Genesys Requirement Relationships Export Table Definition (Function Relationships)

Position	Type	Data / Definition	Based On	Header	Repeat Data	Sort Block	Default Target Class	Single Cell	Show Column
1	Entity Attribute	number	Data	Number	☐				
2	Entity Attribute	name	Data	Requirement	☐				
3	Relationship T...	specifies	Data	Specifies	☐	Alphabetic	Component	☐	☑

Figure 30. Genesys Requirement Relationships Export Table Definition (Component Relationships)

This study then executed the MSOSA import three times using the two exported tables. The first two imports were for *requirements* that *specify functions*, and then for *requirements* that are the *basis of functions*. MSOSA requires this because. *Requirements* in Genesys may both *specify* a *function* and be the *basis of* a *function*, yet, the *refines/ refined by* is the only relationship both map to in MSOSA, per NSWC Crane's *Genesys to Cameo Conversion Process* (2020). This translates to a mapping that is two-to-one. The MSOSA import tool does not allow for mappings that are not one-to-one, therefore it required two separate imports. The third import established the *satisfies/satisfied* by

relationship between *requirements* and *blocks*. These three steps correctly map *requirements* to *refine CallBehaviorActions*, and to be *satisfied by blocks*.

(4) Summary

To summarize the steps taken for importing data from Genesys into MSOSA via the Excel connector and MSOSA mapping tools, Figure 31 illustrates how this study was successful in transferring parts of the GSL model in three steps, aligned to each of the three SysML pillars: physical, functional, and requirements.

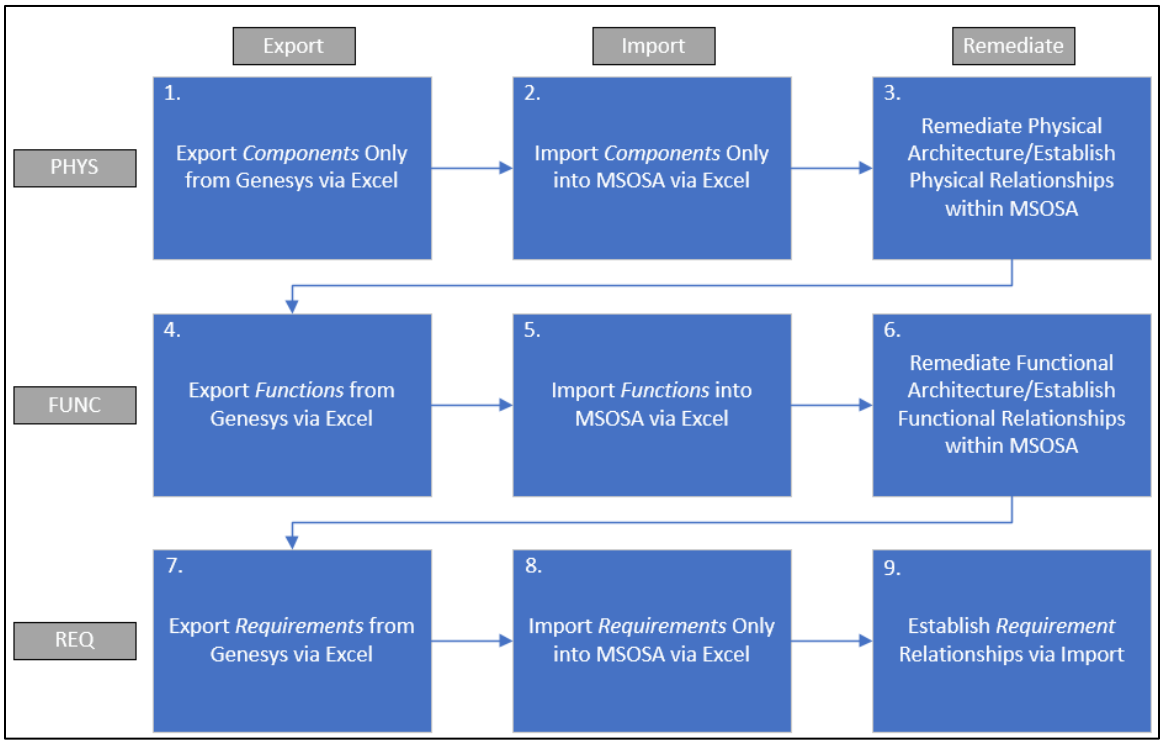


Figure 31. Process Summary for Genesys Export and MSOSA Import via MS Excel

Step 1 through Step 3 of Figure 31 focus on the physical aspects of the SoI. Step 1 first exported only the SoI model *components* from Genesys into Excel using the Genesys Excel Connector tool. The table definition for exporting the *components* from Genesys may be a simplified version of Figure 16, as only the *component* names and numbers are mapped in Step 2 (an observation made in retrospect of the experiment).

Step 2 then used the Excel file generated in Step 1 to import the *components* into MSOSA. The import followed a mapping shown in Figure 18, where the *components* were created as *blocks* in MSOSA. Since Step 1 did not capture any relationship data from Genesys, this Step 2 did not include the import of any relationships among physical entities, such as the physical hierarchy of the model. It also created empty *blocks* within MSOSA due to the structure of the *component* Excel file from Step 1. Step 3 is required to establish the physical decomposition in MSOSA as well as to remove empty *blocks*.

Step 3 removed the empty *blocks* by simply deleting the blank entities in MSOSA. After the deletion, there was still no hierarchical information in the model yet. To establish this, the *owner* SysML property was used to create parent-child relationships and generated the hierarchy of physical *blocks*. However, BDDs did not correctly generate yet and required further manual manipulation in MSOSA. To remedy this, the "directed composition" selection in the BDD was used to establish part properties and thereby successfully created physical decomposition in the BDD.

Step 4 through Step 6 then focus on the functional characteristics of the SoI. Step 4 exported only the *functions* via the Genesys Excel Connector, using a table definition like Figure 23. The entity and description properties, as well as the allocation and decomposition relationships, were exported from Genesys to create an importable Excel file for MSOSA for Step 5.

Step 5 took the export from Step 4 and imported it into MSOSA, using a mapping reflective of Figure 24. The Genesys *functions* were mapped to MSOSA *CallBehaviorAction* elements. The Genesys "entity," "description," "decomposes target" relationship, and "allocated to" relationship mapped to the MSOSA properties "name," "allocated to," "owner," and "documentation," respectively. This created *CallBehaviorAction* entities in MSOSA with allocation relationships to the *block* entities established by Steps 1 through 3. However, due to the one-to-one interpretation of MSOSA's mapping tool, *CallBehaviorAction* entities were duplicated and required manual user input to resolve, eliciting Step 6.

Step 6 removed the duplicated *CallBehaviorActions* created by Step 5 through simple deletion within MSOSA. However, prior to deleting each entity, their properties and relationships were recorded for future recreation. Once only unique *CallBehaviorActions* existed, the recorded properties and relationships that were deleted were recreated within MSOSA to bring about concordance and resolve the data duplication issue.

Step 7 through Step 9 then focus on the requirements of the SoI. The MSOSA 2021x User Manual (2020) details the steps to import requirements into MSOSA. The first step, Step 7, is to export the *requirements* from Genesys using a table definition that contains the *requirement* entity properties of "title," "description," and the "decomposes relationship." Then, per the MSOSA 2021x User Manual (2020), Step 7 modifies the Excel file to include "owner" and "id" columns. The additional columns enable MSOSA to interpret the decomposition of the *requirements*. "Id" is used to enumerate the *requirements*, and "owner" is used to show the parent-child relationship (decomposition) of the *requirements*. Once this step is complete, the Excel file was ready to be imported into MSOSA via Step 8.

Step 8 imported the *requirements* into MSOSA, also per the MSOSA 2021x User Manual (2020). The mapping between Genesys and MSOSA is one-to-one in this case, and therefore demonstrated no issues during the import process. However, no relationship information was captured during this step, warranting Step 9.

Step 9 added the relationships for the *requirements* in MSOSA by constraining the scope of the Genesys export file and importing it multiple times. In Genesys, *requirements* are the *basis of* a *function*, and a *function* may be *specified by requirements*, and *components* may also be *specified by requirements*. Therefore, three separate imports were warranted due to the one-to-one constraint in the MSOSA import tool. The first two imports were for *requirements* that *specify functions*, and for *requirements* that are the *basis of functions*. The third import established the *satisfies/satisfied by* relationship between *requirements* and *blocks*. These three steps correctly mapped requirements as *refining CallBehaviorActions* and as being *satisfied by blocks*.

Overall, the import of elements alone from Genesys to MSOSA was trivial, but issues exist with the end-to-end conversion method. MSOSA creates superfluous model elements with no data or duplicates when using tabulated data as the import method. At the same time, the import process with tabulated data does not efficiently allow for the capture of relationships. A user must manually remediate structural, behavioral and requirement relationships after import process. Furthermore, the MSOSA mapping tool is the only means to resolve semantic and ontological disparities between the tools and requires a user to define it. In this experiment, the user-defined mappings were assumed to be accurate per NSWC Crane's *Genesys to Cameo Conversion Process* (2020), which may not be accurate and would likely require further work to ensure both accuracy of the mappings and standardization across all users.

This study demonstrated the successful transfer of parts of the Genesys GSL model into MSOSA, but the combined limitations between relationship transfer and ontological mappings result in the MSOSA Excel import method as an overall insufficient means for transferring Genesys model data into MSOSA. The Genesys Excel connector and MSOSA Excel import function with associated mappings is effective for transferring basic model entities but is inefficient in transferring a complex system model. This scalability issue is a dramatic obstacle in the pursuit of interoperable MBSE models, as nearly all systems that warrant a model will involve complexity.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS AND CONCLUSIONS

This chapter summarizes the results of this study and draws conclusions from those results. The organization of this chapter includes a section for the results and a section for the conclusions.

A. RESULTS

This study identified conversion methods available to a normal user, the limitations observed during the GSL model experiment from Genesys to MSOSA, and the assessment of informational content after the conversion process.

This thesis was successful in replicating parts of the original Genesys GSL model in MSOSA, but it took a lot of user action to do the transfer and conversion. A review of the .xml files generated by Genesys showed that output model data is comprehensive and without omission. However, no simple means to transfer XML data into MSOSA exists. The Genesys Excel connector, in combination with the MSOSA Excel import option and its associated mapping tool, were the only means to transfer data from Genesys to MSOSA. All other supported file formats for data export and import between the tools were infeasible, making all other means to transfer data from Genesys to MSOSA also infeasible. The experimentation involving the GSL system model as a use-case revealed that there is exists limitations in data transfer between the two tools. Table 5 summarizes these limitations.

Table 5. Limitations Exchanging Data from Genesys to MSOSA Using the Genesys Excel Connector and MSOSA Mapping Tool

Observation	Limitation	Description
Data Duplication or Blank Data	One-to-One Mapping	Unable to correctly transfer data when there are one-to-many relationships between elements. Either MSOSA creates multiple copies of the same element, or MSOSA creates blank entities. In either case, the user must go into the model and correct the mapping.
Multiple Imports Required	One-to-One Mapping	Multiple imports are required and in a particular order in order to correctly transfer relationships between entities.
Order of Import	One-to-One Mapping	The order of the import is of significance during the migration process. A user may incorrectly omit relationship information if both entities for any single relationship do not already exist in MSOSA.
Importing Behavioral Modeling Data	Ontological Differences	There is a major modeling difference between how SDL and SysML model items and the interfaces between components to capture behavior. SDL defines item elements used as inputs/outputs/triggers, whereas SysML uses object and control flows, in addition to various combinations of object nodes, central buffer nodes, pins, etc. (each of which also have specialized types). The ontological mapping between SDL and SysML for system behavior therefore proved difficult to reproduce in MSOSA.
Post Migration Remediation	Inefficiency	The overall process was inefficient and time intensive. As every level of the data exchange process required a degree of manual remediation in MSOSA, the primary advantage of the Excel method seems to be only in establishing new entities.

The experiment revealed how Genesys and MSOSA have limited model data transfer capabilities through the Excel connector method. The Excel connector method facilities transfer of entities well but does not perform well in exchanging relationship information. The primary obstacle in the Excel method was the one-to-one interpretation implicit to the MSOSA mapping tool, as most models involving any degree of complexity will have one-to-many relationships. This effectively restricted the import method to the exchange of entities only. Imports of the physical and functional architectures were successful in MSOSA but required manual manipulation in the MSOSA tool after the

import process to be reflective of the original model. This would require a modeling effort or project to recreate a representative GSL model in MSOSA, which severely undercuts the value of this method.

With these results and using the LCIM (Tolk and Muguira 2003), this thesis assesses the interoperability between Genesys and MSOSA at Level 0. Documentation exists independently for the two tools for exchange of data, but there does not exist documentation specific to the interface between the tools since one does not exist. Since the only feasible method identified by this thesis to transfer data is through Excel, and each tool handles this connection differently, there is no effective means for interoperability. Therefore, this thesis arrives at an LCIM Level of 0 for this use-case.

B. CONCLUSIONS

Various conclusions were drawn based on the data and results of this experiment. This experiment demonstrated that no simple means to transfer data from Genesys to MSOSA exists. Model content can be transferred, but the means of doing the transfer is such that it wouldn't support, by any practical means, a digital thread in an enterprise such as the Navy. The only available data transfer method, as explored by this study, was essentially a "human-in-the-loop" approach using Excel as an intermediary. However, using the Excel connector to export model data from Genesys or import model data from a separate target tool demands meticulous management of the tabulated data during the import/export process to ensure correct ERA assignments. This can over-encumber a systems engineer and be time intensive depending on the complexity of the model.

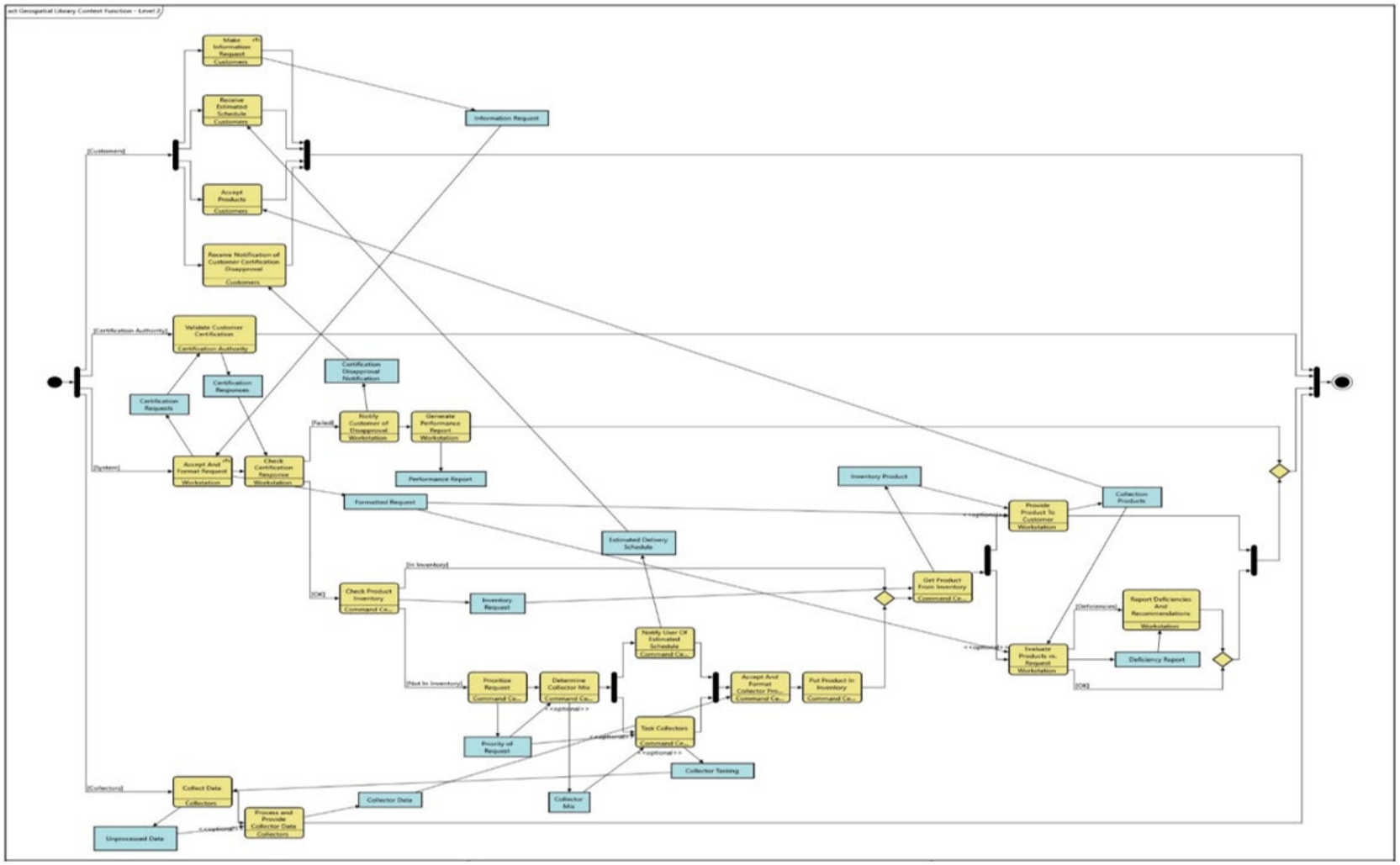
This experiment was successful in transferring 144 of 423 entities and 116 of 1,304 relationships from Genesys to MSOSA. The simple GSL sample model contained less than 2,000 entities and relationships and took the user approximately seven hours to adequately transfer only 17% of the original model's overall contents. In the case of complex military systems, system models may have upwards of 10,000 entities and relationships, or more. Assuming similar user proficiency, extrapolating this time estimate for 100% coverage and five times more data would require a time commitment of roughly 200 hours. This also assumes that the ontological mapping between SDL and SysML is accurate and defined.

To perform its experiment, this study assumed that the ontological mapping created by NSWC Crane is accurate and appropriate. Since the ontological mappings are not standardized, modelers may continue to develop system models with ontological disagreements, even with a technical solution available to transfer data. To resolve ontological differences consistently, the Navy and Marine Corps must establish a standard ontology that tool vendors and users may adopt and implement.

To support a digital thread with this study's approach, the Navy would have to invest in developing automated programs for extracting, transferring, and loading model data so it can piece together models from various sources. The methodology this study presents could theoretically be automated in a fashion like IBM Cloud's extract, transform, and load process, which "combines data from multiple data sources into a single, consistent data store that is loaded into a data warehouse or other target system," (IBM Cloud Education 2020) however this approach can be brittle as changes in either the source or target tool can break the automated process. It is likely more efficient at the XML data level, like Sandia's demonstration (Carroll et al. 2021), instead of using Excel as an intermediary and attempting to automate it.

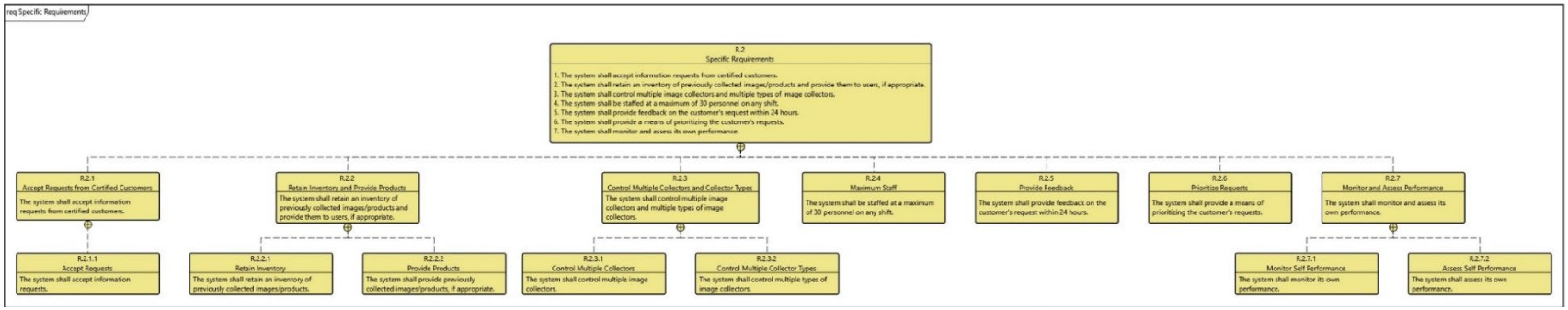
Until the community establishes a universally available technical Genesys-MSOSA solution and standard Navy and Marine Corps ontology, commands utilizing Genesys will continue to diverge from the Naval IME modeling path that their sister SYSCOMS are adopting. This conclusion may force Navy commands, such as NAWCADLKE, to continue modeling outside of the Naval IME and will hinder any future model federation goals. For these reasons, recommended future work includes the establishment of a SysML ontology for the Navy and Marine Corps to universally leverage and describe their systems, and the publication of a Genesys-MSOSA software solution that allows users to import XML data from Genesys to MSOSA. Since MSOSA was assumed to be synonymous with the Cameo Systems Modeler tool employed by the Naval IME, it is also recommended that the software solution be explored for specific use with Genesys and Cameo Systems Modeler.

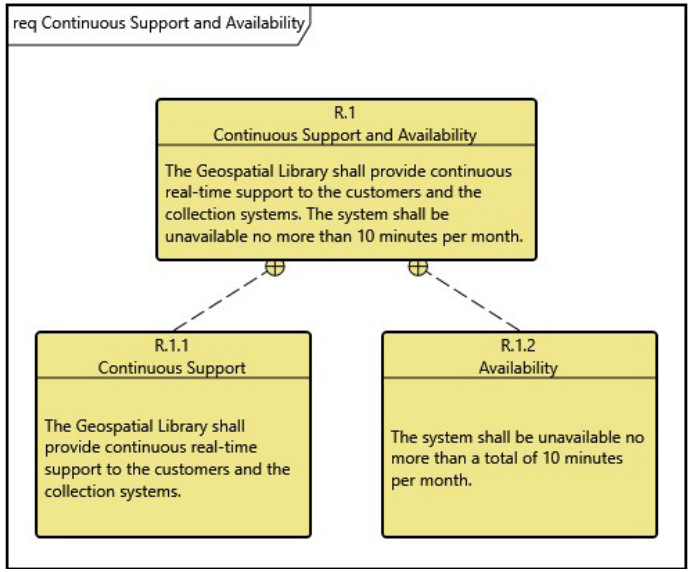
APPENDIX A


act Geospatial Library Context Function - Level 2
Make Information Request
Customers
Information Request
Receive Estimated Schedule
Customers
[Customers]
Accept Products
Customers
Receive Notification of Customer Certification Disapproval
Customers
[Certification Authority]
Validate Customer Certification
Certification Authority
Certification Disapproval Notification
Certification Responses
Certification Requests
Notify Customer of Disapproval
Workstation
Generate Performance Report
Workstation
[Failed]
[System]
Accept And Format Request
Workstation
Check Certification Response
Workstation
Performance Report
Formatted Request
Estimated Delivery Schedule
Inventory Product
Provide Product To Customer
Workstation
Collection Products
[In Inventory]
[OK]
Check Product Inventory
Command Ce...
Inventory Request
Get Product From Inventory
Command Ce...
Report Deficiencies And Recommendations
Workstation
[Deficiencies]
Evaluate Products vs. Request
Workstation
Deficiency Report
[OK]
Notify User Of Estimated Schedule
Command Ce...
[Not In Inventory]
Prioritize Request
Command Ce...
Determine Collector Mix
Command Ce...
Accept And Format Collector Pro...
Command Ce...
Put Product In Inventory
Command Ce...
Task Collectors
Command Ce...
Priority of Request
Collector Tasking
[Collectors]
Collect Data
Collectors
Collector Data
Collector Mix
Process and Provide Collector Data
Collectors
Unprocessed Data

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B


req Specific Requirements
R.2
Specific Requirements
1. The system shall accept information requests from certified customers.
2. The system shall retain an inventory of previously collected images/products and provide them to users, if appropriate.
3. The system shall control multiple image collectors and multiple types of image collectors.
4. The system shall be staffed at a maximum of 30 personnel on any shift.
5. The system shall provide feedback on the customer's request within 24 hours.
6. The system shall provide a means of prioritizing the customer's requests.
7. The system shall monitor and assess its own performance.
R.2.1
Accept Requests from Certified Customers
The system shall accept information requests from certified customers.
R.2.2
Retain Inventory and Provide Products
The system shall retain an inventory of previously collected images/products and provide them to users, if appropriate.
R.2.3
Control Multiple Collectors and Collector Types
The system shall control multiple image collectors and multiple types of image collectors.
R.2.4
Maximum Staff
The system shall be staffed at a maximum of 30 personnel on any shift.
R.2.5
Provide Feedback
The system shall provide feedback on the customer's request within 24 hours.
R.2.6
Prioritize Requests
The system shall provide a means of prioritizing the customer's requests.
R.2.7
Monitor and Assess Performance
The system shall monitor and assess its own performance.
R.2.1.1
Accept Requests
The system shall accept information requests.
R.2.2.1
Retain Inventory
The system shall retain an inventory of previously collected images/products.
R.2.2.2
Provide Products
The system shall provide previously collected images/products, if appropriate.
R.2.3.1
Control Multiple Collectors
The system shall control multiple image collectors.
R.2.3.2
Control Multiple Collector Types
The system shall control multiple types of image collectors.
R.2.7.1
Monitor Self Performance
The system shall monitor its own performance.
R.2.7.2
Assess Self Performance
The system shall assess its own performance.


req Continuous Support and Availability
R.1
Continuous Support and Availability
The Geospatial Library shall provide continuous real-time support to the customers and the collection systems. The system shall be unavailable no more than 10 minutes per month.
R.1.1
Continuous Support
The Geospatial Library shall provide continuous real-time support to the customers and the collection systems.
R.1.2
Availability
The system shall be unavailable no more than a total of 10 minutes per month.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Bone, Mary, Mark Blackburn, Benjamin Kruse, John Dzielski, Thomas Hagedorn, and Ian Grosse. 2018. "Toward an Interoperability and Integration Framework to Enable Digital Thread." *Systems* 6 (4): 46. http://dx.doi.org.libproxy.nps.edu/10.3390/systems6040046.

Carroll, Ed. 2019. "Model Interoperability/Credibility - Demonstration Results." Presentation, Sandia National Laboratories, Albuquerque, NM, February. https://www.osti.gov/servlets/purl/1645980.

Carroll, Ed, Carlos Tafoya, Jonathan Compton, Akinli Cengiz, and Jason Jarosz. 2021. *Retaining Systems Engineering Model Meaning Through Transformation: Demo 2.* Technical Report SAND2021-2143. Albuquerque, NM: Sandia National Laboratories.

Delligatti, Lenny. 2014. *SysML Distilled: A Brief Guide to the Systems Modeling Language*. Upper Saddle River, NJ: Addison-Wesley.

Department of Defense. 2018. *Department of Defense Digital Engineering Strategy*. Washington, DC: Office of the Deputy Assistant Secretary of Defense for Systems Engineering. https://ac.cto.mil/wp-content/uploads/2019/06/2018-Digital-Engineering-Strategy_Approved_PrintVersion.pdf.

Department of the Navy. 2020. United States Navy and Marine Corps Digital Engineering Transformation Strategy. Washington, DC: Department of the Navy. https://ac.cto.mil/wp-content/uploads/2019/06/2018-Digital-Engineering-Strategy_Approved_PrintVersion.pdf.

———. 2021. Digital Engineering. NAVWAR Instruction 5401.9. San Diego, CA: Department of the Navy.

"Genesys 4.1 DOORS Connector Guide." 2016. Vitech Corporation. http://www.vitechcorp.com/support/documentation/genesys/400/DOORSConnectorGuide.pdf.

"Genesys 6.0 Getting Started with the Genesys API." 2018. Vitech Corporation. http://www.vitechcorp.com/support/documentation/genesys/600/GettingStartedWithTheGENESYSAPI.pdf.

"Genesys 2021 R2 Systems Engineering Guided Tour." 2021. Zuken Vitech Inc.

Genesys to Cameo Conversion Process. 2020. Technical Report. Crane, IN: NSWC Crane Division.

Giachetti, Ronald E. 2015. “Evaluation of the DoDAF Meta-Model’s Support of Systems Engineering.” *Procedia Computer Science* 61: 254–60. https://doi.org/10.1016/j.procs.2015.09.208.

Giachetti, Ronald E., and Warren Vaneman. 2021. “Requirements for a System Model in the Context of Digital Engineering.” In *2021 IEEE International Systems Conference (SysCon)*, 1–7. https://doi.org/10.1109/SysCon48628.2021.9447088.

Holt, Jon, and Simon Perry. 2019. *SysML for Systems Engineering - A Model-Based Approach*. 3rd ed. Institution of Engineering and Technology (The IET). https://app.knovel.com/hotlink/toc/id:kpSMLSEA0L/sysml-systems-engineering/sysml-systems-engineering.

IBM Cloud Education. 2020. “ETL (Extract, Transform, Load).” April 28, 2020. https://www.ibm.com/cloud/learn/etl#toc-what-is-et-xeCDpL69.

INCOSE. 2007. *INCOSE Systems Engineering Vision 2020.* Technical Report INCOSE-TP-2004-004-02. International Council on Systems Engineering (INCOSE).

———. 2022. “INCOSE Systems Engineering Vision 2035.” International Council on Systems Engineering (INCOSE). https://www.incose.org/about-systems-engineering/se-vision-2035.

Johnson, Joan L. 2020. “Digital/Systems Engineering Transformation Working Group Charter.” Official Memorandum. Washington, DC: Department of Defense.

“Key Concepts.” n.d. Vitechcorp.Com. Accessed April 9, 2022. https://www.vitechcorp.com/resources/core/onlinehelp/desktop/topic.htm#t=Key_Concepts.htm.

“Magic Systems of Systems Architect 2021x User Manual.” 2020. No Magic, Inc.

Merriam-Webster. n.d. “Semantics.” Accessed March 26, 2022. https://www.merriam-webster.com/dictionary/semantics.

Moschler, Joe. 2019. “Defense Acquisition Magazine.” *Defense Acquisition University* XLVIII (No. 5, DAU 270): 51.

Murdock, Jaimie, and Edward Carroll. 2021. *Simplifying and Visualizing the Ontology of Systems Engineering Models.* SAND2021-7079, 1814061, 698015. https://doi.org/10.2172/1814061.

Pilato, Jeff. 2022. “Breaking Through System Design Tool Dependency with Automatic Model Transformation.” Virtual Naval D/SET Presentation, August 10, 2022. https://wiki.lift.mhpcc.hpc.mil/confluence/display/ND/Breaking+Through+System+Design+Tool+Dependency+with+Automatic+Model+Transformation.

Tolk, Andreas, Saikou Diallo, Jose Padilla, and Charles Turnista. 2012. “How Is M&S Interoperability Different From Other Interoperability Domains?” *M&S Journal* 7 (3): 5–14.

Tolk, Andreas, and James A. Muguira. 2003. “The Levels of Conceptual Interoperability Model.” Old Dominion University.

Vaneman, Warren. 2022. “MBSE Structures.” Class notes for SE4930: Model-Based Systems Engineering, Naval Postgraduate School, Monterey, CA. https://cle.nps.edu/access/content/group/db46b2d2-2c3a-462a-9fc3-53b863247315/Module%204%20-%20Notes/M04-2%20Model%20Structure.pdf.

“What Is SysML?” n.d. OMGSysML. Accessed April 9, 2022. https://www.omgsysml.org/what-is-sysml.htm.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California