



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

**IMPLEMENTING A MODEL-BASED SYSTEMS ENGINEERING
LAND CONSTRUCT FOR THE MARINE CORPS**

by

Dr. Warren Vaneman, ESEP, Systems Engineering, PI
Mrs. Corina White, CSEP, Systems Engineering, Co-PI
Prof. Ronald Carlson, Systems Engineering, Researcher
Mr. Christopher Wolfgeher, Systems Engineering, Researcher
Mr. Christopher Ritter, Systems Engineering, Student
Mr. Geraldo Rodriguez Melo, Systems Engineering, Student
Mr. Henry Sulca, Systems Engineering, Student

October 2023

Approved for public release. Distribution is unlimited.

Prepared for: Marine Corps Systems Command (MARCORSYSCOM).
This research is supported by funding from the Naval Postgraduate School, Naval
Research Program (PE 0605853N/2098).
NRP Project ID: NPS-23-M071-A

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

1. REPORT DATE		2. REPORT TYPE		3. DATES COVERED	
10/21/2023		Technical report		START DATE 10/24/2022	END DATE 10/20/2023
4. TITLE AND SUBTITLE Implementing a Model-Based Systems Engineering Land Construct for the Marine Corps					
5a. CONTRACT NUMBER		5b. GRANT NUMBER		5c. PROGRAM ELEMENT NUMBER 0605853N/2098	
5d. PROJECT NUMBER NPS-23-M071-A		5e. TASK NUMBER		5f. WORK UNIT NUMBER	
6. AUTHOR(S) Warren Vaneman, Corina White, Ronald Carlson, Christopher Wolfgeher, Christopher Ritter, Geraldo Rodriguez Melo, Henry Sulca					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School 1 University Circle Monterey, CA 93943-5000				8. PERFORMING ORGANIZATION REPORT NUMBER NPS-SE-23-004	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Postgraduate School, Naval Research Program; Marine Corps Systems Command				10. SPONSOR/MONITOR'S ACRONYM(S) NRP; MARCORSYSCOM	11. SPONSOR/MONITOR'S REPORT NUMBER(S) NPS-23-M071-A
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The FY23 NRP research, conducted in conjunction with MCSC, analyzed and refined the land domain ontology, and conceptual data model (CDM), developed in the FY22 NRP research entitled, "Developing a Model-Based Systems Engineering Land Domain Construct for Marine Corps Systems Command," with three goals. The first goal analyzes how the land domain ontology and CDM may be implemented to components that interface with land domain. This goal validated the FY22 NRP ontology, and CDM, as mathematically correct by modeling it in the Protégé-OWL ontology tool. This validation illustrated the ontology's utility to serve as the structure for an authoritative source of truth. This goal also supports the development of the MCSC Entity Re-Use Library (the first known authoritative source of truth in DoD) by using the previously developed ontology as the libraries structure. The second goal explores options to exchange data through the representation of complex data and the relationships between the data. This goal explored how various diagram types are implemented in various modeling languages and MBSE tools and demonstrates how data can be shared between each. A key feature of the Entity Re-Use Library is data sharing among disparate tools. This research explored a new approach to data sharing by defining a set of features common to the three tools that were considered. The third goal focuses on evaluating and updating the existing ontology and CDM with corresponding land domain and the Naval Operational Architecture from a mission scenario perspective. This goal explored how the ontology and CDM can be applied to the structuring of a relational database being developed by one of the Naval Surface Warfare Centers. This goal is illustrated that the ontology can be applied to more than MBSE Tool derived data but is applicable to other data sources.					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:				17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 77
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			
19a. NAME OF RESPONSIBLE PERSON Corina White				19b. PHONE NUMBER (Include area code) (904) 658-0877	

THIS PAGE INTENTIONALLY LEFT BLANK

**NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000**

Ann E. Rondeau
President

Scott Gartner
Provost

The report entitled “Implementing a Model-Based Systems Engineering Land Construct for the Marine Corps” was prepared for Marine Corps Systems Command and funded by Naval Postgraduate School, Naval Research Program (PE 0605853N/2098).

Further distribution of all or part of this report is authorized.

This report was prepared by:

VANEMAN.WARR
EN.K.1030886115

Digitally signed by
VANEMAN.WARREN.K.1030886
115
Date: 2023.12.07 14:15:38 -05'00'

Warren Vaneman
Principal Investigator

C.White_local
|

Digitally signed by
C.White_local
Date: 2023.12.07
10:09:13 -08'00'

Corina White
Co-Principal Investigator

Reviewed by:

YAKIMENKO.OLE
G.A.1252335508

Digitally signed by
YAKIMENKO.OLEG.A.1252335508
DN: c=US, o=U.S. Government, ou=DoD,
ou=PKI, ou=USN,
cn=YAKIMENKO.OLEG.A.1252335508
Date: 2023.12.07 12:59:36 -08'00'

Oleg Yakimenko, Chairman
Systems Engineering

Released by:

SMITH.KEVIN.
B.1230026366

Digitally signed by
SMITH.KEVIN.B.1230026366
Date: 2023.12.07 13:47:32
-08'00'

Kevin B. Smith
Vice Provost for Research

THIS PAGE INTENTIONALLY LEFT BLANK

I. EXECUTIVE SUMMARY

PROJECT SUMMARY

The FY23 NRP research, conducted in conjunction with MCSC, analyzed and refined the land domain ontology, and conceptual data model (CDM), developed in the FY22 NRP research entitled, “Developing a Model-Based Systems Engineering Land Domain Construct for Marine Corps Systems Command,” with three goals. The first goal analyzes how the land domain ontology and CDM may be implemented to components that interface with land domain. This goal validated the FY22 NRP ontology, and CDM, as mathematically correct by modeling it in the Protégé-OWL ontology tool. This validation illustrated the ontology’s utility to serve as the structure for an authoritative source of truth. This goal also supports the development of the MCSC Entity Re-Use Library (the first known authoritative source of truth in DoD) by using the previously developed ontology as the libraries structure.

The second goal explores options to exchange data through the representation of complex data and the relationships between the data. This goal explored how various diagram types are implemented in various modeling languages and MBSE tools and demonstrates how data can be shared between each. A key feature of the Entity Re-Use Library is data sharing among disparate tools. This research explored a new approach to data sharing by defining a set of features common to the three tools that were considered.

The third goal focuses on evaluating and updating the existing ontology and CDM with corresponding land domain and the Naval Operational Architecture from a mission scenario perspective. This goal explored how the ontology and CDM can be applied to the structuring of a relational database being developed by one of the Naval Surface Warfare Centers. This goal is illustrated that the ontology can be applied to more than MBSE Tool derived data but is applicable to other data sources.

BACKGROUND

The Department of Defense (DoD) Digital Engineering (DE) Strategy calls for the establishment of a “supporting infrastructure and ecosystem to perform activities, collaborate and communicate system data, using authoritative sources of system data and models as a continuum across engineering disciplines across the acquisition lifecycle” (DASD SE 2018). This is a noble, but significant, goal given the breadth of DE spans no less than eight modeling domains to include: modeling and simulation; model-based testing; model-based engineering; model-based systems engineering; model-based sustainment; model-based manufacturing; model-based requirements; and, model-driven software development.

The overarching goals for this continuing research were derived from the DoD Digital Engineering Strategy. Specifically, Goal 2 – “Provide an enduring, authoritative source of truth”, and Goal 4 - “Establish a supporting infrastructure and environments to perform activities, collaborate, and communicate across stakeholders” (DASD SE 2018).

The FY22 NRP focused on three research areas. First, the importance of a generic and minimal ontology that comprehensively represents the system across the lifecycle. A proposed minimal ontology was analyzed against several representative data dictionaries, which resulted in a “Rosetta Stone” that allows for translation of data elements represented in different models, and modeling tools, by reducing the data to a common baseline. Second, the generic system defined

by the proposed ontology was related to a CDM to show how the data and the entity classes were related. This is important because the complete system model is often developed by different organizations, using different modeling languages, ontologies and tools. This effort highlighted where the data interfaces occurred, and where exchange standards needed to be defined. Third, the concept of an authoritative source of truth was analyzed, and a notional schema was defined. Finally, that research used the ontology to segment the data types by the authoritative organizations.

Data sharing and interoperability between systems engineering tools is not a new concept and has been implemented with mixed results. Today, there is no tool which can fully ingest another tool's data model completely. Nevertheless, an increase in interoperability promises to reduce risk in the design and development of complex naval systems to empower the Nation's warfighters as fast and as cost effective as possible. This research advances the overarching goals by further exploring the use of ontologies to serve not only as a means to identify where data sharing opportunities exist, by defining a meta-model that can provide the structure for an authoritative source of truth (Vaneman et al. 2022). This research also identifies and evaluates data and tool integration alternatives through an exploration of approaches that represent the complex knowledge about entities, the relationships between entities, and additional information needed to exchange data that will provide a more holistic and comprehensive view of the system to enable better decisions.

FINDINGS AND CONCLUSIONS

Goal 2 of the DoD Digital Engineering Strategy (DASD SE, 2018) calls for an “authoritative source of truth,” to serve as a meta-model to show how disparate types of data, in various format, derived from different modeling language, model-based tools, and developed by different organizations come together to represent the system holistically. The first goal of the FY23 research was to analyze how the land domain ontology may be implemented to components that interface with land domain. The starting point for this goal was using the ontology, and CDM, developed during the FY22 NRP. The ontology previously developed was in MS Excel and MS PowerPoint respectively. This focus area validated the FY22 NRP ontology by modeling it in the Protégé-OWL ontology tool.

Protégé-OWL supports the creation, editing, visualization, and management of ontologies. It allows for the establishment of taxonomy hierarchy as representation of the ontology, as well as the representation of entity relationships, and for the visualization of the ontology. Most importantly, Protégé-OWL is based on the mathematical properties of set theory, therefore proved that the ontology was mathematically consistent. This validation illustrates the ontology's utility to serve as the structure for an authoritative source of truth.

The purpose of Goal 4 of the DoD Digital Engineering Strategy is the notion that data will be shared across the digital ecosystem, regardless of the modeling language or tool that was used to generate the data. The predominate thinking within DoD is to standardize to a single tool. Each modeling tool has its strengths and weaknesses; therefore, multiple tools are needed.

This research explored options to exchange data through the representation of complex data and the relationships between the data. Two methods to exchange data between tools exist. First, data can be translated using a XML Metadata Exchange (xmi). However, inconsistencies between tools exist even when using the same modeling language. We explored the System Modeling Language (SysML 2.0) Application Programming Interface (API), and found that this API will

solve many of the data interoperability issues but is not the “holy grail as advertised. Due to tool vendor implementations, the API will likely not perform well across tools.

The second method is to define a set of common features among the tools. The Architecture Interoperability Method for Systems Engineering (AIM SE) was developed, not as a new tool or modeling language, but as a means to define the maximum set of features. AIM SE was tested using three model-based tools, and proved to be successful using the SysML diagrams.

Lastly, the ontology was evaluated using a mission scenario applied to defining the structure of a relational database (RDB). This mapping incorporated the class taxonomy to define the structure of the different types of tables, columns, rows, fields, keys, and data. This demonstrated the versatility of the ontology to be applied to data that is outside the realm of data generated by MBSE tools.

RECOMMENDATIONS FOR FURTHER RESEARCH

This research has significantly advanced digital engineering by demonstrating an ontology that can be used as the basis to structure an authoritative source of truth, that contains data from disparate modeling languages, tools, and database type. Furthermore, the notion of data sharing between disparate modeling-languages and tools was demonstrated using AIM SE. In spite of these advances, more work is needed.

This research has demonstrated that the ontology is mathematically consistent and can theoretically be used as a structure for authoritative source of truth. The ontology was modeled in Protégé-OWL, which claims to be modeling language, and tool, agnostic. However, validating this claim was not within the scope of this research. Further research should validate that Protégé OWL can be used with disparately modeling languages and tools and will function as the implemented structure of the authoritative source of truth.

AIM SE defined a maximum set of modeling-language features that can be used to exchange data between different modeling tools. AIM SE is a promising approach but has only been demonstrated with a single modeling language and three tools. However, digital engineering data is developed from many different modeling languages and tools. As such, the scalability of AIM SE needs to be explored with a broader set of modeling languages and modeling tools.

This research has demonstrated that the ontology, as modeled in Protégé-OWL can be used to define the structure of a relational database (RDB). While many RDBs contain system data, there is a lack of clarity on the role of RDBs within the authoritative source of truth and should be explored. Using the ontology to define a structured database is a promising development, however, the true benefits have not defined, and should be explored.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. EXECUTIVE SUMMARY	V
PROJECT SUMMARY.....	V
BACKGROUND	V
FINDINGS AND CONCLUSIONS	VI
RECOMMENDATIONS FOR FURTHER RESEARCH	VII
II. INTRODUCTION.....	1
A. OVERVIEW	1
B. BACKGROUND	2
C. RESEARCH FOCUS AREAS AND APPROACH.....	2
1. Analyze how the land domain ontology and CDM may be implemented to components that interface with land domain.	3
2. Explore options to exchange data through the representation of complex data and the relationships between the data.	3
3. Evaluate and update the existing ontology and CDM with corresponding land domain and the Naval Operational Architecture from a mission scenario perspective.	3
III. ANALYZING THE ONTOLOGY	4
A. OVERVIEW	4
1. Land Domain Ontology and Conceptual Data Model.....	4
2. Land Domain Ontology and Conceptual Data Model.....	5
B. ONTOLOGY ANALYSIS.....	7
C. SUMMARY	8
IV. DATA EXCHANGE WITHIN DIGITAL ENGINEERING	9
A. OVERVIEW	9
B. MBSE TOOLS USED IN AIM SE	9
C. AIM SE DIAGRAMS	10
3. Structure Diagrams	10
a. <i>Package Diagram</i>	10
b. <i>Block Definition Diagram</i>	11
c. <i>Internal Block Definition Diagram</i>	13
4. Functional Behavior Diagrams.....	15
a. <i>Activity Diagram</i>	15
b. <i>State-Machine Diagram</i>	17
5. Requirements Diagram	18
6. Summary.....	19
V. USING ONTOLOGIES FOR RELATIONAL DATABASES	21
A. OVERVIEW	21
B. APPLYING THE ONTOLOGY TO A RELATIONAL DATABASE	21
1. Relational Database	21
2. Ontology Development	22
3. Mapping OWL to Relational Database.....	25
VI. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS	29

A. OVERVIEW	29
FINDINGS AND CONCLUSIONS.....	29
B. RECOMMENDATIONS FOR FURTHER RESEARCH	31
APPENDIX A: ONTOLOGY	34
APPENDIX B: ONTOLOGY WEB LANGUAGE (OWL) FUNDAMENTALS.....	40
APPENDIX C: SYSML KERML TO AIM SE COMPARISION.....	45
APPENDIX D: DDL FILE.....	48
LIST OF REFERENCES.....	55
INITIAL DISTRIBUTION LIST	60

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1. Conceptual Ontology (Cambridge Semantics. n.d.).	4
Figure 2. Package Diagrams Across the MBSE Tools.	11
Figure 3. Block Definition Diagrams Across the MBSE Tools.	12
Figure 4. Internal Block Definition Diagrams Across the MBSE Tools.	14
Figure 5. Activity Diagrams Across the MBSE Tools.	16
Figure 6. State-Machine Diagrams Across the MBSE Tools.	18
Figure 7. Ontology to Relational Database Overview.	22
Figure 8. OWL Classes and Taxonomy of T&E Information System.	24
Figure 9. OWL Tree Diagram of Analyst Relationships.	24
Figure 10. Ontology to RDM Transformation Process (Ernestas and Nemuraite, 2006).	26

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1.	Ontology New Entities, Sub-entities, and Definitions.....	5
Table 2.	Conceptual Ontology (Cambridge Semantics. n.d.).	6
Table 3.	OWL Components (Adapted from W3C, 2012).....	7
Table 4.	Comparison of Block Definition Diagram Implementations.....	12
Table 5.	Comparison of Internal Block Definition Diagrams.....	14
Table 6.	Comparison of Activity Diagrams.....	16
Table 7.	Comparison of State-Machine Diagrams.....	18
Table 8.	Comparison of Requirements Diagrams.....	19
Table 9.	Notional Requirements for the T&E Information System.....	23
Table 10.	OWL to Relational Database Mapping T&E Information System.....	27

THIS PAGE INTENTIONALLY LEFT BLANK

II. INTRODUCTION

A. OVERVIEW

The Department of Defense (DoD) Digital Engineering (DE) Strategy calls for the establishment of a supporting infrastructure and ecosystem to perform activities, collaborate and communicate system data, using authoritative sources of system data and models as a continuum across engineering disciplines across the acquisition lifecycle (DoD 2018). This is a noble, but significant, goal given the breadth of DE spans no less than eight modeling domains to include: modeling and simulation; model-based testing; model-based engineering; model-based systems engineering; model-based sustainment; model-based manufacturing; model-based requirements; and, model-driven software development.

Data sharing and interoperability between systems engineering tools is not a new concept, and has been implemented with mixed results. In 2006, Airbus lost \$6.1 billion and added four years to their A380 production schedule due to data interoperability issues between model-based design tools (Wong 2006). In 2003, the Standard for the Exchange of Product information (STEP) Systems Engineering Project (AP233), led by PDES, Inc with participation from the Internal Council on Systems Engineering (INCOSE), Object Management Group (OMG), tool vendors (e.g., IBM, Vitech, and I-Logix), government (e.g., NASA and NIST), and industry (e.g., Rockwell Collins and Boeing) attempted to develop data standards across a set of systems engineering modules. However, the only success was the demonstrated requirements exchange between the Unified Modeling Language (UML), IBM's Dynamic Object-Oriented Requirements System (DOORS), and Vitech Core.

Today, there is no tool which can fully ingest another tool's data model completely. Nevertheless, an increase in interoperability promises to reduce risk in the design and development of complex naval systems to empower the Nation's warfighters as fast and as cost effective as possible. This research identifies and evaluates data and tool integration alternatives through an exploration of approaches that represent the complex knowledge about entities, the relationships between entities, and additional information needed to

exchange data that will provide a more holistic and comprehensive view of the system to enable better decisions.

B. BACKGROUND

The FY23 NRP research, conducted in conjunction with MCSC, analyzed and refined the land domain ontology, CDM, developed in the FY22 NRP research with three goals. First, analyze how the land domain ontology and CDM may be implemented to components that interface with land domain. This goal validated the FY22 NRP ontology, and CDM, as mathematically correct by modeling it in the Protégé-OWL ontology tool. This validation illustrated the ontology's utility to serve as the structure for an authoritative source of truth. This goal supports the development of the MCSC Entity Re-use Library (the first known authoritative source of truth in DoD) by using the previously developed ontology as the libraries structure.

Second, explore options to exchange data through the representation of complex data and the relationships between the data. This goal explored how various diagram types are implemented in various modeling languages and MBSE tools and demonstrates how data can be shared between each. A key feature of the Entity Re-Use Library is data sharing among disparate tools. This research explored a new approach to data sharing by defining a set of features common to the three tools that were considered.

And third, evaluate and update the existing ontology and CDM with corresponding land domain and the Naval Operational Architecture from a mission scenario perspective. This goal explores how the ontology and CDM can be applied to the structuring of a relational database being developed by one of the Naval Surface Warfare Centers. This goal is illustrated that the ontology can be applied to more than MBSE Tool derived data, but is applicable to other data sources.

C. RESEARCH FOCUS AREAS AND APPROACH

The following are the fundamental focus areas and approach of this research:

- 1. Analyze how the land domain ontology and CDM may be implemented to components that interface with land domain.**

This focus area validates the FY22 NRP ontology and CDM by modeling it in an ontology tool. This validation illustrates the ontology's utility to serve as the structure from an authoritative source of truth. This focus area is highlighted in Chapter 2.

- 2. Explore options to exchange data through the representation of complex data and the relationships between the data.**

This focus area explores how various diagram types are implemented in various modeling languages and MBSE tools, and demonstrates how data can be shared between each. This focus area is highlighted in Chapter 3.

- 3. Evaluate and update the existing ontology and CDM with corresponding land domain and the Naval Operational Architecture from a mission scenario perspective.**

This focus area explores how the ontology and CDM could be applied to the structuring of a relational database being developed by one of the Naval Surface Warfare Centers. This focus area is highlighted in Chapter 4. While each focus area is related, there is disparity in the way each was treated as the research progressed. As such, each will be treated in separate chapters with a brief literature review at the beginning of each. This report concludes with findings and recommendations in Chapter 5.

III. ANALYZING THE ONTOLOGY

A. OVERVIEW

Ontology provides a foundational framework for understanding the structure of the world and the relationships between entities. In today's increasingly digital and interconnected world, ontology has assumed important significance beyond just philosophy and theoretical concepts. Ontologies establish a unified understanding of information from the real world based on concepts and relationships, represented in specifications, and expressed in language as illustrated in **Figure 1x** (Cambridge Semantics. n.d.).

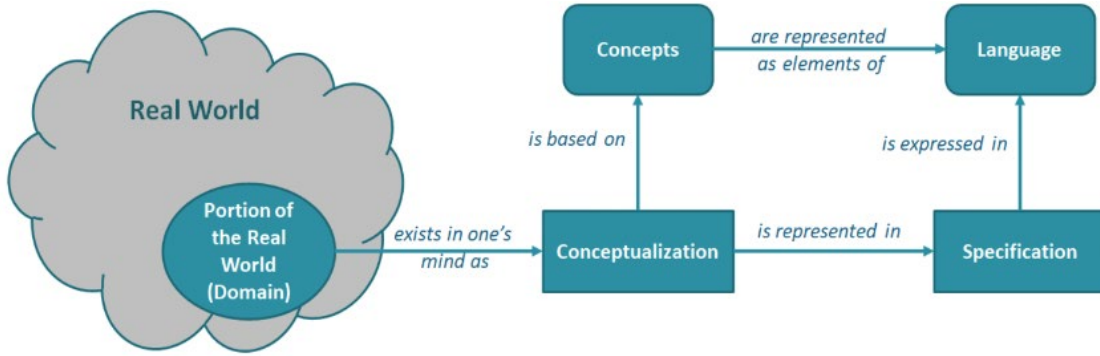


Figure 1. Conceptual Ontology (Cambridge Semantics. n.d.).

1. Land Domain Ontology and Conceptual Data Model

The FY23 research further analyzed and validated the ontology by applying it to a generic mission engineering problem and by modeling it in Protégé-OWL. MCSC further provided analysis by modeling the ontology to using the System Modeling Language (SysML) with the Cameo System Modeler. As a result of this analysis, the ontology was expanded by 3 entities, and their corresponding relationships. The new entities, sub-entities, and definitions are shown in Table 1. The full set of entities is shown in Appendix A.

With these additions the total number of entities increases to 15 entities total. Given that the entities $n \times n$ possible relationships, thus the total number of combinations within the ontology is approximately $15!$ or 1.31×10^{12} .

Table 1. Ontology New Entities, Sub-entities, and Definitions (LML 2015).

Entity	Sub-Entity	Definition
<i>Connection</i>	Conduit	A Conduit entity specifies the means for physically transporting Input/Output entities between Asset entities. It has limitations (attributes) of capability and latency.
	Logical	A Logical entity represents the abstraction of the relationship between two entities (e.g., Asset entities with the type “Entity”)
<i>Decision</i>	Strategic Decision	Strategic decisions are those decisions that have an influence over years, decades, and even beyond the lifetime of the project. Once a strategic decision is made, it is very unlikely to be altered in the short term.
	Routine Decision	A routine decision is a choice that is frequently made with a minimum of uncertainty and involve a standard procedure. This means that the decision maker has tools available to help guide them through the decision. These tools may include rules, policies, procedures, precedents, or computational techniques.
	Programmed Decision	Programmed decision-making involves those decisions that already have a plan or rule in place and is used to reach a solution or conclusion. In other words, managers have already made such decisions before and it's a repetitive and routine process.
	Policy Decision	Policy decisions are those that determine how we will act in the future. How will we do this? What will guide our actions? A policy decision tells us how to make choices in order to act in alignment with our purposes and goals.
	Operational Decision	Operational decisions are short-term choices that are typically made on a weekly, daily, or hourly basis.
	Organizational Decision	Organizational decision making is the process by which one or more organizational units make a decision on behalf of the organization
<i>Input/Output</i>	Input	Input - Resources such as people, raw materials, energy or information that are put into a system or a process to obtain a desired output.
	Output	Output - Goods or services produced by a system or a process i.e. the result of a process.

2. Land Domain Ontology and Conceptual Data Model

According to the World Wide Web Consortium (W3C), an Ontology Web Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things (W3C 2013). The Semantic Web is an extension of the World Wide Web, aimed at making data on the internet

understandable and usable not just by humans but also by machines. It provides a common framework that allows data to be shared and reused across application, enterprise, and communities. The goal is to enable computers to understand the meaning, or semantics, of information without human intervention to allow more intelligent querying of information (Obitko 2007). The initial OWL specification featured three types: OWL Lite, OWL Description Logic, and OWL Full. However, in 2009, the W3C released OWL 2 which consolidated the language to offer a more practical, optimized, and usable solution for diverse applications, addressing the limitations and shortcomings of the original design. The latest OWL 2 version supersedes the initial implementation as the ontology language of the semantic web (Cambridge Semantics n.d.). Figure 2 shows the Semantic Web architecture.

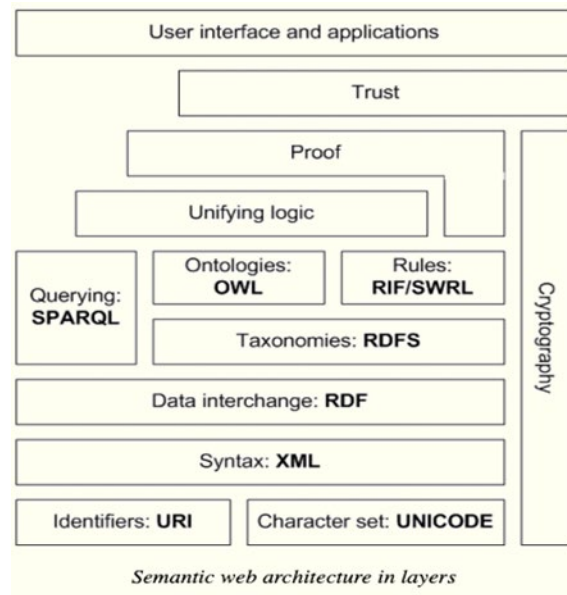


Table 2. Conceptual Ontology (Cambridge Semantics. n.d.).

OWL is a declarative language, which means it describes a model using logic and structure without resorting to using a programming language. The components of OWL are categorized in Table 2 and are further discussed in Appendix B.

B. ONTOLOGY ANALYSIS

The most used tool for developing OWL ontologies is Protégé. Protégé is a free, open-source tool developed by Stanford University. The tool supports the creation, editing, visualization, and management of OWL 2 ontologies. It allows for ontology visualization, such as the generation of the conceptual data model (CDM) via the visual notation for OWL ontologies (VOWL) plugin (Parlar 2019).

The Protégé-OWL modeling methodology consisted of three primary steps. First, the Protégé taxonomy was modeled using the entities, sub-entities, and datatypes from the conceptual ontology shown in Appendix A. This step accurately represents the semantics. The class (entity and sub-entity) hierarchy within Protégé-OWL is shown in Figure 3.

Table 3. OWL Components (Adapted from W3C, 2012).

Components	Description
Axiom	The main component of OWL2. Axioms are “statements or declarations in the ontology that asserts some form of knowledge or constraint about the classes, properties, and individuals within the ontology.”
Classes	“Defines a set of individuals” that share common properties
Datatypes	“ <i>Datatypes</i> are entities that refer to sets of data values. Thus, datatypes are analogous to classes, the main difference being that the former contain data values such as strings and numbers, rather than individuals.”
Object Properties	Connects individuals and forms relationships
Data Properties	“Connect individuals with datatypes. In some knowledge representation systems, functional data properties are called attributes.”
Individuals/ Instances	“Represent actual objects from the domain.”
Annotations	Descriptions of Axioms (metadata)

The second step was to data type properties by modeling the relationships between an individual and an attribute. Due to the time constraints, only the relationships with the CDM were modeled, and not all relationships depicted in Table 2.

In this step the corresponding relationships were added to the entities that were modeled step 1. Protégé-OWL captures various properties as shown in example Figure 4. This capability is important because the data in an authoritative source of truth will have many identifying properties such as data type, relationships, and a numbering schema, just to name a few.

The third step want to create the object properties. Examination of relationships the applicable relationships from Table 2, the CDM identified 15 pairs of inverse relationships and one symmetric relationship. As shown in Table 2, essentially all relationships are inverse relationships. Figure 5 also shows that *related to* has a symmetric relationship. Therefore, *related to* has a relationship of *related to*. When symmetric relationships exist, the name of the inverse relationship is not listed on the right side, only shown in the body. A discussion the OWL Properties is included in Table B.

Steps 1-3 yield a complete set of property assertions – the linking of data property and object property assertions. Figure 6 shows the property assertions for the sub-entity **System**, which is a subordinate of the entity *Asset*. Figure 7 a subset graphical representation relationships that were captured in the CDM.

C. SUMMARY

Using Protégé-OWL, this research has validated that the ontology developed during the FY22 NRP is mathematically consistent. A logical question at this point may be, “So what?”

Goal 2 of the DoD Digital Engineering Strategy (DoD, 2018) calls for an “authoritative source of truth,” to serve as a meta-model to show how disparate types of data, in various format, derived from different modeling language, model-based tools, and developed by different organizations come together to represent the system holistically. This research has made tremendous steps to achieving that goal. This research is specifically relevant to MCSC as they develop the Entity Re-Use Library – an authoritative source of truth!

Protégé-OWL is advertised to be language agnostic. Proof of Protégé-OWL interacts with various modeling languages and model-based tools was out of the scope of this research, but is the next logical step.

IV. DATA EXCHANGE WITHIN DIGITAL ENGINEERING

A. OVERVIEW

This chapter examines options to exchange data through the representation of complex data, and the relationships between the data, by analyzing methods to capture process flows, and the commonality between the way that MBSE tools store and display data. Understanding this commonality is essential to transfer of data between tools. This research explores the common features of subset of the System Modeling Language (SysML), and how they are implemented in three commonly used tools - CATIA MagicDraw, SPEC Innoslate, and Vitech GENESYS.

To address the data exchange problem, this research developed the Architecture Interoperability Method for Systems Engineering (AIM SE). AIM SE is not a new modeling language or tool, but a new methodology for interoperability. It leverages the notion of SysML-Lite which is a simplified version of the language that uses a sub-set of diagram features for six of the nine SysML diagrams (Friedenthal, Moore, and Steiner 2015).

The fundamental principle of AIM SE is the maximization of common diagram features to allow for maximum data exchange within a modeling language and tool ecosystem. AIM SE includes most SysML control constructs of the activity diagram, block definition diagram constraint compartments, block definition diagram operations compartments, block definition diagram ports, directionality of ports, and many other SysML features. Understanding the commonality of how MBSE tools handle SysML data is essential to transfer of data between tools. Common features of subset of the SysML diagrams, and are implemented in three commonly used tools - CATIA MagicDraw, SPEC Innoslate, and Vitech GENESYS. AIM SE is developed and evaluates the executability of diagrams and supports the automatic generation of the SysML sequence diagram in all three tools.

B. MBSE TOOLS USED IN AIM SE

MagicDraw was originally developed to support software development with an emphasis of code generation from class diagrams and modeling of software requirements with use case and sequence diagrams. The tool is based on the Unified Modeling

Language (UML) which is used to support software specification and development. SysML is a UML profile which was extended by Cameo Systems Modeler to support systems development.

Vitech GENESYS derives its lineage from the first MBSE tool, USASDC's Distributed Computing Design System (DCDS), developed by TRW in the 1980s (Smedley 1993). Like DCDS, GENESYS utilizes an ontology, the System Design Language (SDL) to automatically build diagrammatic representations. SPEC Innovations' Innoslate uses LML as its underlying data model. LML includes an ontological data model which was influenced by Vitech's SDL (Lifecycle Modeling Language 2023).

Vitech Core/GENESYS and SPEC Innovations' Innoslate emphasize functional analysis and behavior modeling (Dam 2019). Both Innoslate and GENESYS derive every representation directly from their ontological databases. This allows an autonomous layout and organization of diagrams directly from the database. For example, a sequence diagram can be automatically generated from an activity diagram without the need for a human to redraw this representation. This minimizes manual diagram manipulation and therefore decreases the probability of inconsistencies by negating the development of the same information in different forms. Conversely, Cameo, as with MagicDraw, does not include an ontology editor nor direct ontological representation. This fundamental difference will be explored throughout the comparison between representations. These fundamental differences are important.

C. AIM SE DIAGRAMS

3. Structure Diagrams

The purpose of the SysML structure diagrams is to represent the physical system, how it is structured (including elements, objects, components, packages) and the relationships within the system. Structures diagrams include Package, Block Definition, and Internal Block Definition Diagrams.

a. Package Diagram

A package diagram (PKG) organizes the project into a set of “folders” that allow modelers to plan the modeling effort and quickly discover information from the model and

allow for a straightforward pathway. This construct is supported in CATIA MagicDraw, SPEC Innoslate, and Vitech GENESYS.

Package diagrams are used to logically organize a modeling project. The PKG is represented by a series of folders with the name centered both vertically and horizontally. In Figure 3, the folders coincide with diagram types. However, the package diagram can be organized by the project team's organizational structure, by the hierarchy of the system (e.g. Enterprise, System of Systems, System, etc.), or whatever structure makes sense for the project. Packages may be decomposed with the containment relationship. This is the only relationship supported by all three tools.

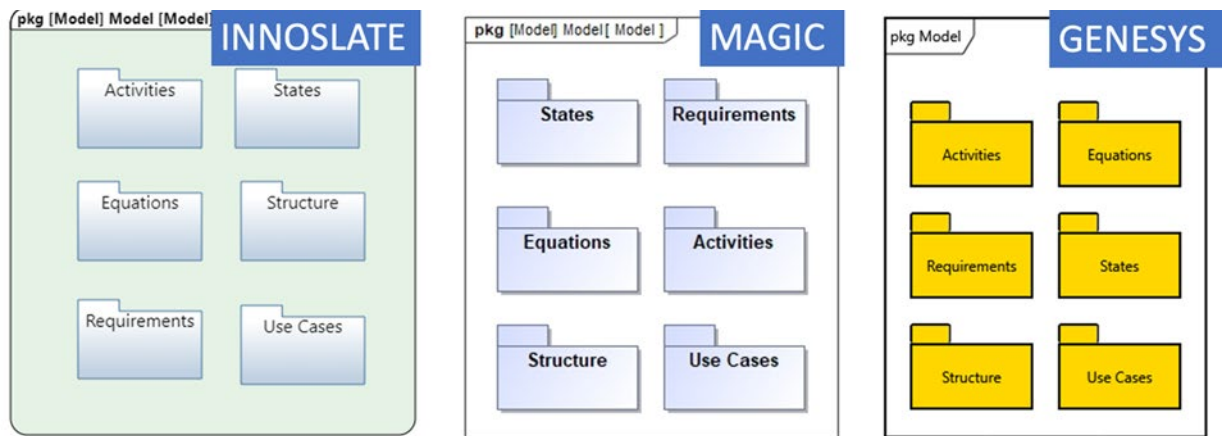


Figure 2. Package Diagrams Across the MBSE Tools.

b. Block Definition Diagram

The Block Definition Diagram (BDD) is one of the most popular diagrams in SysML because it allows the modelers to represent various model viewpoints (i.e. systems, operations, capabilities, etc.) from a hierarchical perspective. In addition to specifying the decomposition of the hierarchy, the BDD expresses attributes such as constraints, values/parameters, and the ports to a block within its visual representation. The BDD within AIM SE is represented as a hierarchy chart with compartments. These compartments can express related constraints, operations, and/or values. Cameo, Innoslate and GENESYS support the BDD in addition to traditional hierarchy charts as shown in Figure 4.

Each tool supports at least one way to represent the decomposition of parts. Innoslate and GENESYS similarly support both hierarchical and BDD representations. Cameo only supports a BDD to represent part decomposition. A comparison is presented in Table 3 below (OMG 2019, No MagicDraw Inc. 2011, Vitech n.d., Innoslate 2022).

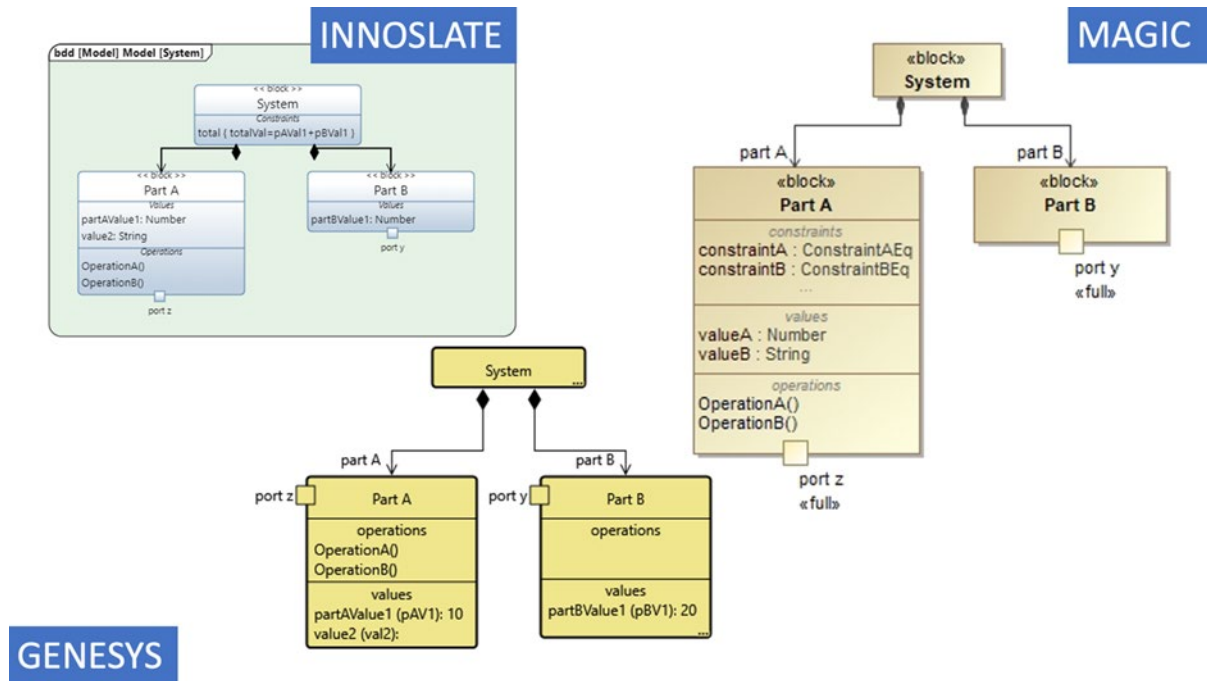


Figure 3. Block Definition Diagrams Across the MBSE Tools.

Table 4. Comparison of Block Definition Diagram Implementations.

Feature	CATIA MagicDraw	GENESYS BDD	Innoslate BDD
Blocks	Yes	Yes	Yes
Decomposition	Yes	Yes	Yes
Constraints	Yes	No	Yes
Values/Parameters	Yes	No	Yes
Multiplicity	Yes	No	Yes
Proxy Port	Yes	No	No
Full Port	Yes	No	Yes

The constraint compartment displays related constraints to the relevant part block. A relationship between this constraint and part is established. While this relationship is consistent across all three tools, GENESYS differs with an explicit constraint block definition diagram to display constraints with their associated part blocks separately.

The values compartment displays all values applicable to the part block. As with operations, these are displayed in alphabetical order by name. Additionally, the units of the value and its default value may also be displayed. All three tools track the name, units, and default value, but all three display this information slightly differently as seen in Figure 4.

AIM SE supports only full ports in both the block definition and internal block diagram representations. These full ports are considered a part of the bill of materials of the system. These ports are displayed, without any connections, in the block definition diagram surrounding the applicable part block.

c. Internal Block Definition Diagram

The Internal Block Diagram (IBD) allows specification of the relationship between parts, their ports, and connections. The IBD can describe the physical conduit which transfers power or information to other parts. For example, the car's computer may connect to various sensors across the power train, cameras, and tire pressure monitoring system. These physical block representations identify which parts are connected between each other. The IBD also allows naming these conduits, for example: Ethernet between the camera and vehicle computer, to further specify vehicle layout. Additionally, the IBD supports the naming of ports, or the means to connect two parts together, for example, an Ethernet jack on the camera and vehicle.

Figure 5 depicts examples of how the IBD is implemented in CSM, GENESYS and Innoslate. Table 4 shows a comparison between how the IBD is implemented in CSM, GENESYS, and Innoslate. The most significant difference between the tools is with the implementation of ports. The three tools support the flow and full ports. Neither GENESYS or Innoslate support nested or proxy ports. The most significant is the difference between supporting full and proxy ports. Proxy ports act as proxies for their

owning blocks or its internal parts, while full ports specify separate elements of the system (OMG 2019; Vitech Corporation nd; Innoslate 2018).

All AIM SE tools support the internal block diagram with several common features. First, the part block includes the name of the type of part plus, optionally, the part's role name as seen in Figure 5. There is always a colon separator between the role name and type.

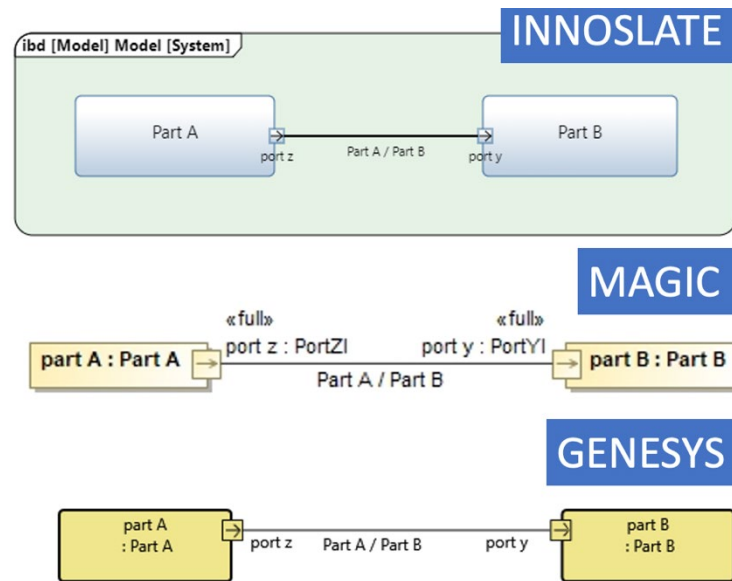


Figure 4. Internal Block Definition Diagrams Across the MBSE Tools.

Table 5. Comparison of Internal Block Definition Diagrams.

Feature	CATIA MagicDraw	GENESYS	Innoslate
Blocks	Yes	Yes	Yes
Decomposition	Yes	Yes	Yes
Item Flow	Yes	Yes	Yes
Nested Ports	Yes	No	No
Flow Ports	Yes	Yes	Yes
Proxy Port	Yes	Yes	No
Full Port	Yes	Yes	Yes

4. Functional Behavior Diagrams

A behavior is an interaction between structural elements, including potential changes of states of the elements. Functional behavior diagrams are each derived from familiar flow charts. Each representation utilizes a box to represent an action and a line to represent flow between actions. This is typically used to specify and analyze the functional behavior of a system. Furthermore, execution of these behavioral representations differs greatly between ontological and UML foundation tools, with ontological tools supporting rich simulation and UML tools focused more on step-through simulators. Capture of non-sequential flow and logical constructs differs significantly between representations and is discussed in this section.

The Activity and State-Machine Diagrams are supported by AIM SE. The Sequence and Use Case Diagrams are not considered due to irreconcilable differences between tools.

a. Activity Diagram

UML introduced activity diagrams (ACT) in UML 1.1 in 1997 (OMG 2017). UML activity diagrams describe an ordered sequential flow through activities. The use of solid lines to signify control flow has remained largely unchanged since the original 1.x UML specification. However, several additions have been included since its introduction, including a migration from a meta-model inherited from State-Machines to a new semantic meta-model specifically for activity modeling (Selic 2005) based upon Petri Nets (Quatrani n.d.). UML 2.0 introduces data flow within control flow, attached data flow, and pin notations to signify input and output pins of activities. Of note, activity diagrams do not include explicit support for probabilistic nor duration modeling; however, this can be accomplished through guards on the control flow (Störrle 2005).

Representative ACTs from CATI MagicDraw, Innoslate, and GENESYS are shown in Figure 6. Each ACT begins with initial nodes and includes activity, action, object nodes (input/outputs), choice, and join nodes. Activity nodes represent a set of actions which perform tasks. These actions and activities are related to each other through control flow and represent the functional sequence of execution. Similarly, object nodes can be related via data flow and/or the control flow itself. Choice nodes represent decisions on the control flow and the selection of a branch to continue execution. The choice path is typically

decided via guards which block entry to a part of the control flow until they are evaluated as true. The end of a flow can be represented with a flow final node which ends the execution of that flow and resumes execution at a higher level of execution. A fork/join represents the start and end respectively of an explicit parallel execution path. An activity diagram ends with an activity final node (Quatrani n.d.). AIM SE requires each activity diagram to begin with an initial node and end with a final node. This reduces ambiguity and guarantees compatibility within the three tools.

Table 5 shows a comparison between activity diagram features generated in CATIA MagicDraw, Innoslate, and GENESYS (OMG 2019; Vitech Corporation n.d.; Innoslate 2018).

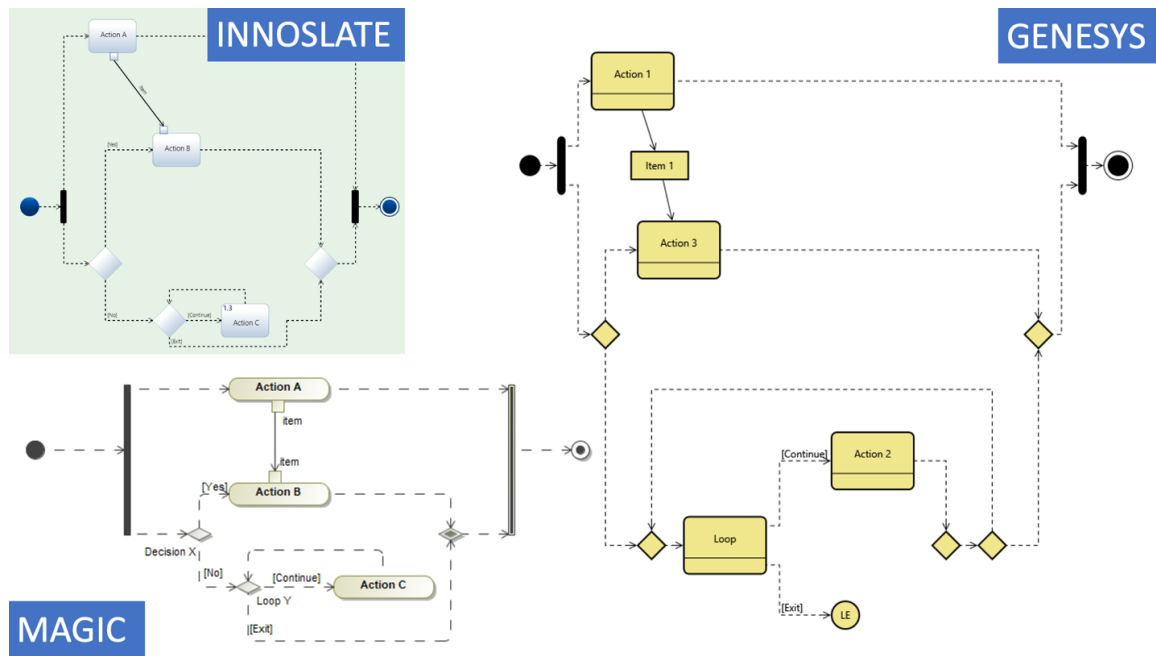


Figure 5. Activity Diagrams Across the MBSE Tools.

Table 6. Comparison of Activity Diagrams.

Feature	CATIA MagicDraw	GENESYS	Innoslate
Action Flow	Yes	Yes	Yes
Control Flow	Yes	Yes	Yes
Decision Node	Yes	Yes	Yes
Merge Node	Yes	Yes	Yes

Start Node	Yes	Yes	Yes
Final Node	Yes	Yes	Yes
Fork Node	Yes	Yes	Yes
Join Node	Yes	Yes	Yes
Event Node	Yes	No	No
Signal Node	Yes	No	No
Time Event	Yes	No	No
Flow Final ¹	Yes	Variant	Variant
Streaming Flows	Yes	No	No
Object Node	Yes	Yes	Yes
Object Flow Ports	Yes	No	No
Activity Partition ²	Yes	Variant	Variant
Interruptible Activity ³	Variant	No	Variant

b. State-Machine Diagram

The UML standard presented state machines to model dynamic behavior of a system. The state machine diagram (STM) expresses transitions between states. This diagram is supported across all three tools; however, only state-to-state transition is supported without any control constructs. Example STMs are depicted Figure 7, and a comparison of how STMs are implementing in modeling-tools us shown in Table 6.

¹ GENESYS uses a kill branch with a slightly different representation to kill flow. Innoslate uses a SYNC action to accomplish this function.

² GENESYS and Innoslate use named fork/join parallels for this function.

³ Innoslate uses a SYNC action to cause interruptions. Interruptible activities are not fully executable in Cameo.

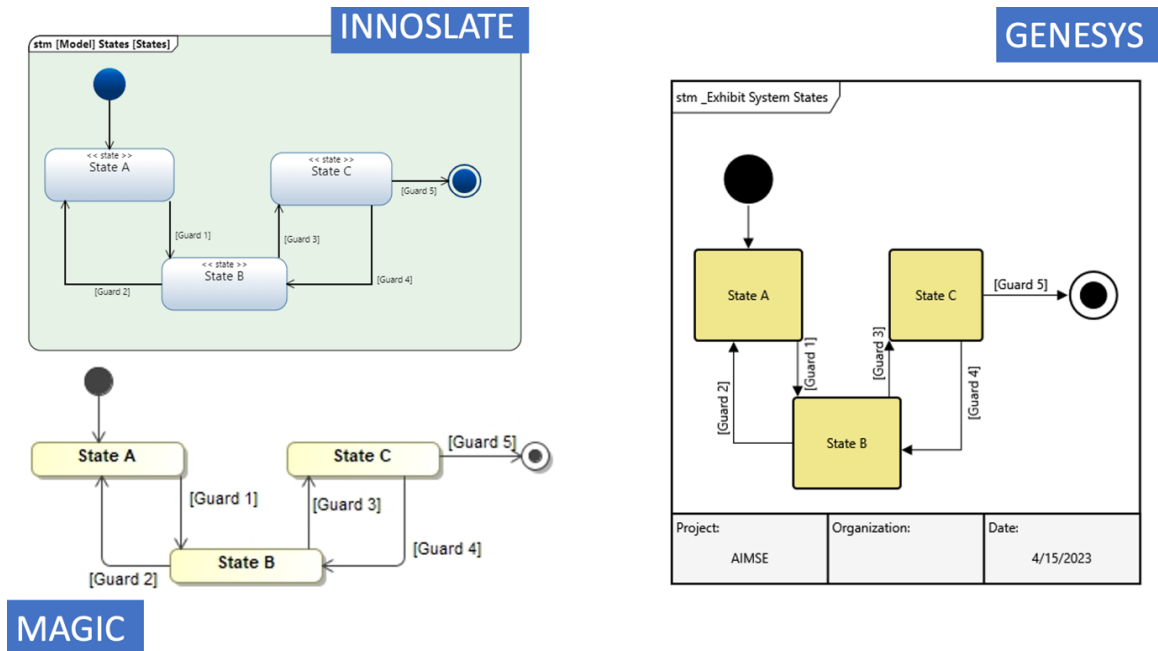


Figure 6. State-Machine Diagrams Across the MBSE Tools.

Table 7. Comparison of State-Machine Diagrams.

Feature	CATIA MagicDraw	GENESYS	Innoslate
States	Yes	Yes	Yes
Transitions	Yes	Yes	Yes
Initial State	Yes	Yes	Yes
Final State	Yes	Yes	Yes
Decomposition	Yes	Yes	Yes
Pseudo States	Yes	No	No
Fork/Join	Yes	No	No
Choice	Yes	No	No

5. Requirements Diagram

Requirements specify legal constraints, which must be realized by a system in order to meet stakeholder needs. SysML includes a requirement diagram (REQ) representation to display, hierarchically, these requirements and the relationship between stakeholder

requirements, system requirements, and model elements. The REQ expresses the relationships between requirements objects and other modeling objects. This diagram is supported across all AIM SE tools studied, and, furthermore, can typically be created directly from a requirements interchange (Raqi) import from a textual based requirements tool. Representative REQ are shown in Figure 8.

The SysML requirement diagram is the common representation across all three tools. Notably, GENESYS and Innoslate each support the same common features across the SysML feature set. GENESYS includes all features except the less frequently used copy relationship. This is seen in Table 7 (OMG 2019; Vitech Corporation n.d.; Innoslate 2018).

Table 8. Comparison of Requirements Diagrams.

Feature	CATIA MagicDraw	GENESYS	Innoslate
Requirement	Yes	Yes	Yes
Copy	Yes	No	Yes
Derive	Yes	Yes	Yes
Satisfy	Yes	Yes	Yes
Verify	Yes	Yes	Yes
Refine	Yes	Yes	Yes
Trace	Yes	Yes	Yes
Decomposition	Yes	Yes	Yes

6. Summary

This chapter introduced a new methodology for maximizing interoperability in SysML with AIM SE. This AIM SE SysML minimization leverages lessons learned from SysML-Lite. Six of the nine SysML diagrams can be represented directly with AIM SE or through auto generation in at least three common systems engineering tools. AIM SE was also cross referenced to the draft SysML 2 specification and was found that it could be a key drive for compatibility. SysML 2 includes a textual notation, KerML, which can be used as an interoperability method between tools. Appendix C shows SysML KerML as compared with its AIM SE equivalent to ensure compatibility with the draft SysML 2 specification.

Future work remains to test AIM SE with each tool vendor's final implementation of SysML 2. It is possible and likely that AIM SE may be expanded in future years pending new tool vendor features across the community. Maximizing interoperability with AIM SE can deliver capabilities to the consumer faster than ever before and ensure a rich supplier ecosystem across the industrial base.

V. USING ONTOLOGIES FOR RELATIONAL DATABASES

A. OVERVIEW

Most organizations utilize information systems to manage diverse data such as products, parts, employees, customers, medical records, financials, stakeholders, and suppliers. Typically, the creation of these systems mirrors the construction of physical systems, involving a phase of requirements analysis followed by detailed diagrams. Often, user needs are relayed to the development team who map out the system design, using a whiteboard or computer. However, adopting ontology in the design and development of these complex information systems might be a more effective method. Ontologies provide a structured framework for knowledge representation. By integrating ontologies to the design phase, information systems can better unify data, ensuring consistent language and rules across various components. This approach not only enhances semantic understanding but also enhances data retrieval of information and interoperability with other systems. In addition, using ontologies to model information systems can give the design team a clearer perspective of the system's structure and relationships. The result is a system that processes information more precisely and efficiently, ultimately enabling improved decision-making.

This chapter explores the conversion of a real-world information system from ontology web language (OWL) to a relational database (RDB) as shown in Figure 8.

B. APPLYING THE ONTOLOGY TO A RELATIONAL DATABASE

1. Relational Database

“A relational database (RDB) is a way of structuring information in tables, rows, and columns. An RDB has the ability to establish links - or relationships - between information by joining tables, which makes it easy to understand and gain insights about the relationship between various data points” (Google n.d.). RDBs are the primary method used worldwide to store and manage information. RDBs support information management across various industries ranging from financial institutions, retail, technology, government, logistics, healthcare and more. Figure 8 shows an Entity Relational Diagram (ERD), also called a schema, which is a diagram that depicts the relationship between entities or tables within a relational database.

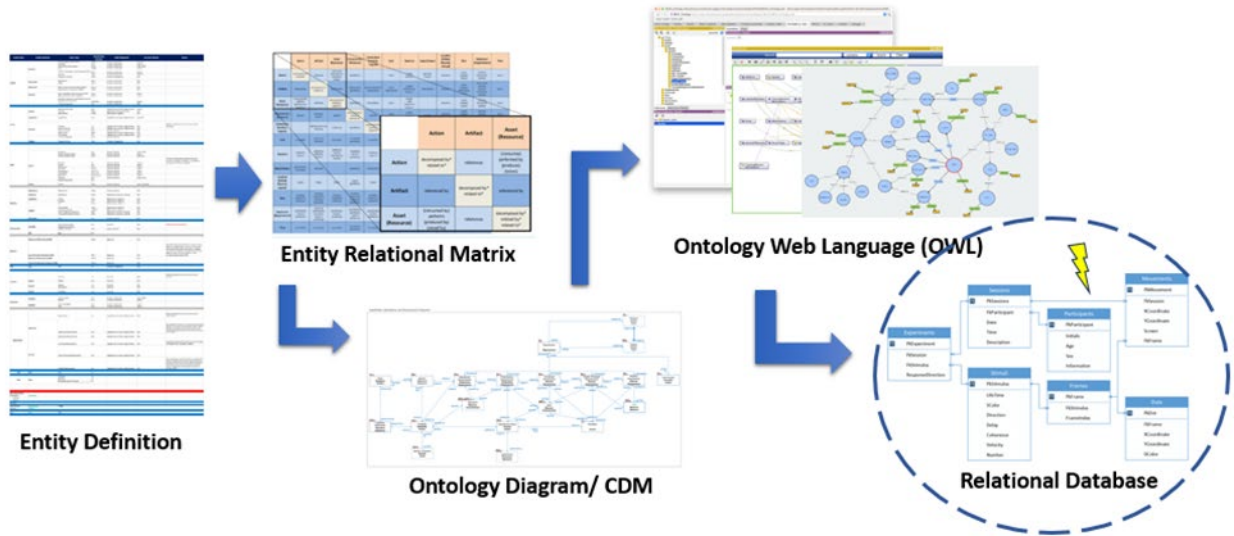


Figure 7. Ontology to Relational Database Overview.

Structured Query Language (SQL) is the standard language for relational databases. SQL was designed for managing and manipulating relational databases. It is used for performing various tasks like creating tables, inserting, updating, managing, deleting, and retrieving data within a relational database management system (RDBMS). SQL serves as the means to interact with databases and is supported by most database systems. SQL commands are specific statements that developers or administrators use to manipulate the data stored in a relational database.

2. Ontology Development

In order to demonstrate that an ontology web language could be mapped to a relational database, a subset of a new Test and Evaluation (T&E) information system being developed was selected. This new information system is intended to track all T&E artifacts used and created in support of naval acquisition T&E programs, such as test plans, scenarios, test objectives, systems issues, analysis results, and reports. Due to the sensitive nature of the real information system, notional data derived for the actual requirement is used, and is shown in Table 8.

Table 9. Notional Requirements for the T&E Information System.

#	Categories	Stakeholder Needs
1	T&E Activities	System must track all T&E artifacts by test event from planning, execution, analysis, and reporting.
2	User Access	Allow T&E personnel (Tester, Analyst, Data Manager, and Program Manager) to access, upload, download, and view test artifacts. Must be accessible anywhere there is a SIPR connection.
3	System data	Must store system data collected from T&E events in both its native and human-readable format.
4	Data Storage	Scalable to support current and future T&E events
5	Measures	Must allow the analysis community to manage measures of effectiveness (MOE) and Measures of Performance (MOP)
6	System Issues	Allow to track Test Observation Reports/ System Issues per event
7	Maintenance	Allow for an admin role to provide routine maintenance, updates, and patches.
8	Security	Implement user roles and permissions to control who can view or modify the artifacts.
9	Backup	Automated backup system to prevent data loss

The development of the ontology web language in Protégé started by defining the concepts or classes involved in the T&E process and providing a taxonomy. The ontology focused on structuring the entire system by test events and users. Test events are the actual execution milestones that all other inputs centered on such as planning, execution, analysis and reporting. Additionally, the second major parent structure was centered on the users or T&E stakeholders, including the tester, data manager, analyst, and program manager as can be seen in Figure 9.

Ontology classes and taxonomy was developed, the next step dealt with creating the relationships within classes or instances. Due to the complexity of ontologies and the scope of this portion of the research, the relationship mapping was primarily focused on the role of the analyst. Therefore, only the actions of the analysts and its relationship with the system was mapped using OWL object properties and data properties. Below are the mapped object properties in the ontology also depicted in Figure 10.

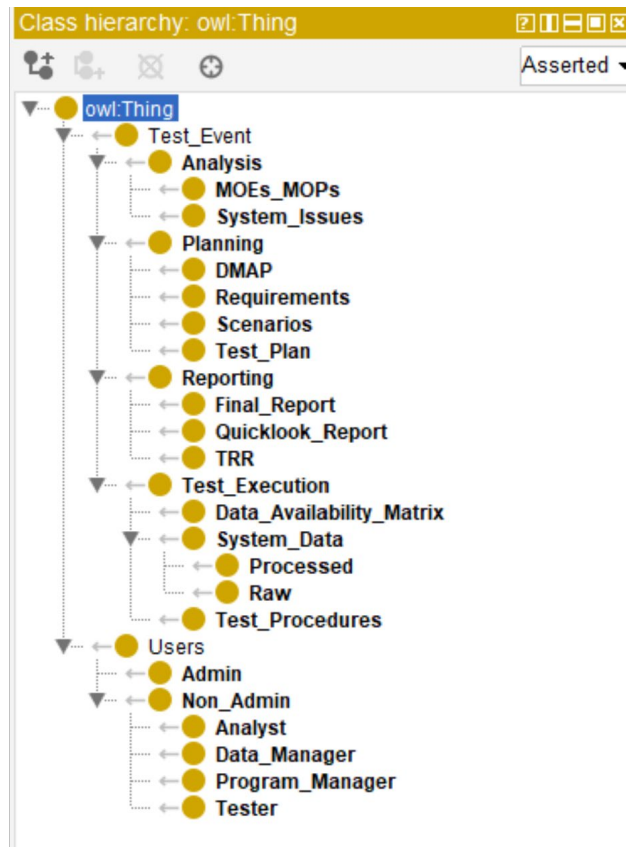


Figure 8. OWL Classes and Taxonomy of T&E Information System.

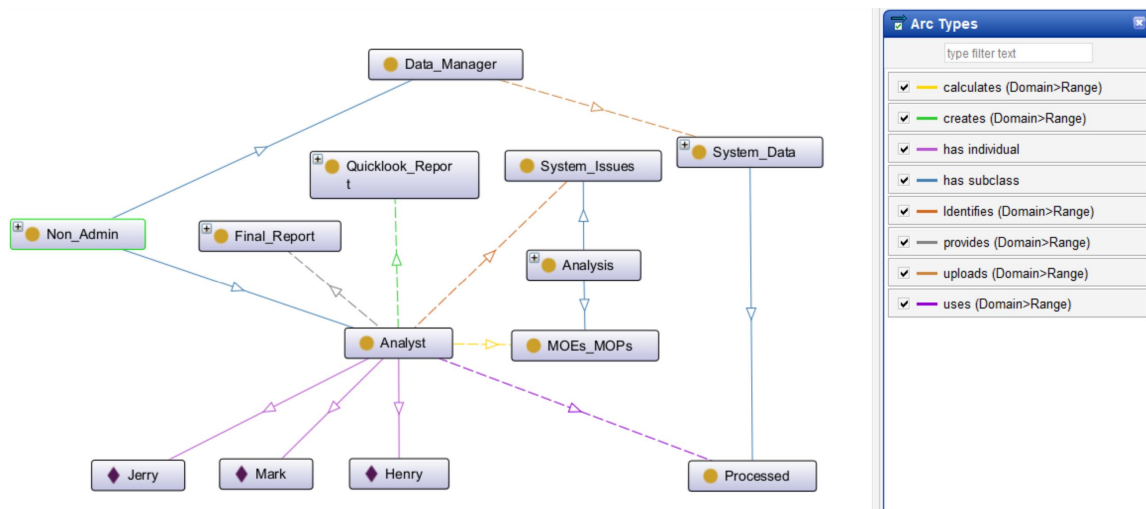


Figure 9. OWL Tree Diagram of Analyst Relationships.

The next step mapped the data properties for the ontology classes. A subset of data properties was developed focused on identifying the datatypes of classes or instances with an identification, name, type, unique ID, and dates as can be seen in Figure 9.

The final step developed the instances of classes – representing actual objects from the ontology model.

7. 3. Mapping OWL to Relational Database

After the ontology for the T&E information system was developed, the next step of this project proceeded with mapping the OWL semantic language to a relational database. The mapping is an essential part of this research project to demonstrate the possibility of developing information systems, based on relational database construct, by using a formal ontology model. To demonstrate this concept, two research papers were identified to have worked on the mapping of ontology to relational database (Ernestas and Nemursite, 2006; Mahmudi, Liem, and Akbar, 2018).

Ernestas and Nemuraite (2006) proposed that a RDB could be transformed to a relational database using a DDL script. Figure 11 depicts the ontology to RDB development and transformation process (Ernestas and Nemuraite, 2006). Additionally, the authors developed an algorithm for this transformation taking into consideration OWL classes, object and data properties, constraints, and instances. The mapping used by Ernestas and Nemuraite (2006) focused on:

- Classes mapped to tables
- Object and data properties mapped to relations [relationship tables]
- Attributes and constraints mapped to metadata.

Mahmudi, Liem, and Akbar (2018) identified a more detailed mapping from OWL to RDB with the goal of creating information systems for web applications based on ontology models with minimal effort from a software development team. Their proposed mapping reveals the following:

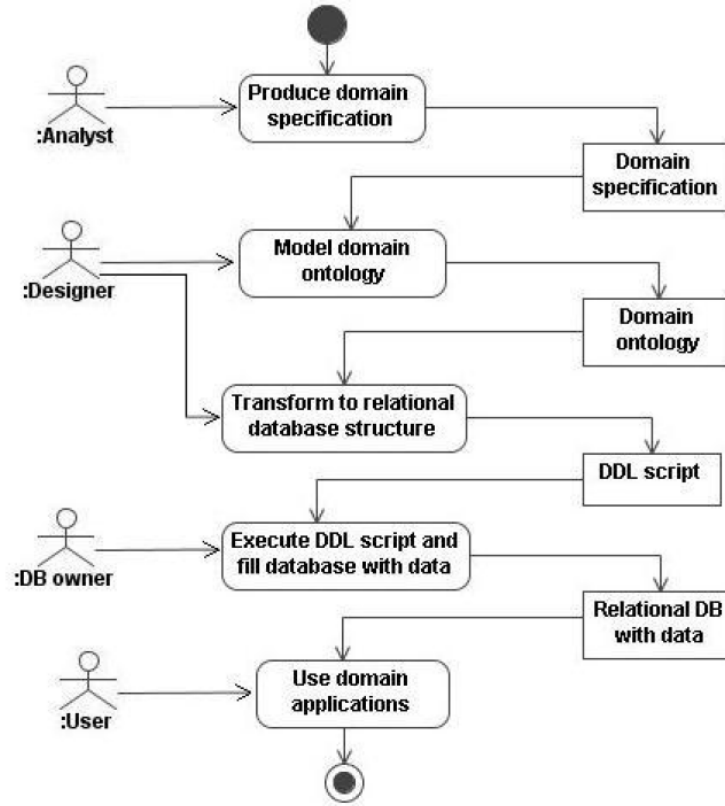


Figure 10. Ontology to RDM Transformation Process (Ernestas and Nemuraite, 2006).

- Classes and subclasses are similar to tables
- Object property similar to relationship tables
- Functional object property is similar to a field
- The value of a functional object property is similar to a row.

Furthermore, Mahmaudi, Liem, and Akbar (2018) developed a mathematical representation to account for the transformation of classes in the ontology to tables in a relational database as shown in Equation (1). The equation shows that an ontology web language will generate more tables than classes in relational database construct due to the addition of relationship tables (Mahmaudi, Liem, and Akbar, 2018).

$$OKL = BTB - BTR \quad (1)$$

where:

OKL : the number of classes in the ontology

BTB : the number of tables in the relational database

BTR : the number of relationship tables in the relational database

Based on the insights obtained from Ernestas and Nemursite (2006), and Mahmudi, Liem, and Akbar (2018) an updated ontology mapping was derived as shown in Table 9. This mapping further incorporates the class hierarchy or taxonomy by using primary and foreign keys concepts. Additionally, the use of DDL and DML SQL command types are incorporated to create the different types of tables, columns, rows, fields, keys, and data. This thoroughly maps the T&E information system ontology model to its relational database counterpart. The outcome of this process highlights the intricate connections amongst various components, emphasizing the alignment between different elements within the T&E information system ontology model and its relational database construct.

Table 10. OWL to Relational Database Mapping T&E Information System.

#	Ontology Web Language (OWL)	Relational Database (RDB)	Database Construct	SQL Method
1	Classes	Tables	26 Tables	DDL
2	Object Properties	Relationship Tables (many-to-many)	6 Tables	DDL
3	Data Properties	Fields (Columns)	16 Fields	DDL
4	Classes Hierarchy (linkage)	Primary & Foreign Key	24 Links	DDL
5	Individual/Instances	Record (Rows)	10 Rows	DML

After completion of the mapping, SQL commands were formulated to generate the DDL and DML files. The SQL command code are available in Appendices C and D. One element that couldn't be encapsulated in the mapping was the use of OWL annotations, also known as metadata descriptions. There was no straightforward, one-to-one mapping of OWL annotations to SQL available. A potential workaround for this restriction could be employing an annotation table and tracking all annotations with corresponding table numbers using a DML; however, this approach would be complicated and compromise the previously noted mathematical equation between classes and tables.

Lastly, to ensure the precise execution of the mapping between the OWL ontology and the RDB, a third-party tool, Oracle SQL Developer Data Modeler, was used during the validation phase. The validation confirmed accurate mappings for classes, object

properties, data properties, and taxonomy. However, the validation of instances could not be verified using the Oracle tool, as it does not process DML files.

VI. SUMMARY, CONCLUSIONS, AND ROCOMMENDATIONS

A. OVERVIEW

The FY23 NRP research, conducted in conjunction with MCSC, analyzed and refined the land domain ontology, and conceptual data model (CDM), developed in the FY22 NRP research entitled, “Developing a Model-Based Systems Engineering Land Domain Construct for Marine Corps Systems Command,” with three goals. First, analyze how the land domain ontology and CDM may be implemented to components that interface with land domain. This goal validated the FY22 NRP ontology, and CDM, as mathematically correct by modeling it in the Protégé-OWL ontology tool. This validation illustrated the ontology’s utility to serve as the structure for an authoritative source of truth. This goal supports the development of the MCSC Entity Re-use Library (the first known authoritative source of truth in DoD) by using the previously developed ontology as the libraries structure.

Second, explore options to exchange data through the representation of complex data and the relationships between the data. This goal explored how various diagram types are implemented in various modeling languages and MBSE tools and demonstrates how data can be shared between each. A key feature of the Entity Re-Use Library is data sharing among disparate tools. This research explored a new approach to data sharing by defining a set of features common to the three tools that were considered.

And third, evaluate and update the existing ontology and CDM with corresponding land domain and the Naval Operational Architecture from a mission scenario perspective. This goal explored how the ontology and CDM can be applied to the structuring of a relational database being developed by one of the Naval Surface Warfare Centers. This goal is illustrated that the ontology can be applied to more than MBSE Tool derived data, but is applicable to other data sources.

FINDINGS AND CONCLUSIONS

This research has taken many twists and turns with seemingly disparate approaches to each of the goals. However, the overarching goals were derived from the DoD Digital Engineering Strategy. Specifically, Goal 2 – “Provide an enduring, authoritative source of

truth”, and Goal 4 - “Establish a supporting infrastructure and environments to perform activities, collaborate, and communicate across stakeholders” (DASD SE 2018).

Goal 2 of the DoD Digital Engineering Strategy (DASD SE, 2018) calls for an “authoritative source of truth,” to serve as a meta-model to show how disparate types of data, in various format, derived from different modeling language, model-based tools, and developed by different organizations come together to represent the system holistically. The first goal of the FY23 research was to analyze how the land domain ontology may be implemented to components that interface with land domain. The starting point for this goal was using the ontology, and CDM, developed during the FY22 NRP. The ontology previously developed was in MSEXcel and MSPowerPoint respectively. This focus area validated the FY22 NRP ontology by modeling it in the Protégé-OWL ontology tool.

Protégé-OWL supports the creation, editing, visualization, and management of ontologies. It allows for the establishment of taxonomy hierarchy as representation of the ontology, as well as the representation of entity relationships, and for the visualization of the ontology. Most importantly, Protégé-OWL is based on the mathematical properties of set theory, therefore proved that the ontology was mathematically consistent. This validation illustrates the ontology’s utility to serve as the structure for an authoritative source of truth.

The purpose of Goal 4 of the DoD Digital Engineering Strategy is the notion that data will be shared across the digital ecosystem, regardless of the modeling language or tool that was used to generate the data. The predominate thinking within DoD is to standardize to a single tool. Each modeling tool has its strengths and weaknesses; therefore, multiple tools are needed.

This research explored options to exchange data through the representation of complex data and the relationships between the data. Two methods to exchange data between tools exist. First, data can be translated using a XML Metadata Exchange (xmi). However, inconsistencies between tools exist even when using the same modeling language. We explored the System Modeling Language (SysML 2.0) Application Programming Interface (API), and found that this API will solve many of the data interoperability issues but is not the “holy grail as advertised. Due to tool vendor implementations, the API will likely not perform well across tools.

The second method is to define a set of common features among the tools. The Architecture Interoperability Method for Systems Engineering (AIM SE) was developed, not as a new tool or modeling language, but as a means to define the maximum set of features. AIM SE was tested using three model-based tools, and proved to be successful using the SysML diagrams.

Lastly, the ontology was evaluated using a mission scenario applied to defining the structure of a relational database (RDB). This mapping incorporated the class taxonomy to define the structure of the different types of tables, columns, rows, fields, keys, and data. This demonstrated the versatility of the ontology to be applied to data that is outside the realm of data generated by MBSE tools.

B. RECOMMENDATIONS FOR FURTHER RESEARCH

This research has significantly advanced digital engineering by demonstrating an ontology that can be used as the basis to structure an authoritative source of truth and that contains data from disparate modeling languages, tools, and database type. Furthermore, the notion of data sharing between disparate modeling-languages and tools was demonstrated using AIM SE. Despite these advances, more work is needed.

This research has demonstrated that the ontology is mathematically consistent and can theoretically be used as a structure for authoritative source of truth. The ontology was modeled in Protégé-OWL, which claims to be modeling language and tool agnostic. However, validating this claim was not within the scope of this research. Further research should validate that Protégé OWL can be used with disparately modeling languages and tools and will function as the implemented structure of the authoritative source of truth.

AIM SE defined a maximum set of modeling-language features that can be used to exchange data between different modeling tools. AIM SE is a promising approach but has only been demonstrated with a single modeling language and three tools. However, digital engineering data is developed from many different modeling languages and tools. As such, the scalability of AIM SE needs to be explored with a broader set of modeling languages and modeling tools.

This research has demonstrated that the ontology, as modeled in Protégé-OWL can be used to define the structure of a RDB. While many RDBs contain system data, it is unclear

the role of RDBs within the authoritative source of truth and should be explored. Using the ontology to define a structured database is a promising development, however, the true benefits have not defined, and should be explored.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: ONTOLOGY

Table 1. Entity Classes, Subclasses. and Data Types.

Entity Class	Entity Subclass	Data Type
<i>Artifact</i>	Guidance	Strategic Objectives & Policies
		Directives
		Defense Planning Guidance
		Plans
		Tactics, Techniques, and Procedures (TTP)
		Doctrine
		Program of Record
	Governance	Regulations
		Policy
	Agreement	Memorandum of Understanding (MOU)
		Memorandum of Agreement (MOA)
	Standard	Joint Capabilities Areas Document (JCA)
		Mission Essential Tsk List (METL)
		Required Operational Capabilities/Projected Operational Environment
		Technical Standard
<i>Action</i>	Activity	Operational Activity
		Training
		Maintenance
	Capabilities	Capabilities
	Function	Function
		System Function
		Service Function
		SOA Function

		Test Process
		Test and Diagnostic
	Program	Program Activity
<i>Asset</i>		
	System	Enterprise
		System of Systems (SoS)
		Family of Systems (FoS)
		Platform
		System
		Sub-System
		Assembly
		Sub-Assembly
		Component
		Hardware
		Software
	Service	Service
<i>Resource</i>		
	Organization	Organization
	Performer	User/Person
	Installation	Facility
		Property
		Site
	Logistics	Consumables
		Support Equipment
		Test & Diagnostic Equipment
		Training Devices & Simulators
	Information	Data
<i>Characteristic</i>		Environmental State
	Condition	Operational Condition
		Quality of Service

	Skill	Skill
Measure	Measures of Effectiveness (MOE)	
	Key Performance Parameters (KPP)	
	Measures of Performance (MOP)	
	Technical Performance Measures (TPM)	
Connection	Conduit	Physical Connection
	Logical	Virtual Connection
Cost	Research, Development, Test and Evaluation (RDT&E)	
	Procurement	
	Operation and Maintenance (O&M)	
	Operation and Maintenance (O&M)	
	Military Construction (MILCON)	
Decision	Strategic Decision	
	Routine Decision	
	Programmed Decision	
	Policy Decision	
	Operational Decision	
	Organizational Decision	
Input/Output	Input	
	Output	
Location	Orbital	Orbital

	Physical	Address, Geolocation
	Virtual	IP Address
Statement	Guidance	Mission
	Standard	Joint Capabilities
		Task
Requirement	Originating	Performance Requirement
		Operational Requirements
	Derived	Functional Requirements
		System (Physical) Requirements
		Interface Requirements
Risk	Risks	
Time	Time	Event
		Milestone
		Phasing/Evolution/Forecast

Table 2. Entity-Relationship Table.

	Action	Artifact	Asset (Resource)	Characteristic (Measure)	Connection	Cost	Decision	Input/Output	Location	Risk	Statement (Requirement)	Time
Action	decomposed by* related to* traced to traced from*	referenced by*	(consumed by*) performed by (produces) (seizes)	specified by*	-	incurred by*	enabled by* results in	generated by* received by*	located at*	caused by* mitigated by* resolved by*	(satisfies by*) traced from* (verified by*)	occurs
Artifact	referenced by*	decomposed by* related to*	referenced by	referenced by* specified by*	defines protocol referenced by*	incurred by* referenced by	enabled by* referenced by* results in	referenced by*	located at*	caused by* mitigated by* referenced by* resolved by*	referenced by* (satisfies by*) source of traced from* (verified by*)	occurs
Asset (Resource)	(consumed by*) Performs* (produced by*) (seized by*)	referenced by*	decomposed by* orbits by* related to*	specified by*	connected by*	incurred by*	enabled by* responds to results in	-	located at*	caused by* mitigated by* resolved by*	(satisfies by*) traced from* (verified by*)	occurs
Characteristic (Measure)	specified by*	referenced by* specified by*	specified by*	decomposes decomposed by* related to* specified by*	specified by**	incurred by* specified by*	enabled by* results in specified by*		located at* specified by*	caused by* mitigated by* resolved by* specified by*	(satisfies by*) specified by* traced from* (verified by*)	occurs specified by*
Connection	-	defines protocol referenced by*	connects to*	specified by*	decomposed by* joined by* related to*	incurred by*	enabled by* results in	transfers	located at*	caused by* mitigated by* resolved by*	(satisfies by*) traced from* (verified by*)	occurs
Cost	incurred by*	incurred by* referenced by*	incurred by*	incurred by* specified by*	incurred by*	decomposed by* related to*	enabled by* incurred by* results in	incurred by*	located at*	caused by* incurred by* mitigated by* resolved by*	incurred by* (satisfies by*) traced from* (verified by*)	occurs
Decision	enabled by* result of*	enabled by* referenced by* result of*	enabled by* made by* responded by result of*	enabled by* result of specified by*	enabled by* result of*	enabled by* incurred by* result of*	decomposed by* related to*	enabled by* result of*	located at*	caused by* enabled by* mitigated by* result of* resolved by*	alternative enabled by* traced from* result of*	date resolved decision due occurs
Input/Output	generated by* received by*	referenced by*	-	specified by*	transferred by	incurred by*	enabled by* result of*	decomposed by* related to*	located at*	caused by** mitigated by* resolved by*	(satisfies by*) traced from* (verified by*)	occurs
Location	located at*	located at*	located at*	located at* specified by*	located at*	located at*	located at*	located at*	decomposed by* related to*	located at* mitigated by*	located at* (satisfies by*) traced from* (verified by*)	occurs
Risk	caused by* mitigated by resolved by	caused by* mitigated by referenced by* resolved by	caused by* mitigated by resolved by	caused by* mitigated by resolved by specified by*	caused by* mitigated by resolved by	caused by* incurred by* mitigated by resolved by	caused by* enabled by* mitigated by results in resolved by	caused by* mitigated by resolved by	located at* mitigated by	caused by** decomposed by* related to* resolved by*	caused by* mitigated by resolved by	occurs mitigated by
Statement (Requirement)	(satisfies by*) traced to* (verified by*)	referenced by* (satisfies by*) sourced by* traced to* (verified by*)	(satisfies by*) traced to* (verified by*)	(satisfies by*) specified by* traced to* (verified by*)	(satisfies by*) traced to* (verified by*)	incurred by* (satisfies by*) traced to* (verified by*)	alternative of enabled by* traced to* results in*	(satisfies by*) traced to* (verified by*)	located at* (satisfies by*) traced to* (verified by*)	caused by* mitigated by* resolved by*	decomposed by* traced to* related to* referenced by*	occurs (satisfies by*) (verified by*)
Time	occurred by*	occurred by*	occurred by*	occurred by* specified by*	occurred by*	occurred by*	date resolved by* decided by occurred by*	occurred by*	occurred by*	occurred by* mitigated by*	occurred by* (satisfies by*) (verified by*)	decomposed by* related to*

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: ONTOLOGY WEB LANGUAGE (OWL) FUNDAMENTALS

A. OVERVIEW

The Ontology Web Language (OWL) is a knowledge representation language, designed to formulate, exchange and reason with knowledge about a domain of interest. OWL is not a programming language, but it is specifying concepts, relationships, as well as characteristics of concepts and relationships in a human and machine understandable model. OWL is declarative in that it describes a state of affairs in a logical way, but goes beyond simple syntactic standards, and is based on sound mathematical (set-theory) logic. OWL is expressive and unambiguous (Adapted from W3C, 2012).

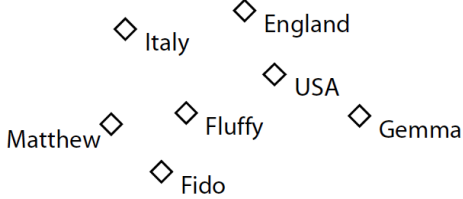
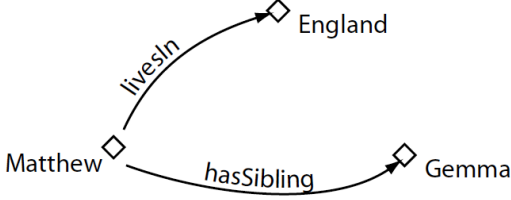
The basic OWL concepts are:

- **Axioms** - The basic statements that an OWL ontology expresses.
- **Entities** - Elements used to refer to real-world objects.
- **Expressions** - Combinations of entities to form complex descriptions from basic ones.

B. OWL FUNDAMENTAL PRINCIPLES AND PROPERTIES

OWL is based on the mathematical principles and properties of set theory. The fundamental principles are shown in **Table-1**, and the OWL properties are shown in **Table-2** (Bedford and Cooke, 2001)

Table 1. OWL Fundamental Concepts

OWL Fundamental Principles	Illustrative Example
Individuals - Represent objects in the domain in which we are interested. Individuals are also known as instances, and can be referred to as being 'instances of classes'.	
Properties - Binary relations on individuals - i.e. properties link two individuals together.	

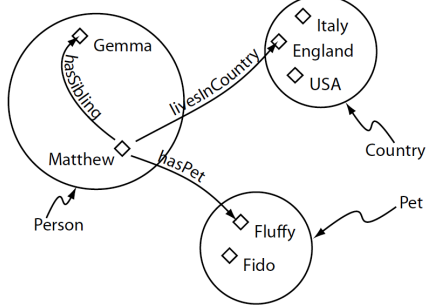
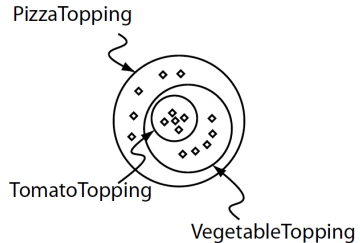
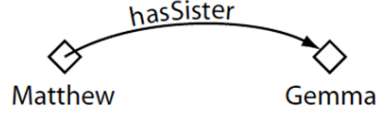
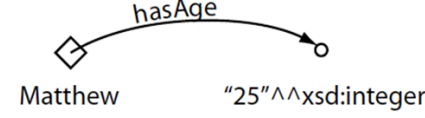
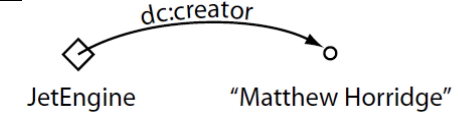
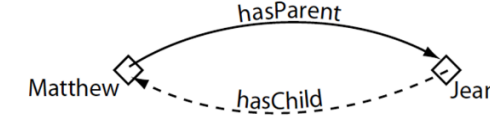
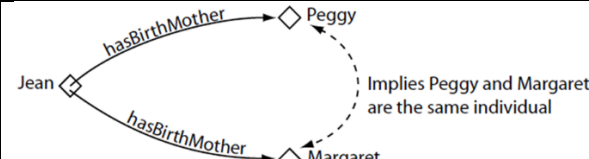
Classes – An interpretation of a set that contains individuals. A collection of individuals or individuals objects The word concept is sometimes used in place of class. Classes are a concrete representation of concepts.	
Taxonomy- An organization of classes into a superclass-subclass hierarchy.	

Table 2. OWL Properties.

OWL Property	Illustrative Example
Object Property - Used to establish relationships between individuals (or instances).	
Datatype Property - Relates individuals to literal data (e.g., strings, numbers, date times, etc.).	
Annotation Property - Used to add information (metadata data about data) to classes, individuals and object/datatype properties	
Inverse Property - If some property links individual <i>a</i> to individual <i>b</i>, then its inverse property will link individual <i>b</i> to individual <i>a</i>.	
Functional Property - A property that can have only one (unique) value <i>y</i> for each instance <i>x</i>	

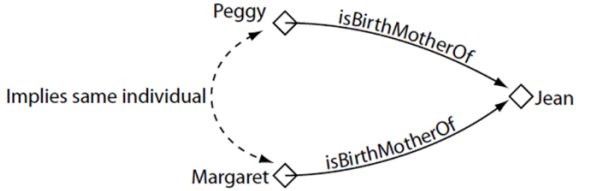
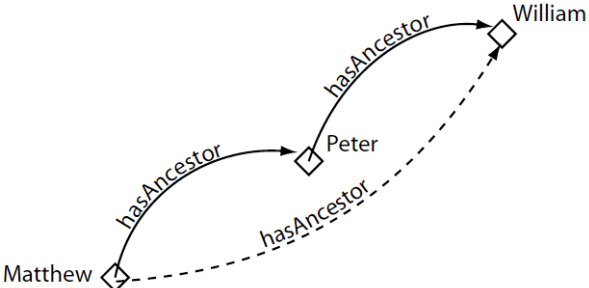
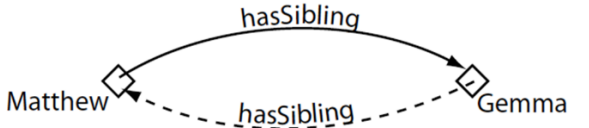
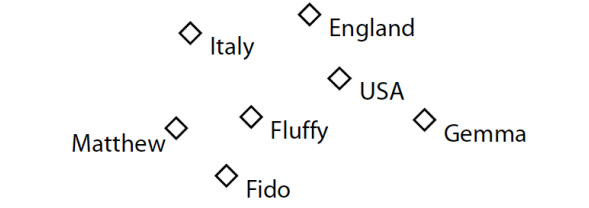
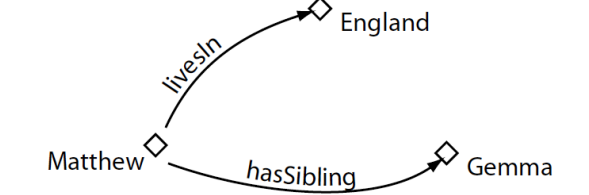
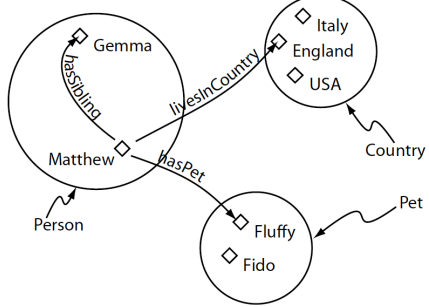
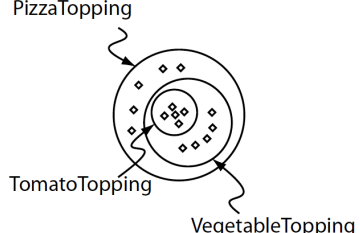
Inverse Functional Property - Asserts that the inverse of the property is functions. This implies there can be at most one incoming relationship along the property for that individual.	
Transitive Property – If the property relates individual <i>a</i> to individual <i>b</i>, and also individual <i>b</i> to individual <i>c</i>, then we can infer that individual <i>a</i> is related to individual <i>c</i> via property <i>P</i>. If a property is transitive, then its inverse property should also be transitive. (NOTE: If a property is transitive then it cannot be functional.)	
Symmetric Property – If property <i>P</i> is symmetric, the property relates individual <i>a</i> to individual <i>b</i> then it relates individual <i>b</i> is individual <i>a</i> via property <i>P</i> also.	

Table 3. OWL Fundamental Concepts

OWL Fundamental Principles	Illustrative Example
Individuals - Represent objects in the domain in which we are interested. Individuals are also known as instances, and can be referred to as being 'instances of classes'.	
Properties - Binary relations on individuals - i.e. properties link two individuals together.	

Classes – An interpretation of a set that contains individuals. A collection of individuals or individuals' objects The word concept is sometimes used in place of class. Classes are a concrete representation of concepts.	
Taxonomy- An organization of classes into a superclass-subclass hierarchy.	

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C: SYSML KERML TO AIM SE COMPARISION

The Architecture Interoperability Method for Systems Engineering (AIM SE) method can be a key driver for SysML 2 compatibility. SysML 2 includes a textual notation, KerML, which can be used as an interoperability method between tools. In the following **Tables 1 through 7**, SysML KerML is compared with its AIM SE equivalent to ensure compatibility with the draft SysML 2 specification.

Table 1. Package Diagram Features and SysML KerML Notation.

Feature	SysML 2 Textual Code
Package Symbol	package packageName { ... }
Containment Relationship	package packageName { package subPackageName { } }

Table 2. Block Definition Diagram Features and SysML KerML Notation.

Feature	SysML 2 Textual Code
Part	part partName { ... }
Constraints Compartment	part partName { assert constraint {a <= b } }
Operations Compartment	part partName { action actionName; }
Values Compartment	part partName { attribute attributeName; }
Containment Relationship	part parentPartName { part childPartName { ... } }
Ports	part partName { port portName; }

Table 3. Internal Block Definition Diagram Features and SysML KerML Notation.

Feature	SysML 2 Textual Code
Ports	partName { port a; port b; ... }
Port Directionality	partName { port in a; port out b; ... }
Port Connection	interface : InterfaceName connect portA ::> ibdPortA to portB ::> ibdPortB;

Table 4. State-Machine Diagram Features and SysML KerML Notation.

Feature	SysML 2 Textual Code
State	state stateName { ... }
State Transition	state firstStateName; accept when true then secondStateName
Initial	entry
Final	exit

Table 5. Activity Diagram Features and SysML KerML Notation.

Feature	SysML 2 Textual Code
Action	action actionName
Decision	decide
Initial	first start
Final	done
Fork	fork
Join	join joinName
Object Flow	flow firstPortName to secondPortName
Merge	merge mergeName
Succession	then actionName

Table 6. Requirements Diagram Features and SysML KerML Notation.

Feature	SysML 2 Textual Code
Requirement	requirement <'Id '> requirementName { doc /* requirement text here */ }
Containment Relationship	requirement <'Id '> subrequirementname : parentrequirementname { doc /* requirement text here */ }
Satisfaction Relationship	action actionName { satisfy requirement requirementName; }
Verification Relationship	action actionName { verify requirement requirementName; }

Table 7. Parametric Diagram Features and SysML KerML Notation.

Feature	SysML 2 Textual Code
Constraint	constraint def ConstraintDefName { ... }
Constraint Property	constraint constraintName : ConstraintDefName { in propName : PropType }
Connections	assert constraint constraintName : equationName { in param = otherParam }

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D: DDL FILE

```
/* ***** DDL File ***** */
/* ***** Test_Event ***** */
CREATE TABLE Test_Event
(
    test_eventId nvarchar(100) NOT NULL PRIMARY KEY CLUSTERED,
    uniqueEventID nvarchar(100) NOT NULL UNIQUE,
    eventName nvarchar(100) NOT NULL
);
/* ***** Users ***** */
CREATE TABLE Users
(
    usersId nvarchar(100) NOT NULL PRIMARY KEY CLUSTERED,
    userType nvarchar(100) NOT NULL
);
/* ***** Test_Execution ***** */
CREATE TABLE Test_Execution
(
    test_executionId nvarchar(100) NOT NULL PRIMARY KEY CLUSTERED,
    eventStartDate nvarchar(100) NOT NULL,
    eventEndDate nvarchar(100) NOT NULL,
    CONSTRAINT FK_Test_Execution FOREIGN KEY (Test_ExecutionId)
        REFERENCES Test_Event(test_eventId)
);
/* ***** Planning ***** */
CREATE TABLE Planning
(
    planningId nvarchar(100) NOT NULL PRIMARY KEY CLUSTERED,
    CONSTRAINT FK_Test_Planning FOREIGN KEY (PlanningId)
        REFERENCES Test_Event(test_eventId)
);
/* ***** Analysis ***** */
CREATE TABLE Analysis
(
    analysisId nvarchar(100) NOT NULL PRIMARY KEY CLUSTERED,
    CONSTRAINT FK_Test_Analysis FOREIGN KEY (AnalysisId)
        REFERENCES Test_Event(test_eventId)
);
/* ***** Reporting ***** */
CREATE TABLE Reporting
(
    reportingId nvarchar(100) NOT NULL PRIMARY KEY CLUSTERED,
    CONSTRAINT FK_Test_Reporting FOREIGN KEY (ReportingId)
        REFERENCES Test_Event(test_eventId)
);
/* ***** Admin ***** */
CREATE TABLE Admin
(
    adminId nvarchar(100) NOT NULL PRIMARY KEY CLUSTERED,
    adminName nvarchar(100) NOT NULL,
    CONSTRAINT FK_Users_Admin FOREIGN KEY
        (AdminId) REFERENCES Users(usersId)
);
```

```

/* ***** Non_Admin ***** */
CREATE TABLE Non_Admin
(
    non_adminId nvarchar(100) NOT NULL
        CONSTRAINT PK_non_adminId PRIMARY KEY CLUSTERED,
    CONSTRAINT FK_Users_Non_Admin FOREIGN KEY
        (Non_AdminId) REFERENCES Users(usersId)
);
/* ***** Data_Availability_Matrix ***** */
CREATE TABLE Data_Availability_Matrix
(
    data_availability_matrixId nvarchar(100) NOT NULL
        CONSTRAINT PK_data_availability_matrixId PRIMARY KEY CLUSTERED,
    CONSTRAINT FK_Test_Execution_Data_Availability_Matrix FOREIGN KEY
        (Data_Availability_MatrixId) REFERENCES Test_Execution(test_executionId)
);
/* ***** Test_Procedures ***** */
CREATE TABLE Test_Procedures
(
    test_proceduresId nvarchar(100) NOT NULL
        CONSTRAINT PK_test_proceduresId PRIMARY KEY CLUSTERED,
    CONSTRAINT FK_Test_Execution_Test_Procedures FOREIGN KEY
        (Test_ProceduresId) REFERENCES Test_Execution(test_executionId)
);
/* ***** System_Data ***** */
CREATE TABLE System_Data
(
    system_dataId nvarchar(100) NOT NULL
        CONSTRAINT PK_system_dataId PRIMARY KEY CLUSTERED,
    CONSTRAINT FK_Test_Execution_System_Data FOREIGN KEY
        (System_DataId) REFERENCES Test_Execution(test_executionId)
);
/* ***** Scenarios ***** */
CREATE TABLE Scenarios
(
    scenariosId nvarchar(100) NOT NULL
        CONSTRAINT PK_scenariosId PRIMARY KEY CLUSTERED,
    CONSTRAINT FK_Planning_Scenarios FOREIGN KEY
        (ScenariosId) REFERENCES Planning(planningId)
);
/* ***** Test_Plan ***** */
CREATE TABLE Test_Plan
(
    test_planId nvarchar(100) NOT NULL
        CONSTRAINT PK_test_planId PRIMARY KEY CLUSTERED,
    CONSTRAINT FK_Planning_Test_Plan FOREIGN KEY
        (Test_PlanId) REFERENCES Planning(planningId)
);
/* ***** DMAP ***** */
CREATE TABLE DMAP
(
    dmapId nvarchar(100) NOT NULL
        CONSTRAINT PK_dmapId PRIMARY KEY CLUSTERED,
    CONSTRAINT FK_Planning_DMAP FOREIGN KEY
        (DMAPId) REFERENCES Planning(planningId)
);

```

```

/* ***** Requirements ***** */
CREATE TABLE Requirements
(
    requirementsId nvarchar(100) NOT NULL
        CONSTRAINT PK_requirementsId PRIMARY KEY CLUSTERED,
    CONSTRAINT FK_Planning_Requirements FOREIGN KEY
        (RequirementsId) REFERENCES Planning(planningId)
);
/* ***** System_Issues ***** */
CREATE TABLE System_Issues
(
    system_issuesId nvarchar(100) NOT NULL
        CONSTRAINT PK_system_issuesId PRIMARY KEY CLUSTERED,
    systemIssueID nvarchar(100) NOT NULL UNIQUE,
    CONSTRAINT FK_Analysis_System_Issues FOREIGN KEY
        (System_IssuesId) REFERENCES Analysis(analysisId)
);
/* ***** MOEs_MOPs ***** */
CREATE TABLE MOEs_MOPs
(
    moes_mopsId nvarchar(100) NOT NULL
        CONSTRAINT PK_moes_mopsId PRIMARY KEY CLUSTERED,
    moeName nvarchar(100) NOT NULL,
    mopName nvarchar(100) NOT NULL,
    CONSTRAINT FK_Analysis_MOEs_MOPs FOREIGN KEY
        (MOEs_MOPsId) REFERENCES Analysis(analysisId)
);
/* ***** Quicklook_Report ***** */
CREATE TABLE Quicklook_Report
(
    quicklook_reportId nvarchar(100) NOT NULL
        CONSTRAINT PK_quicklook_reportId PRIMARY KEY CLUSTERED,
    qlReportName nvarchar(100),
    CONSTRAINT FK_Reporting_Quicklook_Report FOREIGN KEY
        (Quicklook_ReportId) REFERENCES Reporting(reportingId)
);
/* ***** Final_Report ***** */
CREATE TABLE Final_Report
(
    final_reportId nvarchar(100) NOT NULL
        CONSTRAINT PK_final_reportId PRIMARY KEY CLUSTERED,
    finalreportName nvarchar(100) NOT NULL,
    CONSTRAINT FK_Reporting_Final_Report FOREIGN KEY
        (Final_ReportId) REFERENCES Reporting(reportingId)
);
/* ***** TRR ***** */
CREATE TABLE TRR
(
    trrId nvarchar(100) NOT NULL
        CONSTRAINT PK_trrId PRIMARY KEY CLUSTERED,
    CONSTRAINT FK_Reporting_TRR FOREIGN KEY
        (TRRId) REFERENCES Reporting(reportingId)
);

```

```

/* ***** Program_Manager ***** */
CREATE TABLE Program_Manager
(
    program_managerId nvarchar(100) NOT NULL
        CONSTRAINT PK_program_managerId PRIMARY KEY CLUSTERED,
    CONSTRAINT FK_Non_Admin_Program_Manager FOREIGN KEY
        (Program_ManagerId) REFERENCES Non_Admin(non_adminId)
);
/* ***** Tester ***** */
CREATE TABLE Tester
(
    testerId nvarchar(100) NOT NULL
        CONSTRAINT PK_testerId PRIMARY KEY CLUSTERED,
    testerName nvarchar(100) NOT NULL,
    CONSTRAINT FK_Non_Admin_Tester FOREIGN KEY
        (TesterId) REFERENCES Non_Admin(non_adminId)
);
/* ***** Analyst ***** */
CREATE TABLE Analyst
(
    analystId nvarchar(100) NOT NULL
        CONSTRAINT PK_analystId PRIMARY KEY CLUSTERED,
    analystName nvarchar(100) NOT NULL,
    CONSTRAINT FK_Non_Admin_Analyst FOREIGN KEY
        (AnalystId) REFERENCES Non_Admin(non_adminId)
);
/* ***** Data_Manager ***** */
CREATE TABLE Data_Manager
(
    data_managerId nvarchar(100) NOT NULL
        CONSTRAINT PK_data_managerId PRIMARY KEY CLUSTERED,
    dataManagerName nvarchar(100) NOT NULL,
    CONSTRAINT FK_Non_Admin_Data_Manager FOREIGN KEY
        (Data_ManagerId) REFERENCES Non_Admin(non_adminId)
);
/* ***** Raw ***** */
CREATE TABLE Raw
(
    rawId nvarchar(100) NOT NULL
        CONSTRAINT PK_rawId PRIMARY KEY CLUSTERED,
    rawDataName nvarchar(100) NOT NULL,
    CONSTRAINT FK_System_Data_Raw FOREIGN KEY
        (RawId) REFERENCES System_Data(system_dataId)
);
/* ***** Processed ***** */
CREATE TABLE Processed
(
    processedId nvarchar(100) NOT NULL
        CONSTRAINT PK_processedId PRIMARY KEY CLUSTERED,
    processDataName nvarchar(100) NOT NULL,
    CONSTRAINT FK_System_Data_Processed FOREIGN KEY
        (ProcessedId) REFERENCES System_Data(system_dataId)
);

```

```

/* ***** Relationship Tables ***** */
/* ***** provides ***** */
CREATE TABLE AnalystProvides
(
    analystId nvarchar(100) NOT NULL,
    final_reportId nvarchar(100) NOT NULL,
    CONSTRAINT PK_analystProvides PRIMARY KEY CLUSTERED
        (
            analystId ASC,
            final_reportId ASC
        ),
    CONSTRAINT FK_provides_Analyst FOREIGN KEY
        (AnalystId) REFERENCES Analyst(analystId),
    CONSTRAINT FK_provides_Final_Report FOREIGN KEY
        (Final_ReportId) REFERENCES Final_Report(final_reportId)
);
/* ***** Identifies ***** */
CREATE TABLE AnalystIdentifies
(
    analystId nvarchar(100) NOT NULL,
    system_issuesId nvarchar(100) NOT NULL,
    CONSTRAINT PK_analystIdentifies PRIMARY KEY CLUSTERED
        (
            analystId ASC,
            system_issuesId ASC
        ),
    CONSTRAINT FK_Identifies_Analyst FOREIGN KEY
        (AnalystId) REFERENCES Analyst(analystId),
    CONSTRAINT FK_Identifies_System_Issues FOREIGN KEY
        (System_IssuesId) REFERENCES System_Issues(system_issuesId)
);
/* ***** creates ***** */
CREATE TABLE AnalystCreates
(
    analystId nvarchar(100) NOT NULL,
    quicklook_reportId nvarchar(100) NOT NULL,
    CONSTRAINT PK_analystCreates PRIMARY KEY CLUSTERED
        (
            analystId ASC,
            quicklook_reportId ASC
        ),
    CONSTRAINT FK_creates_Analyst FOREIGN KEY
        (AnalystId) REFERENCES Analyst(analystId),
    CONSTRAINT FK_creates_Quicklook_Report FOREIGN KEY |
        (Quicklook_ReportId) REFERENCES Quicklook_Report(quicklook_reportId)
);

```

```

/* ***** uploads ***** */
CREATE TABLE Data_ManagerUploads
(
    data_managerId nvarchar(100) NOT NULL,
    system_dataId nvarchar(100) NOT NULL,
    CONSTRAINT PK_data_managerUploads PRIMARY KEY CLUSTERED
        (
            data_managerId ASC,
            system_dataId ASC
        ),
    CONSTRAINT FK_uploads_Data_Manager FOREIGN KEY
        (Data_ManagerId) REFERENCES Data_Manager(data_managerId),
    CONSTRAINT FK_uploads_System_Data FOREIGN KEY
        (System_DataId) REFERENCES System_Data(system_dataId)
);
/* ***** uses ***** */
CREATE TABLE AnalystUses
(
    analystId nvarchar(100) NOT NULL,
    processedId nvarchar(100) NOT NULL,
    CONSTRAINT PK_analystUses PRIMARY KEY CLUSTERED
        (
            analystId ASC,
            processedId ASC
        ),
    CONSTRAINT FK_uses_Analyst FOREIGN KEY
        (AnalystId) REFERENCES Analyst(analystId),
    CONSTRAINT FK_uses_Processed FOREIGN KEY
        (ProcessedId) REFERENCES Processed(processedId)
);
/* ***** calculates ***** */
/* The table captures a many-to-many relationship between analysts MOEs_MOPs entities.*/
/* An analyst can be associated with multiple MOEs_MOPs entities and vice versa. */

CREATE TABLE AnalystCalculates
(
    analystId nvarchar(100) NOT NULL,
    moes_mopsId nvarchar(100) NOT NULL,
    CONSTRAINT PK_analystCalculates PRIMARY KEY CLUSTERED
        (
            analystId ASC,
            moes_mopsId ASC
        ),
    CONSTRAINT FK_calculates_Analyst FOREIGN KEY
        (AnalystId) REFERENCES Analyst(analystId),
    CONSTRAINT FK_calculates_MOEs_MOPs FOREIGN KEY
        (MOEs_MOPsId) REFERENCES MOEs_MOPs(moes_mopsId)
);

```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- Bedford, T. and R. Cooke, R. 2001. "Probabilistic Risk Analysis: Foundations and Methods," Cambridge University Press, London.
- Cambridge Semantics. n.d. "Owl 101." Accessed Aug 4, 2023.
<https://cambridgesemantics.com/blog/semantic-university/learn-owl-rdfs/owl-101/>
- CSULB. n.d. "Basic SQL statements: DDL and DML." Accessed Sep 11, 2023.
<https://web.csulb.edu/colleges/coe/cecs/dbdesign/dbdesign.php?page=sql/ddldml.php>
- Dam, Steven, "Embedding Command and Control, Information Assurance, and Other Decision Points in Model-Based Systems Engineering (MBSE) Analyses" (PowerPoint presentation, SPEC Innovations, October 24, 2012),
- Dam, S. 2019. "Real MBSE." SPEC Innovations. Manassas, VA.
- Dickson, C. 2020. "Data Analysis in MySQL - Operators, Joins and More in Relational Databases." Accessed Aug 24, 2023. <https://towardsdatascience.com/data-analysis-in-mysql-operators-joins-and-more-in-relational-databases-26c0a968e61e>
- Ernestas, V and L. Nemuraite. 2006. "Transforming ontology representation from OWL to relational database." Information Technology and Control.
- Friedenthal, S., A. Moore, and R. Steiner. 2015. "A Practical Guide to SysML: 3rd edition. Elsevier, Inc., Waltham, MA.
- GENESYS Online Help. n.d.a. "Requirements Diagram." Accessed December 14, 2022.
https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Views/Requirements_Diagram.htm
- GENESYS Online Help. n.d.b. "Spider Diagram." Accessed December 14, 2022.
https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Views/Spider_Diagram.htm
- Google. n.d. "What is a relational database?" Accessed Aug 25, 2023.
<https://cloud.google.com/learn/what-is-a-relational-database#section-1>
- Innoslate Help Center. 2022a. "Action Diagram." User's Guide. Last modified May 5th, 2022.
<https://help.innoslate.com/users-guide/diagrams-view/types/lml/action-diagram>

Innoslate Help Center. 2022b. "Requirement Diagram." User's Guide. Last modified May 5th, 2022. <https://help.innoslate.com/users-guide/diagrams-view/types/sysml/requirement-diagram>.)

Innoslate Help Center. 2022c. "Spider Diagram." User's Guide. Last modified May 6th, 2022. <https://help.innoslate.com/users-guide/diagrams-view/types/lml/spider-diagram>.

Innoslate. 2018a. "Internal Block Diagram (IBD)." Last modified on September 13th, 2018. <https://help.innoslate.com/users-guide/diagrams-view/types/sysml/internal-block-diagram>.

Innoslate. 2018b. "State Machine Diagram." Last modified on September 13th, 2018. <https://help.innoslate.com/users-guide/diagrams-view/types/sysml/state-machine-diagram>.

Innoslate. 2022a. "Activity Diagram." Last modified on May 3rd, 2022. <https://help.innoslate.com/users-guide/diagrams-view/types/sysml/activity-diagram>.

Innoslate. 2022b. "Asset Diagram." Last modified on May 6th, 2022. <https://help.innoslate.com/users-guide/diagrams-view/types/lml/asset-diagram>.

Innoslate. 2022c. "Block Definition Diagram (BDD)." Last modified on May 6th, 2022. <https://help.innoslate.com/users-guide/diagrams-view/types/sysml/block-definition-diagram>.

Innoslate. 2022d. "Hierarchy Diagram." Last modified on May 6th, 2022. <https://help.innoslate.com/users-guide/diagrams-view/types/general/hierarchy-chart>.

Innoslate. 2022e. "Sequence Diagram." Accessed November 28, 2022. <https://help.innoslate.com/users-guide/diagrams-view/types/sysml/sequence-diagram>.

Innoslate. 2022f. "Action Diagram." Privacy & Terms. Last modified on May 5th, 2022. <https://help.innoslate.com/users-guide/diagrams-view/types/lml/action-diagram>.

Lifecycle Modeling Language. 2015. "Lifecycle Modeling Language (LML) Specification." Accessed December X, 2021. https://lifecyclemodeling.org/wp-content/uploads/2021/01/LML_Specification_1_1.pdf.

Mahmudi, Kamal, Inggriani Liem, Saiful Akbar. 2018. "Ontology to relational database transformation for web application development and maintenance." Journal of

Physics: Conference Series 971 (December). DOI 10.1088/1742-6596/971/1/012031

No Magic Inc. 2011. "User Manual."

<https://www.magicdraw.com/files/manuals/MagicDraw%20UserManual.pdf>.

No Magic, Inc. 2015. "SYSML PLUGIN, 18.1, user guide."

<https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/CATIA/NoMagic/pdf/sysml-plugin-user-guide.pdf>.

Object Management Group. 2019. "OMG Systems Modeling Language (OMG SysML™)." formal/19-11-01, OMG.

<https://sysml.org/.res/docs/specs/OMGSysML-v1.6-19-11-01.pdf>.

Office of the Deputy Assistant Secretary of Defense for Systems Engineering (DASD SE). 2018, June. "Department of Defense Digital Engineering Strategy." Department of Defense.

Parlar, Selen. 2019. "Ontology Visualization with Protégé." Last modified October 30, 2019. <https://medium.com/analytics-vidhya/prot%C3%A9g%C3%A9-d533f024087b>

Quantum Leaps. 2015. "Application Note: A Crash Course in UML State Machines." State Machines. http://www.state-machine.com/doc/AN_Crash_Course_in_UML_State_Machines.pdf.

Quatrani, Terry. n.d. "Introduction to UML 2.0." IBM Software Group.

https://www.omg.org/news/meetings/workshops/MDA-SOA-WS_Manual/00-T4_Matthews.pdf.

Smedley, J. 1993. "Distributed Computing Design System (DCDS)." TRW Systems Integration Group. Huntsville, AL.

Störrle, Harald. 2005. "Semantics and Verification of Data Flow in UML 2.0 Activities."

Electronic Notes in Theoretical Computer Science 127(4): 35–52.

<https://doi.org/10.1016/j.entcs.2004.08.046>.

SysML Forum. n.d. "SysML FAQ: Who created SysML and how did it get its name?" Accessed November 28, 2022. <https://sysmlforum.com/sysml-faq/who-created-sysml.html>.

Vaneman, Warren, Ron Carlson, Corina White, and Raymond J. Stone. 2022. Developing an MBSE Land Domain Construct for Marine Corps System Command. Report No. NPS-SE-22-004. <https://apps.dtic.mil/sti/pdfs/AD1184547.pdf>

- Vitech Corporation. n.d.a “Activity Diagram.” Accessed November 28, 2022.
https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Views/Activity_Diagram.htm.
- Vitech Corporation. n.d.b "Block Definition Diagram (BDD)." Accessed November 28, 2022.
[https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Views/Structure_Block_Definition_Diagram_\(BDD\).htm](https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Views/Structure_Block_Definition_Diagram_(BDD).htm).
- Vitech Corporation. n.d.c “Enhanced Function Flow Block Diagram (EFFBD).” Accessed November 28, 2022.
[https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Views/Enhanced_Function_Flow_Block_Diagram_\(EFBD\).htm](https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Views/Enhanced_Function_Flow_Block_Diagram_(EFBD).htm).
- Vitech Corporation. n.d.d "Hierarchy Diagram." Accessed November 28, 2022.
https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Views/Hierarchy_Diagram.htm.
- Vitech Corporation. n.d.e “Internal Block Diagram.” Accessed November 28, 2022.
[https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Views/Internal_Block_Diagram_\(IBD\).htm](https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Views/Internal_Block_Diagram_(IBD).htm).
- Vitech Corporation. n.d.f “Physical Block Diagram.” Accessed November 28, 2022.
https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Views/Physical_Block_Diagram.htm.
- Vitech Corporation. n.d.g "Sequence Diagram." Accessed November 28, 2022.
https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Views/Sequence_Diagram.htm.
- Vitech Corporation. n.d.h “State Transition Diagram.” Accessed November 28, 2022.
https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Views/State_Transition_Diagram.htm.
- Vysniauskas, Ernestas, Lina Nemuraite. 2006. “Transforming Ontology Representations from OWL to Relational Database.” Information Technology and Control 35 No. 3 (September). <https://itc.ktu.lt/index.php/ITC/article/view/11779>
- WC3. 2012. “OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition).” Accessed Aug 25, 2023.
<https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/>.

WC3. 2013. "OWL." Accessed Aug 25, 2023. <https://www.w3.org/OWL/>

Wong, Kenneth. 2006. "What Grounded the Airbus A380?" cadalyst. Last modified on December 5, 2006. <https://www.cadalyst.com/management/what-grounded-airbus-a380-5955>.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California
3. Office of Research and Innovation, Code 400
Naval Postgraduate School
Monterey, CA 93943