

WRT 2508



DOD ENTERPRISE ARCHITECTURE GUIDE DEVELOPMENT SUPPORT

SEPTEMBER 9, 2026

► **PRINCIPAL INVESTIGATOR**
Dr. Dinesh Verma
STEVENS INSTITUTE OF TECHNOLOGY

► **CO-PRINCIPAL INVESTIGATOR**
Mr. Tom McDermott
STEVENS INSTITUTE OF TECHNOLOGY

► **KEY CONTRIBUTOR**
Dr. Valerie Sitterle
STEVENS INSTITUTE OF TECHNOLOGY

TECHNICAL REPORT NO.
SERC-2026-TR-017

CONTRACT NO.
HQ0034-24-D-0023

TASK ORDER NO.
HQ003425FE312

SPONSOR

► Office of the Under Secretary of War
for Research and Engineering
(OUSW(R&E))



Stevens Institute of Technology
1 CASTLE POINT ON HUDSON
HOBOKEN, NJ 07030

Disclaimer: The views, findings, conclusions, and recommendations expressed in this material are solely those of the author(s) and do not necessarily reflect the views or positions of the United States Government (including the Department of War (DoW) and any government personnel).

UNCLASSIFIED

DISCLAIMER

Copyright © 2026 Stevens Institute of Technology, Systems Engineering Research Center

The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

This material is based upon work supported, in whole or in part, by the U.S. Department of War (DoW) through the Office of the Under Secretary of War for Research and Engineering (OUSW(R&E)) under Contract HQ0034-24-D-0023, Task Order (TO) 0312.

No Warranty.

The views, findings, conclusions, and recommendations expressed in this report are solely those of the authors and do not necessarily reflect the views or positions of the United States Government (including the DoW and any government personnel), [or] the Stevens Institute of Technology.

This material has been approved for public release and unlimited distribution.

RESEARCH TEAM

Name	Organization	Labor Category
Dinesh Verma	Stevens	PI
Tom McDermott	Stevens	Co-PI
Valerie Sitterle	Stevens	Other Professional

TABLE OF CONTENTS

Disclaimer	ii
Research Team.....	ii
Table of Contents	iii
List of Figures	iv
List of Tables	iv
Executive Summary	1
1. Project Overview	4
2. Research Activities	5
2.1. Research Foci in Alignment with Project.....	5
2.2. Key Recommendations	5
2.3. More Detailed Task Activity and Technical Products/ Results.....	7
2.3.1. Foundational research to define and characterize warfighting types of SoS and the nature of architecture in that context.....	7
2.3.2. Characterizing current practice and associated MPTs for SoS architecting.....	9
2.3.3. Initial Ontology and Representation Pilot	10
2.3.4. Framing Recommendations for SoS Representational-to-Analytical Approach ..	12
3. Conclusion.....	16
4. Project Timeline & Transition Plan	17
Appendix A: List of Publications Resulted	19
Appendix B: Cited and Related References.....	20
Appendix C: Re-Visiting the Definition of Systems of Systems (SoS) in a Military Context and the Role of Architecture for SoS	22
SoS Defining Criteria and Characteristics.....	22
SoS Governance Typology	22
SoS Architecture and Enterprise Architecture (EA)	23
Clarifying Practice in Military Contexts: Capabilities, Missions, and SoS	24
Key Military SoS Architecture Uses	25
Key Effects-Based Functional Types of Military SoS	27
Are these effects-based functional system configurations considered SoS by the criteria? ...	28
Appendix D: Effects-Based Functional Types of Military SoS in the Context of Architecture Use.....	30
Appendix E: Pilot SoS Description with Condition-Based Heterarchical Relationship.....	33
Overview	33
Starting Point: 3 UAF Operational Views	33
UAF Implementation Gaps and Tool-Based Additions.....	34
SoS-Specific Extensions and the Need for Reification	35
Extended Ontological Representation	38
Simple Notional Scenario	39

LIST OF FIGURES

Figure 1. Illustration of approach to consistent and extensible semantic foundation from which we can generate many analytical views representing and analyzing adaptive warfighting SoS	13
Figure 2. ArbitraryConnector class from UAF 1.2 Domain Metamodel (DMM)	34
Figure 3. SoS pilot ontology shown in Protégé.....	38
Figure 4. Notial scenario for pilot investigation.....	39
Figure 5. Illustration of scenario structure showing condition-based authority shift.....	40

LIST OF TABLES

Table 1. Primary Warfighting Capability SoS Types (excerpt from full table in Appendix)	7
Table 2. Three primary effects-based configurations of systems for warfighting contexts	27
Table 3. SoS characterization of the primary effects-based system configurations	29
Table 4. SoS functional capability type versus architecture use matrix	30
Table 5. Three operational views from UAF selected for pilot	33
Table 6. New classes required for pilot SoS example	35
Table 7. New properties required for pilot SoS example	35
Table 8. Performers and Basic Scenario.....	39

EXECUTIVE SUMMARY

This task is to support the work of the Joint Architectures for Capabilities and Systems (JACS) Directorate, in coordination with the Department of War (hereafter, Department) Enterprise Engineering and Architectures Community of Practice (COP). Through this task, SERC will provide guidance and recommended best practices on enterprise architectures (EAs) for military Systems of Systems (SoS) for Department practitioners in the field. In this effort, the SERC supports the development of the common foundation by reviewing current literature in the field of SoS enterprise architectures to identify (a) general pain points related to the development and use of EAs for SoS, (b) the current state of EA for SoS in the context of military use and how they are being used, (c) the current state of the research in this area the gaps remaining for this context, and (d) how might EAs evolve to become a construct for Government-Contractor co-creation and use.

SERC's research discovered that architecting warfighting SoS requires more than governance typology. There are three primary types of SoS for warfighting capabilities, each with their own distinct needs for representation: kill chains, kill webs, and continuous state (distributed, interconnected) networks. Operationally these are developed and deployed (right to left) as: (a) first, a distributed interconnected network reflecting persistent infrastructure and capabilities as continuous-state operational architecture; (b) a kill web, as a resilient multi-path architecture capable of dynamic orchestration and adaptation to accomplish a mission; and (c) a kill chain as a linearized sequence architecture that is an optimized instantiation from a web pathway selected to execute the mission or a part of it. Each exhibit distinct structural and functional relationships that impact how it can be best represented for what specific use. At present, all three types are not represented effectively in architecture practice.

Further, architecting and EA blur for SoS. The practical distinction between system/SoS architecture and enterprise architecture becomes increasingly difficult to maintain as the boundary of the SoS expands, just as it does for any expansive and complex system. Consequently, who the architecture serves and what decision it supports is not cleanly fixed by DoD guidance. This challenge is compounded by the lack of clarity across the purpose of architecting or EA as exemplified in how it is fragmented across use as a strategic framework, a technical descriptive architecture, and a governance blueprint; and that list may not be complete.

SERC's work found that current architecting frameworks are not sufficient to represent the nature of all three primary SoS warfighting capability types for deeper analytical and decision-making needs. SERC's analysis identifies three key gaps military SoS architectural representation and practice:

- **Gap 1 – Inadequate semantic foundations.** The semantic foundations across current architectural frameworks, their underlying metamodels, and the tools that employ them lack the expressive capacity to represent a heterarchical SoS and support causal reasoning across the necessary dimensions.

- **Gap 2 – Descriptive models lack required explanatory, analytical, and predictive power.** Current architecture practice relies on MBSE languages and tools designed for descriptive fidelity, yet the SoS problem space requires deeper, explanatory analytical capabilities and support for causal reasoning. Current EA tools do not produce inherently executable models of dynamic SoS mission performance; military SoS decision makers need to understand performance of capabilities integrated in architecture and not just the structure of architectural components.
- **Gap 3 – Architectural tempo lags operational tempo.** The tempo of operational adaptation and enterprise decision making needs now exceeds the tempo of architectural description. EA frameworks were designed for environments that change more slowly than a modeling effort takes to complete, i.e., on enterprise planning timescales. In the military SoS context, this assumption fails. Adversarial adaptation, funding volatility, and constituent system change all outpace current enterprise cycles. Automation of EA will be a necessary enabler.

Anchoring on key principles identified in this research, SERC recommends the following:

- 1) **To be useful at speed, architecture must shift from descriptive (artifact-focused with minimal analytical mapping capabilities) to analytical, decision-centric architecting that directly supports computational queries and causal reasoning.** This changes the notion of architectural patterns from structural building blocks to composable analytical threads. In turn, this moves the concept of architectural modeling to one where architectures are directly computable (i.e., machine-readable and machine-understandable), extensible and interoperable with data from other model types or level of fidelity assessment data, and rapidly able to be generated and adapted to keep pace with the warfighting needs.
- 2) **The military SoS architecting problem space needs a formal semantic ontology captured as a subject-verb-object as a necessary foundation of an architecture model that will enable it to be computable and to link the physical and functional representations of the architecture to analytical methods.** This directly supports a shift to analytical and decision-centric architecture. Further, we need an approach with the lowest friction — including friction in integrating with other tools and data — to characterize SoS architectures in a way that directly supports analytical needs.
- 3) **We need a scalable, integrable semantic approach capable of representing all three SoS capability type architectures that directly supports generation of network-level mathematical models and analytics for both operational behavior and capability portfolio investment decisions.** We need a way to anchor a semantic foundation through which we can consistently define capabilities, roles, functions, attributes of both things and their relationships that allows us to (a) directly generate different representations in the context of the problem we are trying to solve, and to (b) support advanced analytics, dynamic

behavioral evaluations, targeted queries, adaptivity vs resource allocation, etc. across all three SoS capability types.

- 4) **We need Enterprise Architecture representations that expand beyond the enterprise; yet we also need foundational representations of SoS governance that are not oriented toward classical SoS type but are more oriented towards SoS mission outcomes.** Namely, how do different models of governance impact successful mission outcomes as represented in Table 1.

SERC has developed an approach for a scalable, integrable semantic representation of architecture across all three SoS capability types (kill chains, kill webs, and continuous-state networks) that supports network-level analytics and can directly inform and link mission engineering with capability portfolio investment decisions.

SERC's approach uses RDF 1.2, a semantic web standard that lets conditions such as jamming state, mission phase, and condition-based authorization attach directly to a relationship between two elements, circumventing the reification workaround object that has made this kind of conditional detail costly to represent in prior semantic modeling efforts. This also enable representation of concepts not captured in current EA or SoS frameworks and tooling. Because the RDF 1.2 format pairs with formal logic reasoning, the model can be checked automatically for consistency as it is built, avoiding manual review. From this semantic basis, we can generate any number of capability graphs describing what any of the three SoS capability types can do for a given mission or scenario, and any number of platform graphs instantiating a given capability graph with specific systems under specific conditions at a given time. This same foundation directly generates the structures needed for multiple types of mathematical analysis — methods suited to tracing alternate pathways, computing feasible reconfigurations, and tracking how the network changes over time — without building a separate, disconnected model for each. This provides a consistent foundation for network-level analysis, including node criticality, redundancy, and adaptivity under changing conditions, that ties directly back to the same architecture data used for assessing mission outcomes and supporting capability portfolio investment decisions.

1. PROJECT OVERVIEW

This task is to support the work of the Joint Architectures for Capabilities and Systems (JACS) Directorate, in coordination with the Department of War (hereafter, Department) Enterprise Engineering and Architectures Community of Practice (COP). Through this task, SERC will provide guidance and recommended best practices on enterprise architectures (EAs) for military Systems of Systems (SoS) for Department practitioners in the field. The goal is to develop an enterprise-wide strategy for systems of systems (SoS) architecting to ensure successful integration and interoperability of the SoS. During the initial planning phases by JACS, the team discovered that systems engineering and architectures across the enterprise lack common terminology, definitions, and methods. Establishing these standards as well as a common framework for requirements will provide the JACS and Department practitioners with the necessary foundations for success in using a common architecture for successful integration and interoperability.

This effort relates to *WRT-2504 DoD Systems Engineering Guide for Systems of Systems Revision, 2025-2026* but does not duplicate its work. This effort, ***WRT-2508 DoD Enterprise Architecture Guide Development Support***, is oriented toward enterprise architectures for the SoS. Further, JACS specifically cites the focus of “guidance, strategy, and standards for enterprise-level SoS engineering and architecture.”¹.

The original tasking for support developing a guidebook to provide guidance and recommended best practices on enterprise architectures (EAs) for Systems of Systems (SoS) to Department practitioners in the field has been put on hold. Instead, the OUSD(R&E)/SE&A Joint Architectures for Capabilities and Systems (JACS) Directorate needs to pursue prototype solutions from small business that address persistent technical and procedural challenges encountered by engineering and architecture practitioners supporting acquisition programs, specific to EA for SoS challenges.² SERC was asked to identify the highest priority challenges in this space, characterize these challenges in the context of how solutions might be realized, and develop a path forward as a foundation implementable and scalable through a combination of vendor support and additional research development required.

In this effort, the SERC supports the development of the common foundation by reviewing current literature in the field of SoS enterprise architectures to identify (a) general pain points related to the development and use of EAs for SoS, (b) the current state of EA for SoS in the context of military use and how they are being used, (c) the current state of the research in this area the gaps remaining for this context, and (d) how might EAs evolve to become a construct for Government-Contractor co-creation and use. Organizational structures, needs, and foci are dynamic; therefore, precise needs for this task may shift. The SERC and Government will discuss key findings and recommendations in this document to refine and prioritize next steps.

¹ <https://www.cto.mil/sea/jacs/>

² <https://marketplace.gocolosseum.org/public/challenges/ec77dae8-d6b7-4fb4-9d4f-0a64a41e131f>

2. RESEARCH ACTIVITIES

2.1. RESEARCH FOCI IN ALIGNMENT WITH PROJECT

- Foundational research to define and characterize warfighting types of SoS and the nature of architecture in that context:
 - Define and characterize SoS in the context of military warfighting capabilities.
 - Define and characterize architecture and enterprise architecture (EA) for those specific SoS types.
- Characterizing current practice and associated methods, processes, and tools (MPTs) for SoS architecting:
 - Identify the state of current practice and where it is focused.
- Initial Ontology and Representation Pilot
- Framing Recommendations for SoS Representational-to-Analytical Approach:
 - Make recommendations with respect to SoS for warfighting capabilities and the nature of architecting they require.
 - Develop an approach to mitigate the constraints and gaps in current practice (e.g., methods, processes, and tools (MPTs)) for architecting of military SoS
 - Recommend next steps

2.2. KEY RECOMMENDATIONS

Our recommendations are anchored on key, guiding principles SERC identified through our research:

- 1) **To be useful at speed, architecture must shift from descriptive (artifact-focused with minimal analytical mapping capabilities) to analytical, decision-centric architecting that directly supports computational queries and causal reasoning.** This changes the notion of architectural patterns from structural building blocks to composable analytical threads. In turn, this moves the concept of architectural modeling to one where architectures are directly computable (i.e., machine-readable and machine-understandable), extensible and interoperable

with data from other model types or level of fidelity assessment data, and rapidly able to be generated and adapted to keep pace with the warfighting needs.

- 2) **The military SoS architecting problem space needs a formal semantic ontology captured as a subject-verb-object as a necessary foundation of an architecture model that will enable it to be computable and to link the physical and functional representations of the architecture to analytical methods.** This directly supports a shift to analytical and decision-centric architecture. Further, we need an approach with the lowest friction — including friction in integrating with other tools and data — to characterize SoS architectures in a way that directly supports analytical needs.

We revisited the nature of SoS in depth in the literature, SEBoK, associated definitional standards from IEEE, and treatment of SoS in the Department's mission engineering guidebooks and processes. We found that while SoS capability types are not clearly distinguished across these sources, **there are three primary warfighting SoS types, each with different descriptive and analytical needs: (i) kill chains, (ii) kill webs, and (iii) continuous-state integrated networks (e.g., integrated multi-modal sensing, communications, and compute).** These are illustrated in **Table 1** (excerpt below; complete table in Appendix). Operationally these are developed and deployed (right to left) as: (a) first, a distributed interconnected network reflecting persistent infrastructure and capabilities as **continuous-state operational architecture**; (b) a kill web, as a **resilient multi-path architecture** capable of dynamic orchestration and adaptation to accomplish a mission; and (c) a kill chain as a **linearized sequence architecture that is an optimized instantiation** from a web pathway selected to execute the mission or a part of it. This leads to our third and fourth primary recommendations.

- 3) **We need a scalable, integrable semantic approach capable of representing all three SoS capability type architectures that directly supports generation of network-level mathematical models and analytics for both operational behavior and capability portfolio investment decisions.** We need a way to anchor a semantic foundation through which we can consistently define capabilities, roles, functions, attributes of both things and their relationships that allows us to (a) directly generate different representations in the context of the problem we are trying to solve, and to (b) support advanced analytics, dynamic behavioral evaluations, targeted queries, adaptivity vs resource allocation, etc. across all three SoS capability types.
- 5) **We need Enterprise Architecture representations that expand beyond the enterprise; yet we also need foundational representations of SoS governance that are not oriented toward classical SoS type but are more oriented towards SoS mission outcomes.** Namely, how do different models of governance impact successful mission outcomes as represented in Table 1.

Table 1. Primary Warfighting Capability SoS Types (excerpt from full table in Appendix)

Key Characteristics	Primary Effects-Based System Configurations		
	<i>Systems performing discrete, sequential operational effects (kill chain)</i>	<i>Systems performing dynamic, multi-path operational effects (kill web)</i>	<i>Systems performing continuous, foundational state-based effects (distributed, interconnected network)</i>
Structure/ Characterization	Linear sequence: Linear, rigid, and time-critical execution paths where a single node failure breaks the sequence. Success is measured by speed and throughput.	Layered, multipath matrix: Layered, combinatorial, and opportunistic mission structures where inputs and outputs are dynamically routed through alternate network nodes. Success is measured by mission resilience and adaptability.	Non-sequenced, ubiquitous grid: Non-sequenced, ubiquitous, and steady-state information environments providing persistent theater transparency and freedom of action. Success is measured by <i>coverage, latency, and system uptime</i> .
Example	Fires kill chain to produce a kinetic effect on a target	Joint fires kill web to achieve synergistic effects and multipath resilience to produce effect on a target	Integrated sensing/ communications/ compute network

2.3. MORE DETAILED TASK ACTIVITY AND TECHNICAL PRODUCTS/ RESULTS

This section summarizes parts of key research activities and references the Appendices in this report capturing the detail. Additional detail may be found in the whitepapers and technical SERC produced as part of this effort, listed in [Appendix A](#) of this document. Additional technical details from the A008 bi-monthly status reports have been captured in [Appendix C](#), [Appendix D](#), and [Appendix E](#) of this report.

2.3.1. FOUNDATIONAL RESEARCH TO DEFINE AND CHARACTERIZE WARFIGHTING TYPES OF SoS AND THE NATURE OF ARCHITECTURE IN THAT CONTEXT

See [Appendix A](#) and [Appendix B](#) for additional details.

This research activity identified several key takeaways:

- When a heterogeneous networks of systems in a military context is referred to as a system of systems (SoS), it represents a very defined context anchored in governance as described by Maier and INCOSE. In the SoS context, architecture reflects the same variation seen for EA in general (technical, strategic alignment of capabilities to resources, reference architecture for governance and standards, etc.) and often shifts toward systems engineering approaches.

- Architecting warfighting SoS requires more than governance typology. There are three primary types of SoS for warfighting capabilities, each with their own distinct needs for representation: kill chains, kill webs, and continuous state (distributed, interconnected) networks. Each exhibit distinct structural and functional relationships that impact how it can be best represented for what specific use.
- For two of the three primary functional capability SoS configurations relevant to warfighting, capability emerges from the network (these are systems performing dynamic, multi-path operational effects (kill webs) and systems performing continuous, foundational state-based effects (distributed, interconnected networks)). These SoS types therefore require network-level analyses. As military concepts of operations shift from kill chains to kill webs, the analytical focus similarly shifts from optimizing a linear chain of engagements to orchestrating a dynamic network of distributed capabilities.
- Architecting and EA blur for SoS. The practical distinction between system/SoS architecture and enterprise architecture becomes increasingly difficult to maintain as the boundary of the SoS expands, just as it does for any expansive and complex system. Consequently, who the architecture serves and what decision it supports is not cleanly fixed by DoD guidance.
- No single enterprise in today's highly dynamic and hyper-connected environment has full control over all the SoS used in their SoS of interest, and the environment for a military SoS is adversarial and adaptive. Even where a military SoS has a designated central authority, that authority rarely exercises complete managerial or technical control over all constituent systems, external dependencies, operational interactions, and environmental conditions affecting mission performance. This means all aspects of the architecture cannot be fully specified in advance and continue to be effective. The architecture can specify constraints, interfaces, roles, authorities and permissible interactions in advance, but cannot necessarily specify the operational configuration that will be instantiated in a particular future mission state.
- Different uses of architecture require different representation capabilities, and current practice is dominated by one of the capability types of SoS and is heavily influenced by tooling and historical MBSE practice. In practice, architecture, and especially EA, is fragmented across use as a strategic framework, a technical descriptive architecture, and a governance blueprint; and that list may not be complete. Any of these could be either descriptive or analytical, or most likely a combination. For example, a strategic framework would define what capabilities should exist and why, a descriptive architecture would describe what actually exists and how it is organized to meet the strategic framework. A governance blueprint would allocate who can control or change, and an analytical framework predicts how the SoS will behave or perform. Each may require a set of documents, structured models, ontologies, graphs, simulations, and dynamic network models.

- We define three primary architecture uses for warfighting capability SoS: reference, descriptive, and analytical architectures. Just as architecture and EA blur for SoS, the line blurs at the basic level between descriptive architectures and analytical architectures. The nature of architecture description – its substrate or modeling construct – changes what one can do with it. UAF in the current mission engineering process, for example, provides a structured semantic description of an architecture, including behavioral and relational information, but a UAF model is not inherently an executable dynamic model of SoS performance. It is useful to query *in the tool* for relationships and gaps across the UAF views or to export certain sets of conditions, elements, and sequences as inputs for a physics-based simulation such as AFSIM. However, the distinguishing feature is that descriptive architectures do not necessarily provide an execution semantics sufficient to calculate dynamic SoS behavior.

2.3.2. CHARACTERIZING CURRENT PRACTICE AND ASSOCIATED MPTs FOR SoS ARCHITECTING

See [Appendix A](#) and [Appendix C](#) for additional detail.

This research activity on current practice and methods, processes, and tools (MPTs) identified several key takeaways:

- We advocate for a shift from solely descriptive (artifact-focused with minimal analytical mapping capabilities) to analytical, decision-centric architecture to directly support computational queries and causal reasoning. This changes the notion of architectural patterns from structural building blocks to composable analytical threads. In turn, this moves the concept of architectural modeling to one where architectures are directly computable (i.e., machine-readable and machine-understandable), extensible and interoperable with data from other model types or level of fidelity assessment data, and rapidly able to be generated and adapted to keep pace with the warfighting needs.
- Current architecting frameworks are not sufficient to represent the nature of all three primary SoS warfighting capability types for deeper analytical and decision-making needs. Our analysis highlights three major gaps in military SoS architectural representation and practice:
 - **Gap 1 – Inadequate semantic foundations.** The semantic foundations across current architectural frameworks, their underlying metamodels, and the tools that employ them lack the expressive capacity to represent a heterarchical SoS and support causal reasoning across the necessary dimensions.

Relationships in these frameworks carry type and direction but lack behavioral preconditions, temporal dynamics, cross-domain causal semantics, and

contextual variation for heterarchical authority. Without these semantic primitives, decision makers cannot pose “what if” questions that cross domain boundaries.

- **Gap 2 – Descriptive models lack required explanatory, analytical, and predictive power.** Current architecture practice relies on MBSE languages and tools designed for descriptive fidelity, yet the SoS problem space requires deeper, explanatory analytical capabilities and support for causal reasoning.

Current EA tools are not inherently executable models of dynamic SoS mission performance. EA models can encode causal and dependency relationships, but generally do not provide the quantitative, temporal, probabilistic, or intervention semantics required to compute causal effects in complex dynamic SoS. Military SoS decision makers need to understand the performance of capabilities as integrated in the architecture — not just the structure of architectural components. This requires bridging MBSE models with modeling and simulation, test and evaluation data, and decision support tools into a unified analytical capability.

- **Gap 3 – Architectural tempo lags operational tempo.** The tempo of operational adaptation and enterprise decision making needs now exceeds the tempo of architectural description. Automation of EA will be a necessary enabler.

EA frameworks were designed for environments that change more slowly than a modeling effort takes to complete, i.e., on enterprise planning timescales. In the military SoS context, this assumption fails. Adversarial adaptation, funding volatility, and constituent system change all outpace current enterprise cycles. An enterprise whose architectural understanding is locked in an 18-month-old enterprise model is not well-informed for today’s decision environment.

A comprehensive solution for Gaps 1 and 2 does not automatically solve the velocity problem in Gap 3. This is an “architect for X” problem: given a mission, a set of available systems, a communications state, an environment, and this adversary behavior, what (X=stable, robust, resilient, efficient, disruptive, etc.) architecture should we instantiate? They may, if not focused, exacerbate it.

2.3.3. INITIAL ONTOLOGY AND REPRESENTATION PILOT

See [Appendix A](#) and [Appendix E](#) for additional details.

The April 2026 whitepaper found that current EA modeling frameworks, including the Unified Architecture Framework (UAF) are able to represent heterarchical structures but

not heterarchical behavioral mechanisms. For example, in UAF, an association between two Resource or ResourcePerformer elements is symmetric. Despite the current tooling conventions that represent this as a tree hierarchy, UAF's metamodel does not require a parent-child topology. However, UAF has no first-class semantic constructs for peer-governed authority, negotiated interfaces (as for dynamic command negotiated interfaces), or context-dependent precedence. Therefore, models in these frameworks can represent what configurations and behaviors exist but will struggle to represent what configurations will form over condition X, what behaviors will emerge, or how the underlying relational topology will change. Consequently, a model built in one of these frameworks may be internally consistent by the framework's own rules yet structurally misrepresent how a military SoS governs itself and behaves in operation.

We consequently developed a notional proof of concept to better evaluate some of the recommendations in that whitepaper. We scoped the work to focus on the dynamic and heterarchical aspects of the SoS problem and how they might be represented. Specifically, we developed a proof of concept to address gaps not representable using native UAF concepts that required augmentation with custom, SoS classes and properties. The proof of concept was successful and suggests directions for future work. The pilot effort investigates how to describe a warfighting SoS with peer-governed authority, negotiated interfaces, or context-dependent precedence.

This work takes a single, tractable slice of UAF (three views from the Operational viewpoint) and demonstrates that (a) these views can be extracted as a formal OWL ontology, and (b) that ontology can be extended with new semantic primitives that represent heterarchical governance relationships absent from UAF's native metamodel. We approached this using RDF and OWL, mainly due to known usage in this context and familiarity with it due to its ubiquity in the computational ontology world, and because that ecosystem provides pathways for formal decision logic reasoning and consistency checking. For a more extensive example of these methods, refer to Mogahed and Mansouri³.

Overall, this pilot demonstrated the importance of being able to identify condition-based authority shifts and the mission contexts that activate them for warfighting SoS. How a structure can perform is critical to design, interface and communication/computing relationship specification, and selection of capabilities in an SoS to provide the necessary balance of redundancy and resilience for contested actions with an adversary. To represent all three types of warfighting SoS, we need to describe relationships with attributes. This capability will enable us to characterize performance of an SoS in contested conditions, e.g., when a certain type of communication drops out.

While this solved one problem, a key limitation became apparent if faced with increased scale: the need for reification. The reason for reification is that you cannot attach a condition to a plain edge in RDF/OWL, and so we needed to promote the relationship itself to a node so that it could carry its own outgoing properties. This reification workaround is a

³ Mohamed Mogahed and Mo Mansouri (2026). An Ontology-Based Architecture for Interoperable Healthcare Systems-of-Systems: Structure, Interaction Patterns, and Covenant-Based Governance, Systems, 14: 376.

consequence of choosing RDF/OWL for representation, and a concept not present at all using UAF. It is not, however, an inherent constraint of graphs.

This realization led to the ontological representation-to-analytical approach described in the following section.

2.3.4. FRAMING RECOMMENDATIONS FOR SoS REPRESENTATIONAL-TO-ANALYTICAL APPROACH

A key objective when considering alternative approaches to architecture representation is, in the recognition that there is no one-size-fits all answer, ensuring the representation (its data and relationships) is inherently combinable with other data and schema to support decision making across more than one stakeholder or use.

We recommend graph schema together with Semantic Web and Linked Data standards as a bridge to facilitate achieving as much as a unified analytical substrate as possible, scoped by an explicit decision taxonomy that defines what data, relationships, and performance measures each critical decision class requires. **More foundationally, we recommend that a formal semantic ontology captured as a subject-verb-object is a necessary component of an architecture model to enable it to be computable and to link the physical and functional representations of the architecture to analytical methods.**

From the prior work and the guiding principles expressed in section 2.2, SERC developed an approach that leverages Semantic Web technologies to build an integrated representation-analytics methodology supporting the full range of analyses at the intersection of mission capability and portfolio investment for complex SoS across all three capability types (kill chains, kill webs, and continuous integrated networks). Our approach uses RDF-star, a semantic web standard that lets conditions such as jamming state, mission phase, and authorization attach directly to a relationship between two elements, circumventing the workaround object that has made this kind of conditional detail costly to represent in prior semantic modeling efforts. RDF-star itself is less a solution to making architecture executable as it is to the highly important problem of maintaining semantic traceability between architectural assertions and analytical evidence. Because this format pairs with formal logic reasoning, the model can be checked automatically for consistency as it is built, avoiding manual review.

From this single semantic foundation, we can generate any number of capability graphs describing what the kill web can do for a given mission or scenario, and any number of platform graphs instantiating a given capability graph with specific systems under specific conditions at a given time. Comparing a capability graph to its platform instantiation shows precisely where a single multi-function system covers more than one part of the web.

This same foundation directly generates the structures needed for multiple types of mathematical analysis — methods suited to tracing alternate pathways, computing

feasible reconfigurations, and tracking how the network changes over time — without building a separate, disconnected model for each. This provides one consistent foundation for network-level analysis, including node criticality, redundancy, and adaptivity under changing conditions, that ties directly back to the same architecture data used for capability portfolio investment decisions.

This is illustrated in Figure 1.

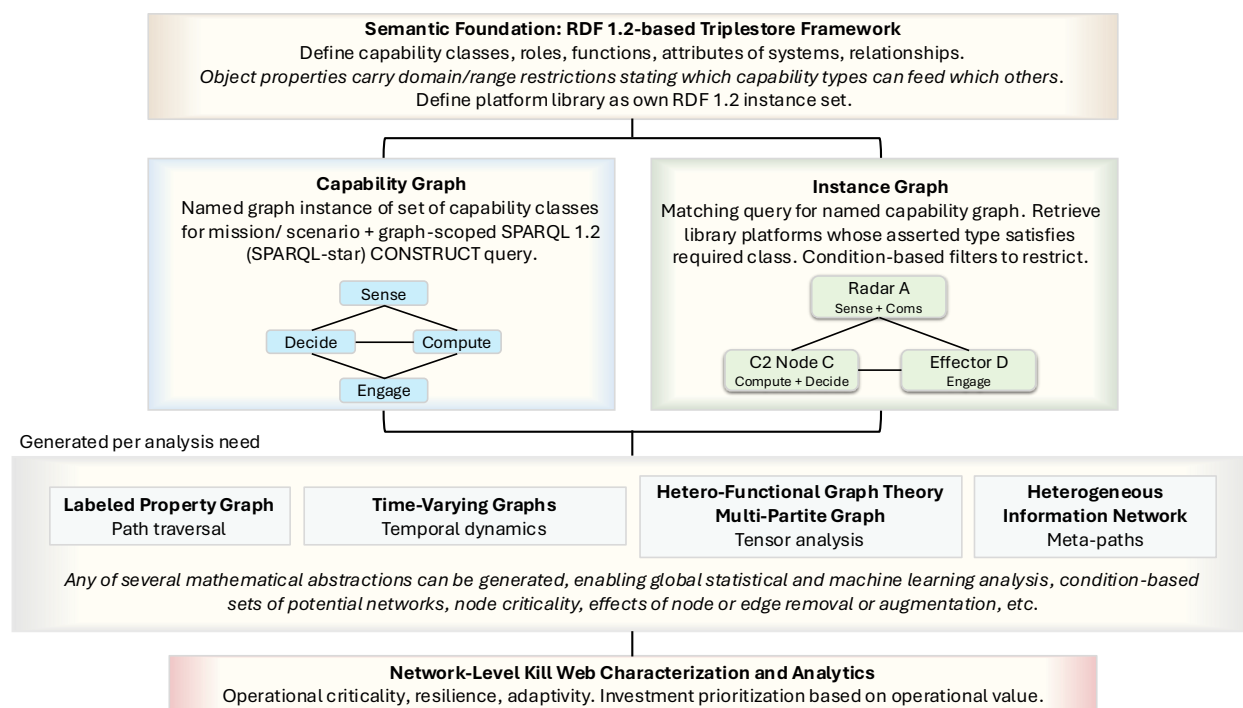


Figure 1. Illustration of approach to consistent and extensible semantic foundation from which we can generate many analytical views representing and analyzing adaptive warfighting SoS

Additional detail on this approach is provided below.

Representational Approach:

- We propose an approach using newer tooling based off a newer (not completely finalized) W3C candidate recommendation, RDF 1.2 (once called RDF* and now RDF-star). RDF 1.2 extends RDF by letting properties attach directly to a triple ($\langle\langle\text{subject- predicate- object}\rangle\rangle$ condition value), eliminating the reification workaround required in plain RDF/OWL noted in our prior report. A W3C Candidate Recommendation Snapshot means the technical design is locked, stable, and ready for industry-wide testing, but it hasn't received the final administrative stamp. <https://www.w3.org/TR/rdf12-concepts/>
- Several existing tools already support RDF 1.2. Triple stores such as Stardog and GraphDB combine RDF-star with native OWL/SHACL reasoning, exposed as a

queryable service over standard SPARQL HTTP endpoints. Exports are then directly consumable for use in any Python client. Lighter-weight tools such as Oxigraph also use RDF-star, but trade built-in reasoning for an embeddable, in-process design via native Python bindings. This removes the client-server hop, which matters for tight, latency-sensitive analytical loops.

- This also removes the burden of choosing a property graph over a semantic-web store as a data store medium because we have relocated it through this approach. OWL decision logic (DL) reasoning provides typed property graph (TPG)-like deterministic type validation, while SHACL shapes provide labeled property graph (LPG)-like flexible, validation-time structural checks. Both are available inside a single RDF-star store instead of forcing a choice between two separate ecosystems.
- The revised pipeline treats RDF 1.2 as the approach to represent the canonical, governance-validated semantic source of truth for the SoS architectural representation problem, and it accommodates all three SoS types (kill chains, kill webs, and continuous integrated networks) through a single representation even though the analytical abstraction generated downstream still differs by type.

Analytical Extensions:

- LPGs, Heterogeneous Information Networks (HINs), multi-partite graphs such as those used in Hetero-Functional Graph Theory (HFGT), or time-varying hypergraph structures can be generated from the RDF-star description on demand for whichever analytical or execution need arises. This preserves flexibility to any downstream graph or database structure.
- Moreover, HFGT provides a mathematical foundation well suited to differentiating a capability graph from how it would be instantiated with platforms/ defined capabilities and solutions (i.e., due to multi-functionality of many systems, these are not at all the same graph), maps cleanly onto RDF 1.2's subject-predicate-object structure. HFGT naturally separates existence (can the resource ever do a thing) from availability (condition-based, can the resource do the thing under these conditions). RDF 1.2's reified triple is structurally the same as HFGT's capability node, and RDF 1.2 triple properties (e.g., which could be mission phase, jamming state, authorization) would map onto HFGT's K_S dynamically, while its J_S could stay comparatively stable. RDF 1.2 triple properties map naturally onto an HFGT availability layer, so a multi-role node's assignment to sensing, comms, or compute is a condition on the query, not a fixed identity in a schema. This means that a SPARQL 1.2 (formerly SPARQL-star) query over an RDF 1.2 store can produce a relational binding set that's already most of the way to tensor-shaped as used for HFGT. This is advantageous due to HFGT's abilities to balance description with analysis, strong tensor-based math as needed by kill web and continuous network SoS types, and how the execution method can speed up computation.

This more integrated representation to analytics framing — RDF 1.2 for semantic foundation and governance, alongside OWL DL (open world) and SHACL (closed world) reasoning, which can be used to directly generate different mathematical abstractions as needed (HINs, multi-partite graphs, etc.) — provides the ability to:

- **Field directly into downstream physics-based simulations as a template structure that provides the correct entities and relationships already wired up.** While this will still require augmentation with physics parameters (e.g., sensor detection curves, antenna gain patterns, and weapon probability-of-kill values), this offers a lot less friction than current approaches trying to export parts of MBSE models from tools with proprietary internal formats or vendor-specific XML dialects.
- **Provides analytical foundations to support evaluation of pre-positioning of assets.** Pre-positioning of capabilities is one of the approaches used to shorten fires execution times for kill chains and kill webs. For example, loitering munitions, by being already launched and waiting for command, shorten the overall chain time by reducing the latency on the path. In graph terms, pre-positioned assets can be thought of as nodes with a certain latency (which could relate to time to get on-station) inserted at a specific capability intersection or chokepoint. This can be readily supported through the defined approach.
- **Provide a more direct ability to map closely onto and inform measures missing from current Capability Portfolio Management (CPM) practice but needed for the more complex kill web and continuous network SoS types.** For example, meta-path analyses that determine whether a new capability adds a new pathway in a kill web or improved capabilities that are impactful such as improved resolution or increased range versus duplicates an existing system capability can directly inform investment decisions: invest in the new system or scale up production of the existing one. Graph centrality measures (e.g., betweenness, eigenvectors) and min-cut/vertex-cut analysis over the same instance can help identify identifying which shared system (node) is the highest-value target for cross-portfolio investment due to its criticality or represents a single point of failure.

Cross-portfolio analysis follows naturally from the same data. Consider a kill-web path that routinely crosses sensor, communications, command-and-control, and strike portfolios whose resilience contribution is not visible in isolation. If each node carries a portfolio-owner property, identical centrality and redundancy analysis support cross-portfolio investment evaluation because they can identify which portfolios' capabilities depend on an asset owned by a different portfolio.

Notably, the level of abstraction this approach will capture maps well to needs like those above. Kill-chain-level physics fidelity does not need to be replicated in this new modeling approach for CPM decision support. These three types of warfighting SoS need network-level abstraction that trades dynamic accuracy for tractable, decision-relevant insight. The structures and approaches presented here support that and are, due to their nature, extensible and integrable with other data.

3. CONCLUSION

A military SoS Enterprise must make analytical, rationalized decisions across its portfolio of constituent systems concerning capability selection, modernization, obsolescence priorities, and resource allocation. Architectural analyses must offer actionable insight into whether decisions will improve mission outcomes and business objectives and not merely technical capability delivery.

Current EA frameworks and tools, including the Unified Architecture Framework (UAF) and The Open Group Architecture Framework (TOGAF), can describe that a military SoS exists as a network of autonomous systems, but their underlying metamodels lack semantic primitives to represent behavioral concepts such as peer-governed authority, negotiated interfaces, or context-dependent precedence that will drive overall SoS capability. As a result, an architecture built in any of these frameworks can be internally consistent and complete by the framework's own rules yet structurally misrepresent how the SoS behaves and governs itself. Since governance is the distinguishing feature of SoS types, this is a critical gap.

As highlighted in Section 2.2, we recommend shifting to an analytical use of architecture where representations are queryable, translatable to other models with the capability to evaluate and characterize graph-level properties and thereby support decision making that integrates operational performance with business decisions such as those required for Capability Portfolio Management (CPM). The approach we recommend shifts the architecting paradigm from descriptive to executable, using a decision-first framing can close the gaps in current practice and drive the generation of a multi-source, rapidly configured, graph-based architectures.

Gap/ Closure: Close the semantic gap. We recommend graph schema together with Semantic Web and Linked Data standards as a bridge to facilitate achieving as much as a unified analytical substrate as possible. Specifically, we recommend that a formal semantic ontology captured as a subject-verb-object is a necessary component of an architecture model to enable it to be computable and to link the physical and functional representations of the architecture to analytical methods. Rather than replacing current frameworks, extract what they describe well and federate their metamodels into a unified, extensible graph-based schema augmented with constructs from C2 theory, complexity science, and defense ontologies. The tools and technologies for this work exist today; the gap is a rigorous, repeatable methodology.

Gap/ Closure: Shift from descriptive to decision-centric. The computational ontological substrate should be informed and scoped by an explicit decision taxonomy that defines what data, relationships, and performance measures each critical decision class requires. This will be derived from CPM needs, needs facing the new role of Portfolio Acquisition Executives (PAEs) as described in the 2025 Acquisition Transformation Strategy and associated memoranda, and operational performance needs defined by the Combatant Commands and Joint Staff.

Further, no single tool spans the full analytical range a military SoS requires across structural alignment, functional catalogs, physics-based simulation, empirical test data, and causal reasoning. A federated, extensible architecture using Semantic Web and Linked Data standards provides the consistent machine-readable and machine-understandable foundation from which multiple mathematical abstractions can be derived. This changes the notion of architectural patterns from structural building blocks to composable analytical threads. Ensuring non-tool-locked machine-readability and machine-understandability also provides the basis for artificial intelligence (AI) technologies to be deployed throughout the entire pipeline for improved speed and validation.

Gap/ Closure: Match architectural tempo to operational tempo. Architectural currency must be treated as a design requirement and not a process problem. We recommend a structured progression: (1) develop a military SoS decision taxonomy to a usable level of specificity, (2) identify minimum viable descriptive elements for graph export from current frameworks, (3) define composable analytical patterns drawing from multiple tools and data sources, and (4) validate the approach end-to-end against two or three contrasting decision classes. AI-assisted automation is essential for maintaining currency, but only once the ontological and analytical foundations are in place.

4. PROJECT TIMELINE & TRANSITION PLAN

SERC recommends a follow-on effort to mature the concepts developed under WRT-2508 to a level that can be transitioned in partnership with industry and key Government stakeholders.

Specifically, we recommend starting with defining a decision taxonomy that specifies what data, relationships, and performance measures are required for key decisions with respect to more comprehensively understanding mission capabilities and performance trades. Specifically, this should be tailored to provide insights for portfolio decision making for kill webs and continuous state capability networks. These concepts are just as critical to supporting operational capabilities as the prevailing emphasis on kill chains. The decision taxonomy should reflect key decisions and concepts needed to support them across R&E mission engineering, needs facing the new role of Portfolio Acquisition Executives (PAEs) as described in the 2025 Acquisition Transformation Strategy, and operational performance needs defined by the Combatant Commands and Joint Staff.

Simultaneously, begin developing the methods, processes, and foundations for the semantic base to analytical execution approach illustrated in Figure 1. Start with an RDF 1.2 representation of this decision architecture together with aspects of military SoS that will orient toward mission outcomes across the capability type characteristics as highlighted in Table 1 and detailed in Table 2 and Table 4 in [Appendix C](#). This representation will include capability classes, roles, functions, attributes of systems, relationships, and constraints on which capability types can feed which others.

Ideally, this work will target a simple exemplar of each SoS capability type and develop the capabilities shown in Figure 1 to demonstrate the flow from RDF 1.2 semantic representation, to capability graph, to an instance graph that is populated from a platform library, to the ability to generate one or more types of analytical graphs that support network-level assessment and key investment decisions.

Results and recommendations from this work will: (1) refine how this work can best be transitioned to use across Services and by industry partners; (2) expand Mission Engineering analytic structures to encompass network properties and dynamics of kill webs and continuous-state networks; and (3) provide the foundations that can integrate and mature decision making across the Mission Engineering and Integration Activity (MEIA) and Capability Portfolio Management (CPM), linking portfolio decisions to mission outcomes.

APPENDIX A: LIST OF PUBLICATIONS RESULTED

Direct Technical Report Publications:

In order of creation and conceptual build:

Systems Engineering Research Center (SERC) (December 2025). Enterprise Architectures for Systems of Systems: Part 1, EA for Direct Support of Engineering and Acquisition Decision-making. Technical Report for WRT-2508, Task Order 0312, Contract No. HQ0034-24-D-0023.

Systems Engineering Research Center (SERC) (February 2026). Enterprise Architectures for Systems of Systems: Part 2, Investigation of Pain Points and Prioritization. Technical Report for WRT-2508, Task Order 0312, Contract No. HQ0034-24-D-0023.

Systems Engineering Research Center (SERC) (April 2026). Enterprise Architectures for Systems of Systems (SoS): Part 3, Challenges with Current Practice Facing the Needs of Military SoS. Technical Report for WRT-2508, Task Order 0312, Contract No. HQ0034-24-D-0023.

Related Publications:

Blackburn, MB, Clifford, M, Hagedorn, T, McDermott, TA, Sitterle, VB, Zimmerman, P (Sep 2026). Data interoperability foundations for being AI-ready. Presentation at Artificial Intelligence for Systems Engineering (AI4SE) & Systems Engineering for AI (SE4AI) Workshop, Washington DC, September 2026.

APPENDIX B: CITED AND RELATED REFERENCES

(Numerous other references are cited in the technical reports that will be provided along with this final technical report. Citations below are those referenced explicitly in this document.)

ISO. (2019). ISO/IEC/IEEE 21839: 2019. Systems and software engineering -- System of systems (SoS) considerations in life cycle stages of a system.

ISO. (2019). ISO/IEC/IEEE 21841:2019. Taxonomy of systems of systems.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering: The Journal of the International Council on Systems Engineering*, 1(4), 267-284.

Dahmann, J. S. (2015). Systems of systems characterization and types. *Systems of Systems Engineering for NATO Defence Applications (STO-EN-SCI-276)*, 1-14.

Dahmann, J. (2024, June). Current landscape of system of systems engineering. In *2024 19th Annual System of Systems Engineering Conference (SoSE)* (pp. 1-9). IEEE.

Dahmann, J. S., & Baldwin, K. J. (2008, April). Understanding the current state of US defense systems of systems and the implications for systems engineering. In *2008 2nd Annual IEEE Systems Conference* (pp. 1-7). IEEE.

Dahmann, J. and Henshaw, M. (2026). SEBoK: Architecting approaches for systems of systems. System Engineering Body of Knowledge (SEBoK), SEBoK v. 2.14, released 18 May 2026.

Klein, J., & Gagliardi, M. (2010). *A workshop on analysis and evaluation of enterprise architectures* (No. CMUSEI2010TN023). <https://apps.dtic.mil/sti/html/tr/ADA532700/>

Lapalme, J. (2012). Three schools of thought on Enterprise Architecture. *IT Professional*, 14(6), 37-43.

Mohamed Mogahed and Mo Mansouri (2026). An Ontology-Based Architecture for Interoperable Healthcare Systems-of-Systems: Structure, Interaction Patterns, and Covenant-Based Governance, *Systems*, 14: 376.

Saint-Louis, P., Morency, M. C., & Lapalme, J. (2017). Defining enterprise architecture: A systematic literature review. In *2017 IEEE 21st International Enterprise Distributed Object Computing Workshop (EDOCW)* (pp. 41–49). IEEE.

Systems Engineering Body of Knowledge (SEBoK). (SEBoK v. 2.14, released 18 May 2026). Enterprise Architecture. [https://sebokwiki.org/wiki/Enterprise_Architecture_\(glossary\)](https://sebokwiki.org/wiki/Enterprise_Architecture_(glossary))

Systems Engineering Body of Knowledge (SEBoK). (SEBoK v. 2.14, released 18 May 2026). Systems of Systems. [https://sebokwiki.org/wiki/Systems_of_Systems_\(SoS\)](https://sebokwiki.org/wiki/Systems_of_Systems_(SoS))

US Department of Defense, Office of the Under Secretary of Defense for Research and Engineering (2023). Mission Engineering Guide 2.0 (MEG 2.0). October 2023.

US Department of Defense, Office of the Under Secretary of Defense for Research and Engineering (2025). Mission Architecture Style Guide (MASG). January 2025.

Vargas, I. G., & Braga, R. T. V. (2020). Understanding system of systems management: A systematic review and key concepts. *IEEE Systems Journal*, 16(1), 510-519.

APPENDIX C: RE-VISITING THE DEFINITION OF SYSTEMS OF SYSTEMS (SoS) IN A MILITARY CONTEXT AND THE ROLE OF ARCHITECTURE FOR SoS

SoS definition and characterization build directly from Maier's original work (1998) and are now represented as described below.

SoS Definition: Set of systems or system elements that interact to provide a unique capability that none of the constituent systems can accomplish on its own. Systems elements can be necessary to facilitate the interaction of the constituent systems in the system of systems. ISO/IEC/IEEE 21839 (ISO, 2019)

SoS DEFINING CRITERIA AND CHARACTERISTICS

- i. *Operational independence of component systems* (defining criterion as per Maier)
- ii. *Managerial independence of component systems* (defining criterion as per Maier)
- iii. *Geographical distribution*
- iv. *Emergent behavior*
- v. *Evolutionary development processes*
(i.e., that developments may take place asynchronously based on the independent development processes of the constituent systems, and therefore the SoS will evolve incrementally rather than be 'delivered' as normally envisioned in a single system development or acquisition (Dahmann, 2015))

Maier emphasized the operational and managerial independence aspects ((i) and (ii)) as the defining criteria, and that a system failing either of these criteria was not an SoS regardless of how well it met the others. The IEEE definition nearly entirely emphasizes the property of emergence (iv). Dahmann (2015, 2024) has consistently captured the complete set of five criteria, which are now established across the US DoW, INCOSE, and NATO due to her active work and engaged publication history.

SoS GOVERNANCE TYPOLOGY

Maier's emphasis on operational and managerial independence as the defining criteria led to a typology of SoS, embraced by INCOSE, ISO 21839 (2019), and the DoW (Dahmann and Baldwin, 2008), that is based on governance relationships between its constituent systems:

- a) *Directed SoS* — centrally owned and managed (e.g., a single program like Terminal High Altitude Area Defense (THAAD))
- b) *Acknowledged SoS* — recognized interdependencies, shared authority, cooperative agreements between constituent systems and SoS for any system changes (e.g., Advanced Battle Management System (ABMS))

- c) *Collaborative SoS* — voluntary cooperation among independent systems, collective decision making about service (e.g., coalition Intelligence, Surveillance, and Reconnaissance (ISR) sharing)
- d) *Virtual SoS* — emergent behavior with no central management or purpose, large scale behavior emerges (e.g., *ad hoc* Command and Control (C2) in contested environments)

Importantly, This does not yet differentiate between SoS functional capability types, e.g.: those that are linearly sequenced in their behavior and relationships (kill chains), those that are sequenced but in layers and in parallel pathways in their behavior and relationships (kill webs), and those that are highly interconnected and complex but non-sequenced to achieve an objective (integrated multimodal sensing, communications, and compute). This is a separate consideration and will be addressed in the sections to follow.

SoS ARCHITECTURE AND ENTERPRISE ARCHITECTURE (EA)

SoS architecture is defined in the Systems Engineering Body of Knowledge (SEBoK) as: *“In a SoS, the architecture is the technical framework for the systems comprising the SoS which designates how the systems will be employed by the users in an operational setting (sometimes called the concept of operations (CONOPs or CONOPS), the internal and external relationships and dependencies among the constituent systems and their functions and, finally, the end-to-end functionality and data flow as well as communications among the systems.”* (Dahman and Henshaw, SEBoK, 2026)

This definition is more generalized than that of Maier, which anchors strongly on communication/ information flow interoperability as the sole focus of architecture: *“When the components of a system-of-systems are highly independent, operationally and managerially, the architecture of the system-of-systems is the interfaces. There is nothing else to architect.”*

We noted in our part 3 report that **Enterprise Architecture (EA)** has been defined in many ways. The SEBoK lists five (SEBoK v. 2.14), and literature reviews have found similar variation (Saint-Louis et al, 2017). As explained by Lapalme (2012), *“Many of these definitions are variations on the theme of “a description (and/or the process of achieving a description) of the interrelated components of an enterprise in order to guide their evolution”. These types of definitions are unclear on two key elements which make them ambiguous: (a) scope and (b) purpose.”*

We therefore adopted the view put forward by a 2010 Carnegie Mellon University (CMU) workshop on EA (Klein and Gagliardi, 2010), which extended the IEEE-1471 view of architecture to create a working definition of EA: ***“The fundamental organization of an enterprise embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.”*** The CMU workshop participants noted that *“EA is different from other architecture disciplines (software, system, and system-of-systems) in that the enterprise generally exists before any EA activity is started and must continue to exist and function as it is being changed. The evolution*

perspective included in the IEEE definition emphasizes that evolution is an essential consideration in EA in practice.” This latter point is directly aligned with the evolutionary development characteristic of SoS.

The practical distinction between system/SoS architecture and enterprise architecture becomes increasingly difficult to maintain as the boundary of the SoS expands, just as it does for any expansive and complex system. Consequently, for a military SoS, who the architecture serves and what decision it supports is not cleanly fixed by DoD guidance.

CLARIFYING PRACTICE IN MILITARY CONTEXTS: CAPABILITIES, MISSIONS, AND SoS

The Digital Engineering Body of Knowledge (DEBoK), cites DoDAF 2.02 to define **capability** as “*the ability to achieve a Desired Effect under specified [performance] standards and conditions through combinations of ways and means [activities and resources] to perform a set of activities.*”

Joint Publication 3-0, DoD Dictionary, defines **mission** as “*the essential task or tasks, together with the purpose, that clearly indicates the action to be taken and the reason for the action.*” (JP 3-0).

The Mission Engineering Guide (MEG 2.0, 2023) states that the “goal of mission engineering is to *engineer missions by identifying the right things* (i.e., technologies, systems, SoS, or processes) to achieve the intended mission outcomes and provide mission-based inputs into the systems engineering process to aid the Department in *building things right.*” The MEG 2.0 and the associated Mission Architecture Style Guide (2025) define mission threads (MTs) as sequenced, task-based capability representations (non-attributed to platforms), while Mission Engineering Threads (METs) are defined as those threads with platforms or other resources and actors assigned to each task. Mission engineering in current DoW practice directly relates capability and mission in the same process.

There are two important nuances that stem from a study of these references and concepts.

First, a capability is not limited to a sequenced, effects-based concept. A mission, however, is definitively a task-based construct, which implies sequenced relationships. An effects-based outcome such as that for fires is very much a sequenced set of tasks (a kill chain or its more complex and dynamic manifestation as a kill web).

Second, mission engineering is not limited to SoS by its own definition. The MEG 2.0 and the MASG both use the phrase “systems and systems of systems” in their general descriptions. However, every example and the over-riding emphasis is on SoS. Nearly all the worked examples, artifacts, and the entire MASG apparatus (METs, E2E views, DODAF/UAF) are built around multi-system, SoS-characterized missions. **The result is that it is easy to confuse the relationship between mission engineering and its architectures to SoS and the larger set of SoS-related decisions needed.**

KEY MILITARY SoS ARCHITECTURE USES

The following represent major uses of architecture (or EA) for SoS in a military context as discovered throughout the previous research and reports for this effort.

SoS Interoperability/Interface Reference Architecture or Reference Architecture Modules for Design Guidance.

A Reference Architecture or reference design guidance compatibility modules describe best-of-breed guidance for system design and SoS configuration. This is typically based on interoperability standards, waveforms, and protocols that make information exchange physically possible between different systems. This representation is independent of mission authority (i.e., command execution) or which mission phase is active. It provides a description of which platform types are cleared to interoperate under specified standards, serving as design guidance for SoS configuration or for a new system needing to interoperate with an existing set of systems. (This is aligned with Maier's focus.)

Descriptive Architecture for Military SoS.

Here, we directly use the definition from (Dahman and Henshaw, SEBoK, 2026): "In a SoS, the architecture is the technical framework for the systems comprising the SoS which designates how the systems will be employed by the users in an operational setting (sometimes called the concept of operations (CONOPs or CONOPS), the internal and external relationships and dependencies among the constituent systems and their functions and, finally, the end-to-end functionality and data flow as well as communications among the systems."

This is descriptive. Both capability architectures (capabilities and their relationships in context) and mission architectures (with platforms/systems, resources, and actors in the context of a mission) are typically descriptive in practice.

Computational Analytical/ Decision Support Architecture.

An architecture for direct use in analytical decision making needs to surface the relationships and dependencies between different dimensions of the meta-system in question. It needs to support analytical queries to determine characteristics like which nodes are structurally load-bearing and what impact a candidate system addition/removal/upgrade has on network performance and resilience. An analytical, decision-centric architecture is directly queryable, supports execution of algorithms to assess impacts of choices made in one dimension against others and time, etc. The shift from a descriptive to decision-centric architecture is achieved by making the architecture model formal, computable, and able to integrate with other data, other data models and computational ontologies, and other analysis and decision support tools. This integration is key.

At some level in the decision-making process, performance in an operational context must be weighed against cost, schedule, and other business considerations. Therefore, an architecture to support decision making needs to be extensible and readily combinable with other types of data for integrated assessment. Specifically, this needs to support

computational analyses, so the architecture (its elements and relationships) needs to be machine readable and machine understandable by various systems and not locked into a single framework or tool.

The line blurs between descriptive architectures and analytical architectures. The nature of architecture description – its substrate or modeling construct – changes what one can do with it. UAF in the current mission engineering process, for example, is a static picture of a thing represented in an architecture model. It is useful to query in the tool for relationships and gaps across the UAF views or to export certain sets of conditions, elements, and sequences as inputs for a physics-based simulation such as AFSIM. However, the distinguishing feature is that the descriptive architecture is a static representation and that network-level properties and what-if types of dynamic queries require a different modeling approach.

What happened to governance as per the SoS governance typology? One of the few papers to discuss SoS based on Maier's governance types is the review by Vargas and Braga (2020). The authors define SoS governance as *"as a multidimensional concept related to a more strategic vision, containing regulations, policies, requirements, best practices, and decision-making criteria to guide the SoS toward the achievement of its objectives."* They distinguish this from SoS management, which they describe as covering *"business/scoping activities, such as governance, stakeholder management from a political perspective, contract negotiations with organizations responsible for some of the CSs <constituent systems> in directed SoS (e.g., international SoS, or across company boundaries), necessary assessments from a users' perspective, overall funding, and analyses of alternatives."* The authors note that SoS in the spectrum of SoS types and characteristics must be understood prior to the development of a management process, since they will affect the way SoS and its constituent systems are managed. However, architecting, and the role of architecture in achieving SoS governance was not fleshed out. The authors briefly referenced business process architecture as a tool for SoS management, and listed architecture development and enhancement as activities found in SoS engineering management (referring to the management of associated engineering activities to support SoS capabilities) rather than SoS management.

The MEG 2.0, MASG, and the MASG UAF-oriented material organize entirely around missions, capabilities, and threads. They do not represent or discuss the four SoS governance types; MASG's only reference to governance is about the governance of the architecture artifacts and architecting process. The SEBoK article on *Architecting Principles of SoS* (Dahmann and Henshaw, 2026) does tie governance type to *which of Maier's four architecting principles apply* (e.g., stable-intermediate-forms and the Wave Model fit directed and acknowledged types but not the other two). Even so, this is used more as analytic commentary about applicability. **There is no governance typology based active design framework with its own artifacts present across the military problem space and its guiding documents.**

KEY EFFECTS-BASED FUNCTIONAL TYPES OF MILITARY SOS

In JP 3-0, the DoD Dictionary, **effect** is defined as “1. *The physical or behavioral state of a system that results from an action, a set of actions, or another effect.* 2. *The result, outcome, or consequence of an action.* 3. *A change to a condition, behavior, or degree of freedom* (JP 3-0).”

Table 2 describes three major functional types of interconnected system configurations to achieve effects-based capability in warfighting contexts.

Table 2. Three primary effects-based configurations of systems for warfighting contexts

Key Characteristics	Primary Effects-Based System Configurations		
	<i>Systems performing discrete, sequential operational effects</i>	<i>Systems performing dynamic, multi-path operational effects</i>	<i>Systems performing continuous, foundational state-based effects</i>
Structure/ Characterization	Linear sequence: Linear, rigid, and time-critical execution paths where a single node failure breaks the sequence. Success is measured by speed and throughput.	Layered, multipath matrix: Layered, combinatorial, and opportunistic mission structures where inputs and outputs are dynamically routed through surviving network nodes. Success is measured by mission resilience and adaptability.	Non-sequenced, ubiquitous grid: Non-sequenced, ubiquitous, and steady-state information environments providing persistent theater transparency and freedom of action. Success is measured by <i>coverage, latency, and system uptime</i>.
Example	Kill chain	Kill web	Integrated sensing/ communications/ compute network
Primary capability effect type	Discrete, kinetic	Adaptive, multi-path, likely multi-domain kinetic	Foundational, state-based
Operational logic	Driven by a specific end-state (e.g., target neutralized)	Driven by a specific end-state (e.g., target neutralized)	Driven by maintaining a continuous, optimized operational condition or set of conditions (e.g., persistent, area dominated, and network resilient)
Temporal nature	Have a distinct beginning, middle, and end to achieving envisioned capability	Have a distinct beginning, middle, and end to achieving envisioned capability	Run indefinitely in a steady state; persistent and continuous over some defined time period
Task relationship	Tasks follow a logical, linearly sequential dependency	Tasks follow a logical dependency, sequenced in layers, adaptive across multiple paths that can be executed in parallel or asynchronously	Tasks happen concurrently with little to no serialization across nodes
Primary metrics of success	Time-to-target, effect achieved	Resilience, effect achieved	Coverage, latency, and uptime

Expanding from Table 2, a **kill chain** is defined as *a sequence of actions and flow of information from detection through target engagement and post-attack assessment* [JP 3-09]. As a conceptual model, a kill chain is a sequential, step-by-step process to identify and engage a target. It is serial in nature, with each phase tightly coupled to the steps immediately prior and following in the sequence; effectiveness relies on every phase being achieved. Though not the only model, Find, Fix, Track, Target, Engage, Assess (F2T2EA) is a common framework and the one used most prevalently in mission engineering.

A **kill web** is defined as *multiple, interconnecting kill chains, specifically forming a network map, interconnected for resiliency. Kill webs ensure multiple avenues for each step of the kill chain, ensuring the disruption or destruction of one node does not prevent the joint force from engaging.* [JP 3-09⁴]. Design aspirations for a kill web focus on rapid adaptation at speed to execute functions across platforms (and typically also domains) in parallel. The goals are resilience through adaptability to changing conditions and increased speed that come from being able to reconfigure network pathways during an engagement. When each key system element can relate directly to numerous other system elements, these dynamics can occur to complete the sequence across multiple pathways in parallel. Kill webs can still decompose to kill chains when defining a mission and workflows; it is the information and resource sharing and pre-positioning of capabilities that must be designed into the web via shared nodes in the web's graph structure. Kill web capability emerges from the network and therefore requires network-level analyses. As military concepts of operations shift from kill chains to kill webs, the analytical focus similarly shifts from optimizing a linear chain of engagements to orchestrating a dynamic network of distributed capabilities.

Continuous, state-based effects are achieved through **highly interconnected networks of distributed systems**. This is commonly realized by integrated sensing, communications, and/or compute networks, e.g., for area surveillance. Each node in the overall network may exhibit variation within its type (e.g., sensors of multiple modalities within a network). Nodes may also be multi-functional, where some may sense, communicate, compute, decide, and/or execute (i.e., act on a decision whether as an aspect of selection such as prioritizing a different sensing capability, location of compute in the network, etc.). A single system may also perform multiple of those functions, such as integrated electromagnetic sensing and communications. The resultant interconnected system is adaptive, non-linear, and its elements, behavior, functional capabilities, requirements, system boundaries, etc. may all vary in time).

ARE THESE EFFECTS-BASED FUNCTIONAL SYSTEM CONFIGURATIONS CONSIDERED SoS BY THE CRITERIA?

⁴ Joint Chiefs of Staff (24 July 2025). Joint Publication (JP) 3-09, Joint Fires.

Table 3 shows the characterization of each of these effects-based system configuration types according to the five SoS criteria.

Table 3. SoS characterization of the primary effects-based system configurations

SoS Criterion	Primary Effects-Based System Configurations		
	<i>Systems performing discrete, sequential operational effects (kill chain)</i>	<i>Systems performing dynamic, multi-path operational effects (kill web)</i>	<i>Systems performing continuous, foundational state-based effects (distributed, interconnected network)</i>
Operational independence	Yes. Each sensor, C2, shooter, etc fulfill their own purpose outside the chain	Yes. Same as for kill chain, just with more constituent systems	Yes, Each sensor/comms/ compute node serves its own purpose outside the surveillance aggregate
Managerial independence	Yes, if genuinely separately acquired/run (service- or program-specific)	Yes, typically even more diffuse across services	Yes, typically across services/programs (closest of the three to Maier's Internet/WAN example)
Geographic distribution	Usually, but not required	Usually	Usually
Emergent behavior	Yes. Engagement closure emerges only from the chain and not a single system in the chain	Yes, Same reasoning as kill chain but more pronounced. Adaptivity and multipath capabilities provide emergence absent in any one chain	Yes. A surveillance picture, for example, emerges from a combination of nodes and capabilities (sensing, comms, compute) not realizable by a single node even if it has all 3 functions (i.e., a single node cannot triangulate).
Evolutionary development	Yes	Yes	Yes
SoS by definition?	Yes. Operational and managerial independence in most all cases in practice. Also exhibits the other criteria.	Yes. Same reasoning.	Yes. Same reasoning.

From these characterizations, all three system configuration types can be SoS under the two principal criteria of operational and managerial independence; nothing about being sequenced, multi-path, or continuous and state-based changes whether each example qualifies as an SoS. It is the internal structure and dynamics of the SoS that changes across these functional capability types.

APPENDIX D: EFFECTS-BASED FUNCTIONAL TYPES OF MILITARY SoS IN THE CONTEXT OF ARCHITECTURE USE

To support OUSD(R&E)'s Joint Architecture for Capabilities and Systems (JACS)⁵ priorities, SoS and enterprise architectures must capture the unique internal structures and dynamics of each functional capability type to the depth required for their use. This structural depth is essential to analyze, govern, and achieve seamless integration and interoperability across the entire enterprise portfolio, and the methods to capture these SoS will not be one-size fits all. In Table 4, these SoS types are evaluated in the context of three main architectural uses.

Table 4. SoS functional capability type versus architecture use matrix

Key Characteristics	Primary Effects-Based System Configurations		
	<i>Systems performing discrete, sequential operational effects (kill chain)</i>	<i>Systems performing dynamic, multi-path operational effects (kill web)</i>	<i>Systems performing continuous, foundational state-based effects (distributed, interconnected network)</i>
Structure/ Characterization	Linear sequence: Linear, rigid, and time-critical execution paths where a single node failure breaks the sequence. Success is measured by speed and throughput.	Layered, multipath matrix: Layered, combinatorial, and opportunistic mission structures where inputs and outputs are dynamically routed through alternate network nodes. Success is measured by mission resilience and adaptability.	Non-sequenced, ubiquitous grid: Non-sequenced, ubiquitous, and steady-state information environments providing persistent theater transparency and freedom of action. Success is measured by <i>coverage, latency, and system uptime</i> .
Example	Fires kill chain to produce a kinetic effect on a target	Joint fires kill web to achieve synergistic effects and multipath resilience to produce effect on a target	Integrated sensing/ communications/ compute network
Primary capability effect type	Discrete, kinetic	Adaptive, multi-path, likely multi-domain kinetic	Foundational, state-based
Reference Architecture: At SoS level, best-of-breed guidance for system design and SoS configuration; typically based on interoperability standards, waveforms, and protocols that make information exchange possible between different systems	- Fixed compatibility matrix per platform-pair; waveform/ protocol standards scoped to a known, set of chain participants. - Interface standards for deterministic transitions; messaging profiles matching specific sequence phases. - Model federation of best -of-breed reference	- Larger combinatorial interface surface; interoperability standards must support many alternate path combinations; reconfigurability becomes a design requirement. - Dynamically discovered interfaces require specifying agnostic data brokers and pub/sub protocols, polymorphic data schemas for varying	- Standards shift from who may talk to whom to universal data/ exchange models (any-sensor-to-any-node); no privileged path, so closest to Maier's "architecture is the interfaces" case. - Interfaces defined as continuous data transport layers. Specification requires describing patterns of distributed mesh and ad hoc routing

⁵ <https://www.cto.mil/sea/jacs/>

	designs / standards as modules for connectivity in MBSE form are widely used and successful in practice. These can be kept in “libraries” and composed.	payload types, and looser coupling to allow ad-hoc registration of elements. • Same model federation of best -of-breed reference designs / standards apply here.	protocols, patterns of connectivity possible via different waveforms, and how both result in patterns for automated data and execution prioritization. • Dynamics of structure cannot break function, so the aggregate of models needs to reflect tested SoS stability to be a reference.
Descriptive Architecture: At SoS level, the structure, behaviors, and global rules for an ensemble of nodes, systems, elements, or other resources that collaborate to fulfill the functions of an overall objective	Deterministic, static mapping of defined and largely hierarchical sequences: • Capability description via functions and sequenced tasks required to achieve effect • Explicit allocation matrices binding one platform (e.g., F-35) to one performer function (e.g., a target) within a specific mission engineering thread. • Linear, sequential timeline diagrams depicting static step transitions; Sequential, time-ordered behavioral workflows; Binary global rules governing step-by-step triggers.	Polymorphic, dynamic mapping of multiple possible, heterarchical, and layered sequences: • Many-to-many (M:N) mapping where a single platform may hold multiple roles simultaneously or swap roles mid-mission. • Dynamic, flat, or mesh network topologies; Represent branching/ adaptive/parallel pathways to achieve effect. • Event-driven, non-linear conditional execution paths; mapping dynamic tasking reassignment rules, heterarchical relationships based on node attrition or target changes.	Continuous, dynamic and adaptive infrastructure-as-a-platform characterization: • Ubiquitous mapping where nodes do not map to episodic tasks but continuously feed a baseline infrastructure layer. • Spatially continuous, high-density asset grids; spatial and geographical descriptions for resource density, sensor coverage, and computational edge clusters; range of/ adaptive modalities and associated resolution and quality/nature/ temporal aspects of network exchange and compute pathways. • Concurrent, always-on resource behaviors combined with event-driven/ conditions-based, mapping of dynamic tasking and assignments.
Analytical Architecture: At SoS level, directly queryable, supports execution of algorithms to assess impacts of choices made in one dimension against others and time, etc.; use to support analytical queries, determining which nodes are load-bearing, impact of node addition/removal/upgrade has on network performance and resilience; combinable with other data for enhanced analytics.	Linear/ aggregate dependency analysis, interoperability, and optimization: • Structural gap analysis focused on a single-point failure breaking the asset chain, inclusive of resource and performer mapping to tasks. • Path analysis to identify critical links/single failures. • Simple matrix analysis for end-to-end timelines, impact calculations based on structural severing of a branch.	Graph theoretic optimization, dynamic dependencies, and resiliency: • Gap analysis that models structural redundancies and alternative routing availability; combinatorial analysis of matrixed resource options and connectivity. • Identification of load-bearing nodes and paths, where what is load bearing can change along with the dynamics of	Continuous field analytics: • Gap analysis that highlights coverage holes (modality, resolution, spatial), bandwidth bottlenecks, latency characteristics, viable reassignment and rerouting capabilities, and data management and processing constraints. • Network capacity, signal latency, and uptime analysis under different conditions; Throughput resilience/ degradation

<i>(Here, this is described for behavioral, or operational performance, characteristics. Business needs linked to these analytical needs will be addressed separately.)</i>	• Multiple chains may be modeled and assessed for optimality, probabilistic success, or substitution in the aggregate.	shifting C2 nodes and pathway options. • Data cubes and Tensor-based analysis methods. • Network-level properties such as resilience, dynamic flow properties, behavioral time constants, node criticality, set of realizable multipaths, impacts to overall performance based on node addition/ removal, redundancies, etc.	based on spatial attrition or interference. • Identification of load-bearing nodes how the overall network structure and therefore functional capability can change with dynamics of conditions/ network nodes; System-wide health, state-maintenance, and load rules. • Network-level properties such as resilience, dynamic flow properties, behavioral time constants, node criticality, set of realizable multipaths, impacts to overall performance based on node addition/ removal, redundancies, etc.
--	--	--	--

As you move from left to right across the columns (SoS functional capability types), the architecture shifts from modeling a static capability representation in a strictly sequenced process (kill chain, even if a set thereof), to modeling an adaptive, highly interconnected set of resources and capabilities that require a network-level understanding (kill web), to modeling a more continuous and dynamic capability (continuous, distributed, interconnected network). The implications of these differences are become significant at the analytical architecture level. The EA and SoS analytic needs are not “or,” they are “and”; complex systems decompose from the right side of the table to the left. Analysis methods are needed for all three types.

For two of the three primary functional capability SoS configurations relevant to warfighting (kill webs and continuous networks), capability emerges from the network (these are systems performing dynamic, multi-path operational effects (kill webs) and Systems performing continuous, foundational state-based effects (continuous, distributed, interconnected networks)). These SoS types therefore require network-level analyses. As military concepts of operations shift from kill chains to kill webs, the analytical focus similarly shifts from optimizing a linear chain of engagements to orchestrating a dynamic network of distributed capabilities.

APPENDIX E: PILOT SoS DESCRIPTION WITH CONDITION-BASED HETERARCHICAL RELATIONSHIP

OVERVIEW

In this work, we built on the prior SERC whitepaper that identified gaps facing enterprise architecture (EA) for military Systems of Systems (SoS) and provided recommendations to close those gaps⁶. Specifically, we scoped the work to focus on the heterarchical aspects of the SoS problem and how they might be represented. This work takes a single, tractable slice of UAF (three views from the Operational viewpoint) and demonstrates that (a) these views can be extracted as a formal OWL ontology, and (b) that ontology can be extended with new semantic primitives that represent heterarchical governance relationships absent from UAF's native metamodel. The approach used is entirely open source.

STARTING POINT: 3 UAF OPERATIONAL VIEWS

Three UAF Operational viewpoint specifications were selected from the UAF 1.2 Domain Metamodel (DMM)⁷ as the minimum viable scope as listed in Table 5.

Table 5. Three operational views from UAF selected for pilot

View	Abbreviation	Purpose in UAF
Operational Taxonomy	Op-Tx	Defines the root class hierarchy of operational concepts: elements, agents, performers, architectures
Operational Structure	Op-Sr	Defines how performers are organized: roles, ports, capabilities, locations, and compositional relationships
Operational Connectivity	Op-Cn	Defines what flows between performers: exchanges, connectors, interfaces, and exchange items

Together these three views provide the vocabulary for who performs, how they are structured, and what they exchange. This was determined to be the minimum needed to represent a functioning operational network. Op-Tx supplies 5 classes, Op-Sr supplies 4 classes and 5 properties, and Op-Cn supplies 6 classes and 7 properties. All properties have explicit `rdfs:domain` and `rdfs:range` declarations.

In ontology terms:

⁶ Systems Engineering Research Center (SERC) (April 2026). WRT-2508 Enterprise Architectures for Systems of Systems (SoS): Challenges with Current Practice Facing the Needs of Military SoS. Technical Report for Task Order 0312, Contract No. HQ0034-24-D-0023.

⁷ <https://www.omg.org/spec/UAF/1.2/DMM/PDF>

- A Class is a *thing*, a noun that can represent a physical object, category, or a concept.
- A Property is a characteristic, attribute, or *relationship between things*, a verb that connects one entity (a domain) to another entity (its range). In ontology languages like OWL, a domain restricts which class a property can originate from, while a range restricts what class the target entity must be

UAF IMPLEMENTATION GAPS AND TOOL-BASED ADDITIONS

The UAF 1.2 DMM Op-Tx view includes a class named *ArbitraryConnector* (UAF DMM Figure 9-183) as shown in Figure 2. The DMM defines it as *"a visual indication of a connection used in high-level operational concept diagrams."* It is a typed structural element with from and to roles both pointing to *ConceptItem*. It is not a property or relationship type. It carries no semantic qualification.

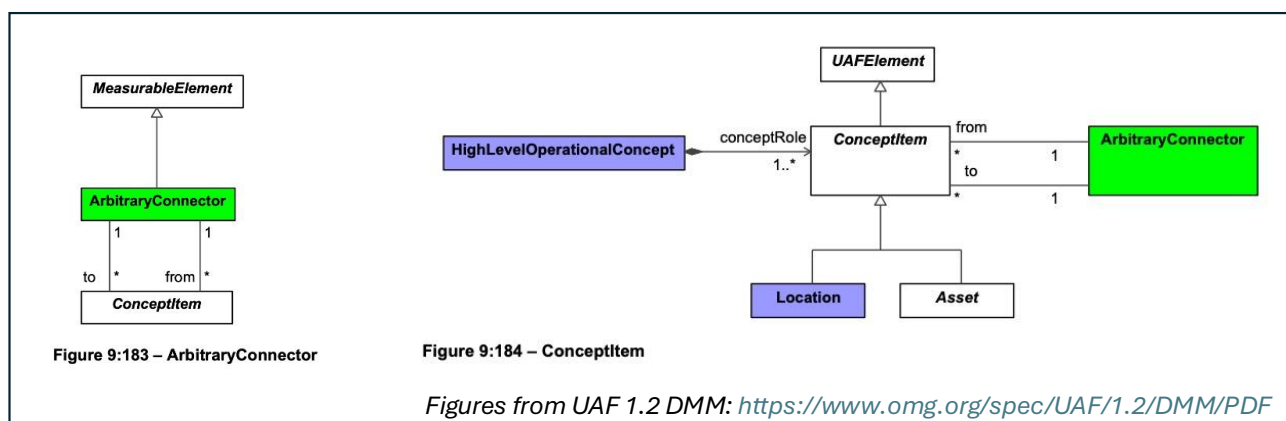


Figure 2. ArbitraryConnector class from UAF 1.2 Domain Metamodel (DMM)

(The 1's indicate that a single instance of an element is involved in the relationship on that "to" or "from" end. In standard Unified Modeling Language (UML) and Systems Modeling Language (SysML) notation used across the UAF 1.2 Specification, these numbers define exactly how many instances can participate in an association.)

Commercial MBSE tools (e.g., Cameo Systems Modeler / No Magic) render this element as "Arbitrary Relationship" or "Operational Association" in their tool-layer representations:

- *operationalAssociation* Under Op-Tx: "Operational Association: An operational association describes a structural relationship between operational performers. An operational role can provide the means to pass operational exchanges between operational performers."⁸

⁸ No Magic / 3DS. Cameo UAF/Quick Reference Guide (November 2016).

<https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/CATIA/NoMagic/pdf/enterprise-architecture-uaf-quick-reference-guide.pdf>

- *ArbitraryRelationship*. And separately, under Op-Sr: "Arbitrary Relationship: The simplest indication that there is some kind of relationship that must be detailed in the architecture." ^{ibid}

This is tool-defined paraphrasing, not an extension of UAF. The elements are normative UAF, building from the *ArbitraryConnector*, for which semantic emptiness is intentional in the UAF DMM. That this relationship concept is not well-defined and exists more as a loose placeholder in UAF (which is why the tool vendors moved to add some specificity) speaks to the semantic gap pointed out in the SERC whitepaper concerning military SoS.

The notional extensions to an ontology for military SoS representable as a graph schema discussed in the next section begin to address this gap directly.

SoS-SPECIFIC EXTENSIONS AND THE NEED FOR REIFICATION

SoS-Specific Extensions

Four new classes and five new object properties (designated with the prefix “sos:”) were added to represent heterarchical governance relationships absent from UAF's native metamodel. These are listed in Table 6 and Table 7, respectively.

Table 6. New classes required for pilot SoS example

Class	Semantic Role
sos:OperationalAssociationInstance	Reified, typed peer relationship between two performers
sos:AuthorityType	Controlled vocabulary: Formal, Negotiated, Emergent
sos:AuthorityShift	Event in which de facto coordination authority migrates between performers
sos:MissionContext	Operational condition or phase that activates an authority shift

Table 7. New properties required for pilot SoS example

Property	Domain → Range	Semantic Role
sos:coordinationAuthorityShiftsTo	AuthorityShift → Performer	Authority migration target
sos:authorityShiftFrom	AuthorityShift → Performer	Authority migration source
sos:underMissionContext	AuthorityShift → MissionContext	Condition binding
sos:negotiatedInterface	Performer → Performer	Lateral interface by agreement
sos:participatesInSoS	Performer → OperationalArchitecture	Multi-SoS membership

Reification

In OWL, an object property (predicate) is an arc, meaning it is represented as a directed edge between two nodes. An arc cannot itself be the subject of another arc. More precisely, in RDF and OWL, a named arc between two nodes (an owl:ObjectProperty assertion) cannot itself be the subject of further triples. Given a triple of the form:

uaf:ISR_Platform -> uaf:connectedTo -> uaf:StrikeCell

it is syntactically impossible to add a qualifier such as "this connection is governed by agreement" or "authority on this arc shifts to EW Node under condition X" without promoting the arc to a node.

This promotion is called **reification**. A new element, `sos:OperationalAssociationInstance`, is introduced as a named resource whose properties describe the original relationship. The two participant performers are then attached to this node via `sos:associationParticipant` properties.

This is the standard OWL n-ary relation pattern. It is required because the UAF DMM's ArbitraryConnector provides only a visual arc with no semantic content. The reification pattern converts that arc into a typed, queryable resource.

Two concrete cases illustrate why promotion from arc to node is necessary in an SoS ontology.

Example 1, Qualifying a Peer Relationship. Consider that we need to express that Performer A (ISR Platform) and Performer B (Strike Cell) have a connection that is negotiated in character.

OWL will allow: *uaf:ISR_Platform sos:negotiatedInterface uaf:Strike_Cell*.

This triple asserts that some negotiated interface exists between the two performers. But OWL treats `sos:negotiatedInterface` as a bare arc, a directed edge with no identity of its own. Therefore, there is no way to attribute this relationship as a property with additional discriminators such as authority type, a trust level, or any conditioning information to that edge.

If we attempt to write: *sos:negotiatedInterface sos:hasAuthorityType sos:Negotiated*, this is illegal in OWL; a property assertion cannot be the subject of another triple.

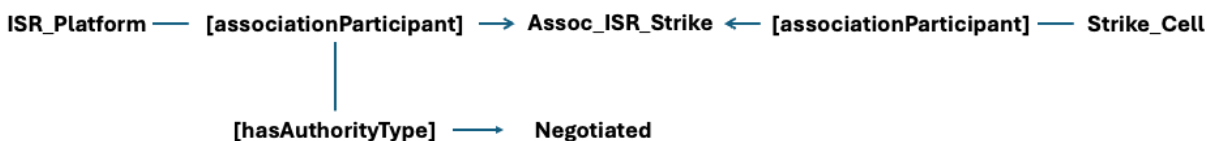
So, we cannot structure this as shown below:



Instead, we need to create a new relationship as a *named node*, e.g., `sos:Assoc_ISR_Strike` as a type of `sos:OperationalAssociationInstance`. Because it is a node, it can be the subject of any number of additional triples. This would look more like the following:

```
sos:Assoc_ISR_Strike a sos:OperationalAssociationInstance ;
  sos:associationParticipant uaf:ISR_Platform ;
  sos:associationParticipant uaf:Strike_Cell ;
  sos:hasAuthorityType      sos:Negotiated .
```

And, this structure would look as follows:



Also, a SPARQL query can now retrieve every negotiated association in the graph by pattern-matching on `sos:hasAuthorityType sos:Negotiated`.

Example 2. Conditioning an Authority Shift on Mission Context. Consider that we need to express that a theater commander currently holds coordination authority over the entity EW Node, but under the condition EW Suppression Phase that authority shifts to the EW Node itself (i.e., a degraded communications situation).

OWL will allow: `uaf:TheaterCommander sos:negotiatedInterface uaf:EW_Node`.

There is no UAF-native property for authority, so this is the closest bare arc available. Even if we coined a custom property like `sos:holdsAuthorityOver`, it would still be an unqualified arc because OWL has no mechanism to assert that a property holds only under a condition (i.e., properties cannot carry conditional qualifiers). For example, we could not write:

```
uaf:TheaterCommander sos:holdsAuthorityOver uaf:EW_Node
```

```
IF sos:context = sos:EWSuppressionPhase
```

Instead, we create `sos:AuthShift_TC_to_EW` as a first-class node of type `sos:AuthorityShift`. Because it is a node, we can attach properties to it. Here, these are source performer, target performer, and activating context, all independent triples on the same node.

```
sos:AuthShift_TC_to_EW a sos:AuthorityShift ;
  sos:authorityShiftFrom      uaf:TheaterCommander ;
  sos:coordinationAuthorityShiftsTo uaf:EW_Node ;
  sos:underMissionContext     sos:EWSuppressionPhase .
```

```
sos:EWSuppressionPhase a sos:MissionContext .
```

The sos:underMissionContext triple is only expressible because the shift is a node, not an arc. The second SPARQL query later in this report directly queries this; it pattern-matches on the sos:AuthorityShift and then joins on the three outgoing properties. Without reification, that query would have nothing to bind to and would return zero results. (A query using only native UAF concepts would also return zero results, because these concepts are not present in the UAF DMM.)

EXTENDED ONTOLOGICAL REPRESENTATION

The extended ontology is split across two @prefix namespaces to maintain a clean boundary between normative UAF content and novel SoS extension content. (This type of separation allows the UAF base graph to be updated independently as the DMM evolves in the future without requiring changes to the SoS extension layer.)

The resulting extended ontology contains 28 classes and 32 properties (31 object properties and 1 data type property (from the uaf: trustLevel present in Op-Cn). All uaf: classes and properties are derived directly from the Op-Tx, Op-Sr, and Op-Cn element lists in the DMM. The 4 sos: classes and 5 sos: properties described above are added.

The combined classes and object properties described above were serialized as an OWL ontology in Turtle (.ttl) syntax and loaded into Protégé 5.6.9 (Stanford University, Apache License 2.0), shown in Figure 3. The Protégé HerMiT reasoner confirms no logical inconsistencies in the combined graph.

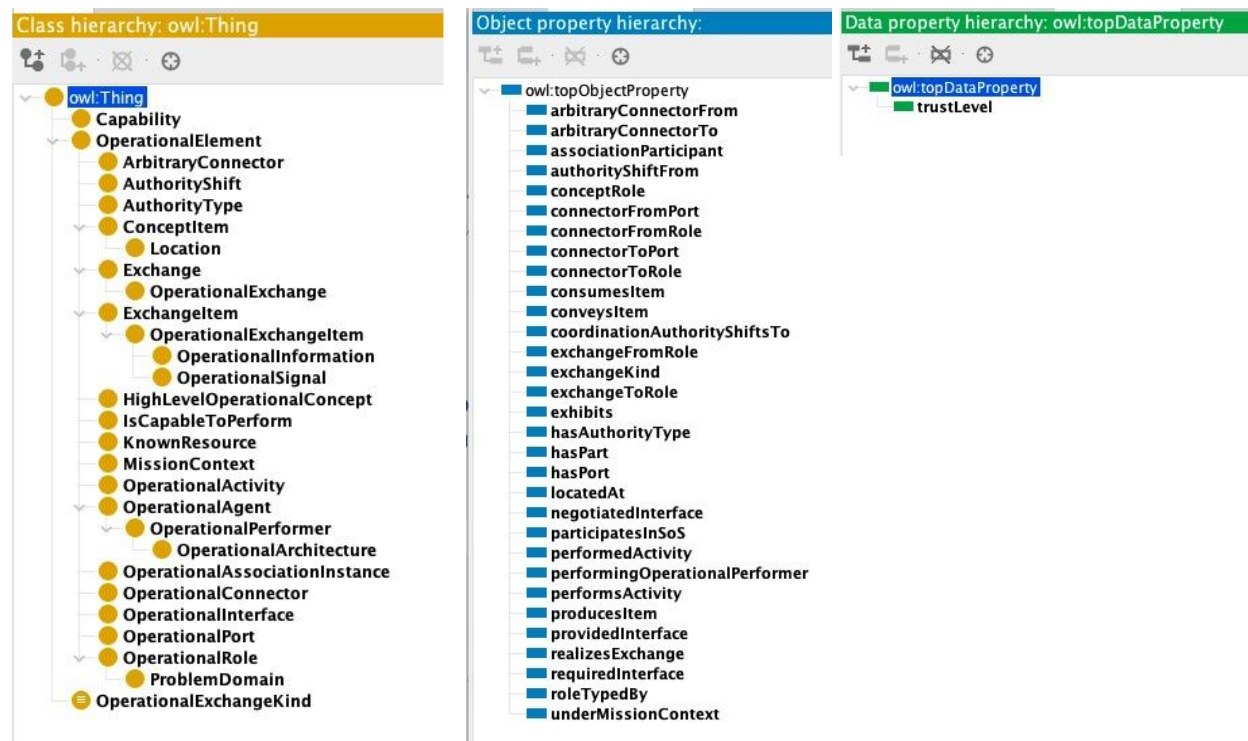


Figure 3. SoS pilot ontology shown in Protégé

The complete list of classes and properties for the extended ontological representation were provided in an Appendix to the A008 Bi-monthly Status Report for April-May 2026.

SIMPLE NOTIONAL SCENARIO

A minimal four-performer scenario was constructed to instantiate the ontology and verify schema conformance as listed in Table 8.

Table 8. Performers and Basic Scenario

Individual	UAF Class	Role in Scenario
ISR Platform	OperationalPerformer	Produces target coordinate data; participates in two SoS simultaneously
Strike Cell	OperationalPerformer	Receives target data via negotiated interface
Theater Commander	OperationalPerformer	Holds formal coordination authority
EW Node	OperationalPerformer	Acquires de facto coordination authority during EW Suppression Phase

This scenario is illustrated in Figure 4 and consists of the following activities based on UAF-native exchange (Op-Cn):

- ISR Platform exposes Port_ISR_Out; Strike Cell exposes Port_Strike_In.
- A Connector_ISR_Strike links these ports and realizes Exchange_TargetCoords, which conveys an OperationalInformation item ("Target Coordinates") with exchangeKind: InformationExchange.
- Exchange from and to endpoints are OperationalRole instances per UAF DMM Op-Cn specification.

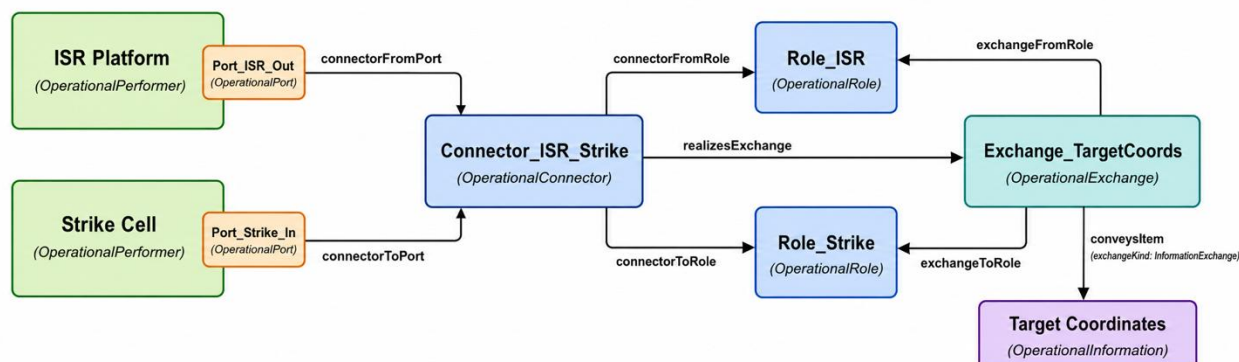


Figure 4. Notial scenario for pilot investigation

Gaps the Scenario Addresses

The scenario represents a four-performer joint operational construct in which three structural gaps absent from UAF-native representation are explicitly modeled.

SoS gap 1 — Multi-SoS participation: ISR Platform asserts `sos:participatesInSoS` to both SoS_Alpha and SoS_Bravo. This S5-type shared-resource pattern (Maier, 1998) is inexpressible in any UAF structural decomposition.

SoS gap 2 — Context-conditioned authority shift: `sos:AuthShift_TC_to_EW` asserts that under `sos:EWSuppressionPhase` (a `MissionContext`), de facto coordination authority migrates from Theater Commander to EW Node. This is illustrated in Figure 5. The formal command hierarchy is not altered; the shift is purely operational and context-conditioned. This relationship cannot be expressed in UAF because the DMM provides no class for mission-context-conditioned authority.

SoS gap 3 — Reified, typed peer relationship: `sos:Assoc_ISR_Strike` reifies the ISR Platform ↔ Strike Cell connection as an `OperationalAssociationInstance` with `AuthorityType: Negotiated`. This converts the UAF ArbitraryConnector visual arc into a semantically typed, SPARQL-queryable node.

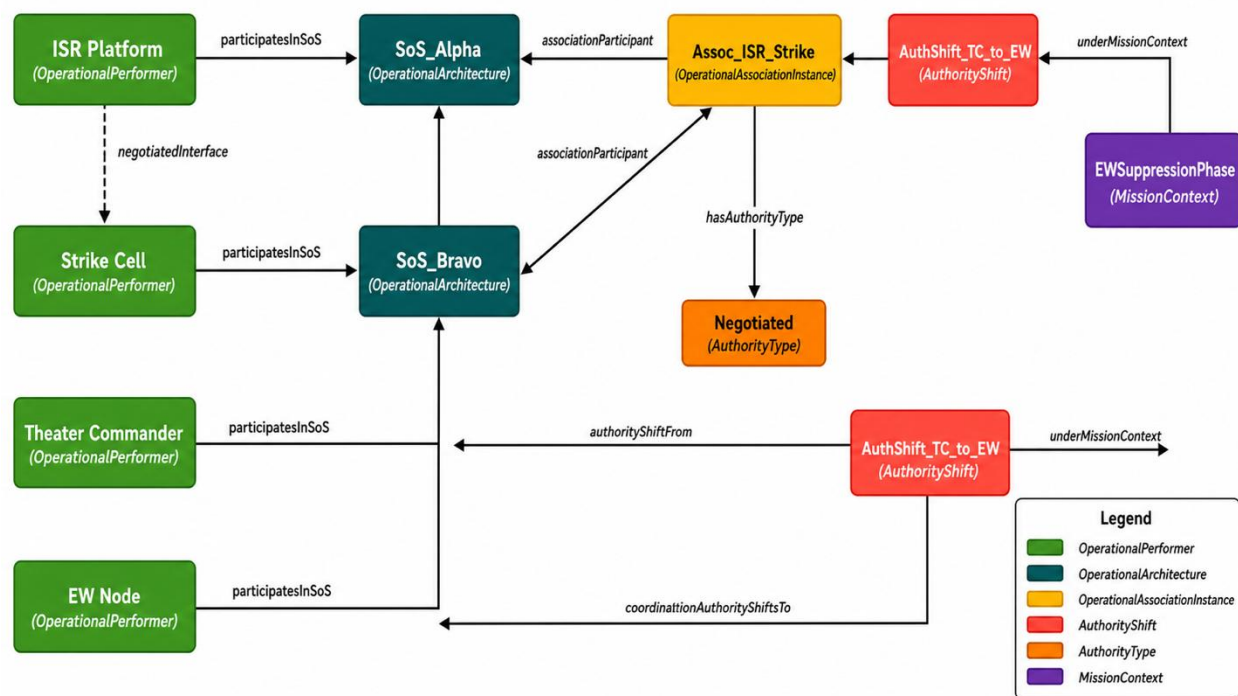


Figure 5. Illustration of scenario structure showing condition-based authority shift

Protégé Reasoner Check

The scenario .ttl file was loaded into Protégé alongside the extended ontology .ttl and the built-in Hermit 1.4 OWL reasoner did not find any errors related to unsatisfiable classes or assertions not-consistent with domain/range constraints. Further:

- sos:EWSuppressionPhase is correctly inferred as a sos:MissionContext
- sos:AuthShift_TC_to_EW is correctly typed as sos:AuthorityShift

SPARQL Queries for Initial Verification

Two SPARQL queries (performed using <https://www.semantechs.co.uk/turtle-editor-viewer/>) extend the demonstration of how this extended schema supports military SoS considerations.

Query 1: UAF-native (Op-Cn). Retrieve all information exchanges between operational roles.

```
PREFIX uaf: <https://example.org/uaf-operational-ontology#>
SELECT ?exchange ?fromRole ?toRole ?item WHERE {
  ?exchange a uaf:OperationalExchange ;
    uaf:exchangeFromRole ?fromRole ;
    uaf:exchangeToRole ?toRole ;
    uaf:conveysItem ?item .
}
```

This correctly returns one result.

Exchange_TargetCoords from Role_ISR to Role_Strike conveying Target Coordinates.

This is the class of query answerable via UAF tooling.

Query 2: SoS extension only. Identify authority shifts and mission contexts that activate them.

```
PREFIX uaf: <https://example.org/uaf-operational-ontology#>
PREFIX sos: <https://example.org/sos-heterarchy-extension#>
SELECT ?shift ?fromPerformer ?toPerformer ?context WHERE {
  ?shift a sos:AuthorityShift ;
    sos:authorityShiftFrom ?fromPerformer ;
    sos:coordinationAuthorityShiftsTo ?toPerformer ;
    sos:underMissionContext ?context .
}
```

This correctly returns one result.

AuthShift_TC_to_EW from Theater Commander to EW Node under EW Suppression Phase.

This query would return no results using a standard UAF tool because there are no native UAF classes and associated properties to support it.