System of Systems Engineering Guidebook



September 2026

Office of the Under Secretary of War for Research and Engineering

Office of the Assistant Secretary of War for Mission Capabilities

Washington, D.C.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

The appearance of external hyperlinks does not constitute endorsement by the United States Department of War (DoW) of the linked websites or the information, products, or services contained therein. The DoW does not exercise any editorial, security, or other control over the information you may find at these locations.

System of Systems Engineering Guidebook

Office of the Under Secretary of War for Research and Engineering
Office of the Assistant Secretary of War for Mission Capabilities
Office of the Deputy Assistant Secretary of War for Multi-Domain Joint Operations
Systems Engineering and Architecture
3030 Defense Pentagon
Washington, DC 20301
osd-sea@mail.mil
https://www.cto.mil/sea/

Distribution Statement A. Approved for public release. Distribution is unlimited.
DOPSR Case # 26-T-2752.

Approved by
Principal Deputy, Multi-Domain Joint Operations
Office of the Assistant Secretary of War for Mission Capabilities

System of Systems Engineering Guidebook Change Record

Date	Change	Rationale
September 2026	Reorganized document. Added: 2.6 SoS Challenges 2.7 Applicable Standards 2.9 SoS Engineering Management Appendix A: Additional Engineering Considerations for SoS Appendix C: Data Centricity, Artificial Intelligence and Zero Trust Cybersecurity for SoS Updated: 3 SoS Engineering Implementation Appendix B: SE Processes in SoS Environment	Fact of life revisions for currency. Alignment with acquisition transformation efforts.

This page is intentionally blank.

Contents

1 Introduction 1
1.1 Purpose 1
1.2 Scope 1
2 Overview 3
2.1 Background 3
2.2 SoS Structure 5
2.3 Comparison of Systems and Systems of Systems 6
2.4 Types of SoS 8
2.4.1 Virtual 8
2.4.2 Collaborative 9
2.4.3 Acknowledged 10
2.4.4 Directed 10
2.4.5 Additional Considerations 11
2.5 Purpose of SoS Engineering 13
2.6 SoS Challenges 13
2.7 Applicable Standards 16
2.8 SoS Engineering Core Elements 17
2.9 SoS Engineering Management 18
2.9.1 Collaboration Forum and Hierarchy 19
2.9.2 Stakeholder Agreements for SoS 21
2.9.3 Enterprise Engineering Management Plan 22
3 SoS Engineering Implementation 26
3.1 SE Processes in an SoS Environment 26
3.2 Implementing SoS Engineering Core Elements 26
3.2.1 Translating Capability Objectives 30
3.2.2 Understanding Systems and Relationships 35
3.2.3 Assessing SoS Performance Against Capability Objectives 41
3.2.4 Developing and Evolving an SoS Architecture 47
3.2.5 Monitoring and Assessing Changes 56
3.2.6 Addressing Requirements and Solution Options 62
3.2.7 Orchestrating Upgrades to SoS 71

3.2.8 SE Processes in Orchestrating Upgrades to SoS 74
A.1 Digital Engineering in an SoS Environment 79
A.2 Modeling and Simulation in an SoS Environment 81
A.3 Modular Open Systems Approach in an SoS environment 82
A.4 Mission Engineering in an SoS Environment 83
A.5 Software Engineering in an SoS Environment 84
A.6 Value Engineering in an SoS environment 91
A.7 Cyber Considerations 93
A.8 Resource Management Considerations 94
B.1 SE Technical Management Processes in SoS Environment 95
B.2 SE Technical Processes in SoS Environment 104
C.1 SoS Development for Real-Time Campaign Support 113
C.2 Transition from Net-Centricity to Data-Centricity Paradigm 114
C.3 Data Centricity Attributes 119
C.4 AI Benefits and Best Practices in SoS Environment 122
C.5 Zero Trust 125

Figures

Figure 2-1. Notional Depiction of a Typical Enterprise-Level SoS Structure 5
Figure 2-2. SoS Hierarchy and Collaboration 20
Figure 2-3. SEP and EEMP Comparison 23
Figure 2-4. SoS EEMP and System SEPs 24
Figure 2-5. SoS Schedule Example 25
Figure 3-1. SoS Engineering Implementers' View 27
Figure 3-2. Seven Core Elements Provide the Context for Applying the Systems Engineering Processes 28
Figure 3-3. Relationship between Translating Capability Objectives and Other SoS Engineering Core Elements 32
Figure 3-4. Relationship between Understanding Systems and Relationships and Other SoS Engineering Core Elements 38
Figure 3-5. Relationship between Assessing Performance Against Capability Objectives and Other SoS Engineering Core Elements 44
Figure 3-6. Relationship between Developing an SoS Architecture and Other SoS Engineering Core Elements 51

Figure 3-7. Relationship of Monitoring and Assessing Changes to Other SoS Engineering Core Elements 59
Figure 3-8. Relationship between Addressing Requirements and Solution Options and Other SoS Engineering Core Elements 67
Figure 3-9. Relationship between Orchestrating Upgrades to SoS and Other SoS Engineering Core Elements 74

Figure C-1. Layered Architecture 117
Figure C-2. Data-Centric End State 117
Figure C-3. Transition to End State 118
Figure C-4. Transition to Data-Centric SoS End State 119

Tables

Table 2-1. Comparing Systems and SoS 7
Table 2-2. Summary of Types of SoS 11
Table 2-3. SoS Challenges Addressed by SoS Engineering Core Elements 18
Table 3-1. SE Technical and Technical Management Processes Applied to the SoS Engineering Core Elements 29

Table B-1. Technical Management Processes as Described in the SE Guidebook 95
Table B-2. Potential Problems in Managing Risk at SoS Level 101
Table B-3. Technical Processes as Described in the SE Guidebook 105

Preface

This guidebook builds from and supersedes the Department of Defense (DoD) Systems Engineering Guide for Systems of Systems (2008) and presents advances in system of systems (SoS) engineering practices in digital ecosystems. The guidance is intended for SoS in the warfighting context. It describes strategies and processes for SoS engineering that should begin early in development and should reevaluate, revise, and reapply throughout the acquisition life cycle. It aligns with the acquisition reform directed by Executive Order 14265, "Modernizing Defense Acquisitions and Spurring Innovation in the Industrial Base," and fully supports the Secretary of War memorandum, "Transforming the Defense Acquisition System into the Warfighting Acquisition System (WAS) to Accelerate Fielding of Urgently Needed Capabilities to Our Warriors," November 7, 2025, and the accompanying Acquisition Transformation Strategy.

> "The Department must establish new paradigms for acquisition and pivot from program- and platform-centric structures, processes, and strategies to deliver integrated suites of capabilities across platforms and systems."
>
> — 2025 Acquisition Transformation Strategy: Rebuilding the Arsenal of Freedom

As stated in the Acquisition Transformation Strategy, the Department of War (DoW) must establish new paradigms for acquisition and pivot from program- and platform-centric structures, processes, and strategies to deliver integrated suites of capabilities across platforms and SoS. This guidebook provides guidance and best practices to help support this goal.

Each program should tailor the practices and avoid adding a process that does not add value. Identifying the program's key uncertainties and challenges early can help inform decisions on the basic program structure and the activities needed to enable the program to deliver the intended product or services successfully and efficiently in an SoS environment.

> "The WAS is dependent on many parts to deliver the world's most sophisticated and capable weapon systems; systems engineering is one of its most critical disciplines."
>
> — 2025 Acquisition Transformation Strategy: Rebuilding the Arsenal of Freedom

As technologies advance and the nature of threats evolves, SoS engineering plays a crucial role in defining and delivering mission capabilities. This guidebook supports the ongoing modernization of SoS engineering methodologies to keep pace with advancements and emerging challenges.

The Office of the Under Secretary of War for Research and Engineering (OUSW(R&E)) prepared this guidance in coordination with experts in the engineering community. OUSW(R&E) will update the guidance periodically to incorporate advances in the field and reflect relevant/applicable changes in DoW policy.

The guidebook is organized as follows:

Section 1: Provides the purpose and scope of the document.

Section 2: Discusses the types of SoS, common challenges, SoS engineering core elements, and SoS engineering management.

Section 3: Discusses the implementation of systems engineering technical management processes and technical processes in the context of an SoS.

Appendix A: Presents additional SoS considerations related to digital engineering, modeling and simulation, modular open systems, mission engineering, and software engineering (SWE).

Appendix B: Presents detailed considerations for applying individual SE processes in an SoS environment.

Appendix C: Presents SoS considerations related to data centricity, artificial intelligence, and Zero Trust security.

The table below depicts important topics identified in the Acquisition Transformation Strategy and the locations in the SoS Engineering Guidebook where they are addressed. Additional detailed guidance for these topics is located on the OUSW(R&E) Systems Engineering and Architecture (SE&A) website.

Acquisition Transformation Strategy Topic	SoS Engineering Guidebook Section Location
Agile/Iterative Development	3.2.2 Understanding Systems and Relationships, A.5 Software Engineering in SoS Environment
Application Programming Interfaces	3.2.2 Understanding Systems and Relationships, 3.2.4 Developing and Evolving an SoS Architecture, A.2 Modeling and Simulation in an SoS Environment, A.5 Software Engineering in an SoS Environment
Digital Engineering	2.6 SoS Challenges, 3.2.1, Translating Capability Objectives, A.1 Digital Engineering (DE) in an SoS Environment, A.2 Modeling and Simulation in an SoS Environment
Mission Engineering	2.1 Background, 2.3 Comparison of Systems and Systems of Systems, A.3 Modular Open Systems Approach in an SoS Environment, A.4 Mission Engineering in an SoS Environment
Model-Based Systems Engineering	2.6 SoS Challenges, 3.2.1 Translating Capability Objectives, A.1.1 Key Components of Digital Engineering in SoS, A.2 Modeling and Simulation in an SoS Environment
Modeling and Simulation	3.2.4 Developing and Evolving an SoS Architecture, 3.2.7 Orchestrating Upgrades to SoS, A.1.3 Challenges in Implementing DE in SoS, A.2 Modeling and Simulation in an SoS Environment

Acquisition Transformation Strategy Topic	SoS Engineering Guidebook Section Location
Modular Open Systems Approach	2.6 SoS Challenges, 2.7, Applicable Standards, 3.2 Implementing SoS Engineering Core Elements, 3.2.4 Developing and Evolving SoS Architecture, 3.2.8 SE Processes in Orchestrating Upgrades to SoS, A.3 Modular Open Systems Approach in an SoS Environment
Software Engineering	A.5 Software Engineering in an SoS Environment
Technology Insertion	3.2.5 Monitoring and Assessing Changes, 3.2.8 SE Processes in Orchestrating Upgrades to SoS, B.1.3 Technical Assessment Process in SoS Environment
Value Engineering	A.6 Value Engineering in an SoS Environment

1 Introduction

1.1 Purpose

This guidebook addresses systems engineering (SE) considerations for integrating independently useful systems into a larger system that delivers unique capabilities—a system of systems (SoS)—within the DoW. SoS engineering encompasses planning, analyzing, organizing, and integrating multiple system capabilities into existing and/or new SoS that provide a capability not achievable by individual systems alone. The guidebook identifies challenges for SoS engineering awareness that may need to be addressed by systems engineers doing SoS work. The guidebook aims to equip warfighting acquisition professionals with the knowledge and tools necessary to implement effective SoS engineering in complex and dynamic environments. As experience with SoS grows along with incorporating new DoW policy tenets, subsequent versions will be updated accordingly. This guidebook assumes an understanding of SE processes as described in the DoW Systems Engineering Guidebook (2026) (SE Guidebook).

1.2 Scope

This guidebook presents aspects of more than a decade of research and application of SoS engineering in the context of DoW military systems. The guidebook presents an update to the application of SE in today's SoS environment and describes how the eight technical management and eight technical processes outlined in the SE Guidebook are employed in this SoS context.

The SE Guidebook describes these 16 processes in the context of acquisition programs. The characteristics of today's SoS environment have an impact on how these basic processes are applied by the systems engineer of the SoS. This guidebook describes how focusing on the mission, using mission engineering practices, is an enabler of SoS engineering to plan, organize, and integrate system warfighting capabilities. It also discusses the influence of digital engineering on system development and the ability of digital engineering to enhance the architecture development and analysis in an SoS environment. Additional SoS engineering considerations are covered in Appendix A.

The following roles are typically involved within the scope of SoS engineering and are used in this guide (specific role titles may vary from those shown below):

- Stakeholder: An individual or organization having a right, share, claim, or interest in a system or SoS and its possession of characteristics that meet their needs and expectations. Stakeholders at both the system level and SoS levels (including the system owners) may have competing interests and priorities. In some cases, a system stakeholder may have no vested interest in the SoS and may not be known or recognized.

- Program Manager (PM): Designated individual with responsibility for and authority to accomplish program objectives for development, production, and sustainment of a system or SoS to meet the user's operational needs. The PM is accountable for credible cost, schedule, and performance reporting to the decision authority.
- Systems Engineer: Refers to the chief engineer, lead engineer, program lead systems engineer, or the engineering staff responsible for SE processes, including the planning, conduct, or management of SE activities and the incorporation of design considerations throughout the program life cycle.
- SoS Manager: Leads and is accountable for the integration and management of multiple interconnected systems, ensuring they function cohesively to achieve SoS mission objectives. The SoS manager works collaboratively with PMs of the constituent systems to leverage and influence the development of those systems to address SoS needs. Ideally, the SoS manager can influence constituent system development schedules. The SoS manager leads the stakeholder agreements that form the foundations of development and operational management between the SoS level and constituent systems.
- SoS Systems Engineer: Provides technical support to the SoS managers. The SoS systems engineer is responsible for planning, analyzing, organizing, and integrating the capabilities of multiple systems into an SoS capability greater than the sum of its constituent systems. SoS systems engineers must be able to function in an environment in which the SoS manager may not control all the systems that affect the SoS capabilities, and stakeholders have interests beyond the SoS mission objectives.
- System Manager: Manages an assigned system or systems, to include enterprise-wide, and the supporting information technology (IT) infrastructure around them. System managers plan, coordinate, and direct related activities within the organization to meet systems and supporting IT infrastructure needs.
- SoS Engineering Team: Consists of members of SoS technical personnel or sub-teams in the areas of systems engineering, mission engineering, software engineering, digital engineering, requirements management, data management, configuration management (CM), risk management, test and evaluation (T&E), interface management, security/cybersecurity, and other subject matter experts (SMEs) that are responsible for planning, analyzing, organizing, and integrating capabilities of existing and new systems to achieve a capability greater than the sum of its parts. (Note: The SoS engineering team for a given SoS is staffed and governed in accordance with established policy and guidance set forth by the leadership, and the definition can be tailored to comply with applied policy and guidance.) SoS engineering teams complete the tasks necessary to develop and maintain SoS capabilities, even if they are in an organization associated with a constituent system.

2 Overview

2.1 Background

In the DoW today, an SoS generally is recognized as a formal entity when there is an identified operational need, or an SoS may be created to address operational problems in which new needs cannot be supported without cooperative efforts of multiple systems.

With the adoption of the DoW data-centric paradigm (DoD Data Strategy 2020) (transforming from net-centric approach) for information management, developers recognize that systems operate in a broader context with a wider set of implications or impacts more now than they have in the past. Most military systems today are part of an SoS, even if they are not explicitly recognized as such. Operationally, DoW battlespace commanders form an SoS as they bring together a mix of systems in an operation to meet mission objectives. From the standpoint of development and acquisition, however, the DoW has historically focused on independent systems. Most military systems were created and then evolved without explicit SE at the SoS level, leading to post-development integration challenges, unintentional operational constraints, capability limitations or ineffectiveness, as well as operational inefficiencies, which ultimately conspire to limit or restrain warfighting capabilities. SoS engineering is intended to change this historical approach by introducing a mission-focused, SoS lens early in the system acquisition and development life cycle.

The notion of "system of systems" is becoming a critical perspective in thinking about individual system development. SoS are not typically new acquisitions; rather, they tend to overlay an ensemble of existing, evolving, and new systems with the objective of improving the way the systems work together to meet a new need. Under these circumstances, SoS managers, when designated, typically do not control all the requirements or funding for all the individual systems in the SoS and consequently find that they can only influence—rather than direct—system managers to meet SoS needs. The SE approach for the SoS, therefore, should recognize that SoS needs may not be accommodated in the individual systems' development.

Acquisition roadmaps and capability portfolios are being explored as mechanisms for investment decisions. Changes to both the requirements development and acquisition processes have resulted in increased emphasis on addressing broad "user capability needs" as a context for developing new systems. The DoW is updating systems and processes as decisions about requirements and priorities increasingly need to be made in a broad capability or portfolio context. Capabilities-based analyses are becoming the basis for defining user needs.

Once stakeholder organizations confirm the need for an SoS, they typically identify one organization as responsible for the SoS. In most cases, however, designation of SoS leadership does not change the ownership of the constituent systems (independent systems that form part of an SoS) or reduce those systems' existing objectives. In addition, the SoS objective is often framed in terms of improved capabilities and not as a well-specified technical performance objective.

These capabilities-based perspectives expose the interactions and relationships among what were previously considered independent systems, so an SoS becomes apparent as the DoW adopts a capabilities-based approach. As part of this approach, it is essential that SoS programs identify and manage risks at both the SoS level and the system level when considering the interactions and relationships among the independent systems that may form the SoS. To effectively identify and manage technical risks during acquisition, programs need a complete and thorough knowledge of the SE domain. Managing risk implies taking measured and calculated risks deemed appropriate for the effort.

The application of mission engineering practices feeds the front-end processes of SoS engineering by addressing joint mission capability needs to achieve mission success. Mission engineers identify mission threads (MTs) and mission engineering threads (METs). As defined in the DoD Mission Engineering Guide 2.0 and Mission Architecture Style Guide 1.0, mission threads are a sequence of end-to-end mission tasks, activities, and events presented as a series of steps to achieve a mission. METs are the applications of those mission threads in context that include the details of the capabilities, technologies, systems, and organizations required to execute the mission. Analysis of baseline mission architectures (MTs and METs) identify potential gaps or single points of failure for use in identifying capabilities.

SoS engineering typically focuses on the evolution of capability over time, with initial efforts working to enhance the way current systems work together, anticipating change in internal or external effects on the SoS and eventually adding new functionality through new systems or changes to existing systems. In some cases, the aim may be to eliminate or re-engineer systems to provide better or more efficient SoS capability; however, providing more efficient capability may be problematic because features that are redundant across systems may be needed when the systems operate apart from the SoS.

2.2 SoS Structure

An SoS is a set or arrangement of systems that results when independent systems are integrated into a larger system to deliver unique capabilities that none of the individual systems can deliver on their own. The collection of task-oriented or dedicated systems combines resources and capabilities to create a new and integrated system that offers more functionality and performance than simply the sum of the constituent systems. An SoS may deliver capabilities by combining multiple collaborative and autonomous-yet-interacting systems. The mix of systems may include existing, partially developed, and yet-to-be-designed independent systems.

The interdependencies between or among the individual systems being integrated need to be tracked. An SoS may consist of varying technologies that matured decades apart and were designed for different purposes but are now used to meet new objectives that may not have been defined (or known) at the time the systems were designed, developed, and deployed.

Figure 2-1 depicts a notional example of SoS and interdependencies that exist or are developed between or among individual systems. The figure illustrates two notional SoS and their intertwined structural levels, in which each entity at a higher level comprises multiple systems from its immediate lower level. Level 0 represents the lowest level in both SoS structures, composed of independent constituent systems (CSs). Level 1 represents the collection of CSs when they begin to interact with one another and form a SoS. Level 2 represents the SoS as a whole by showing the interactions between Level 1 SoS. The figure also highlights that some CS, for example, the satellite CS in this figure, can simultaneously participate in more than one SoS and become a shared resource between multiple SoS.

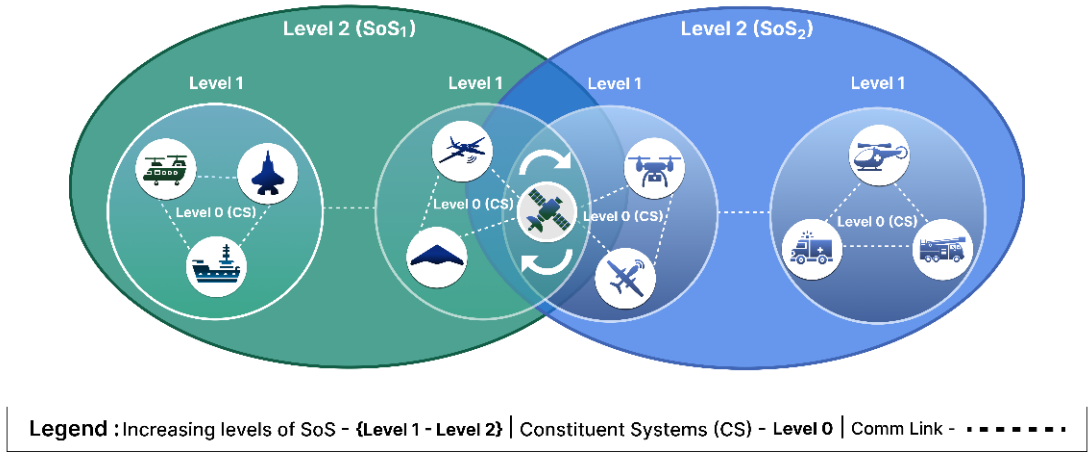


Figure 2-1. Notional Depiction of a Typical Enterprise-Level SoS Structure

While Figure 2-1 provides a notional, high-level depiction, it is crucial for engineers to understand how to represent this structure formally within a model-based systems engineering (MBSE) environment. Adopting a standards-based approach, such as Object Management Group (OMG) Systems Modeling Language (SysML), elevates the architecture from a simple picture to a true engineering model.

2.3 Comparison of Systems and Systems of Systems

An important consideration for SoS engineering is to distinguish a system from an SoS. Although there may be similarities, a key difference is that in SoS the constituent systems can maintain their purpose and usefulness outside of the SoS. The managerial and operational independence of systems is a foundational aspect of an SoS. In addition, SoS engineering literature discusses characteristics that distinguish a system from an SoS. For example, Boardman and Sauser (2008) describe SoS characteristics as follows:

- Autonomy: a system maintains a purposeful existence outside of SoS with operational and managerial independence.
- Belonging: a system may or may not participate in the SoS; the belonging of a system needs to be established from both system's autonomy perspective and the SoS value proposition.
- Connectivity and Diversity: in SoS, heterogeneous constituent systems connect with one another to produce SoS capability.
- Emergence: the interaction between heterogeneous systems results in SoS behavior that is not attributable to an individual constituent system.

The above characteristics distinguish SoS from individual systems, indicating a need for SoS engineering. In addition, Table 2-1 lists common observations on differences between individual systems and an SoS.

Table 2-1. Comparing Systems and SoS

Aspect of Environment	System	System of Systems
Stakeholder Involvement	Clearer set of stakeholders.	Stakeholders at both system level and SoS levels (including the system owners), with competing interests and priorities; in some cases, the system stakeholder has no vested interest in the SoS; all stakeholders may not be recognized.
Governance	Aligned program management and funding.	Added levels of complexity due to management and funding for both the SoS and individual systems; SoS does not have authority over all the systems.
Operational Focus	Designed and developed to meet operational objectives.	Called upon to meet a set of operational objectives using systems whose objectives may or may not align with the SoS objectives.
Acquisition	Aligned to Acquisition Category, milestones, documented requirements, and systems engineering with a Systems Engineering Plan.	Added complexity due to multiple system life cycles across acquisition programs, involving legacy systems, systems under development, new developments, and technology insertion; typically, have stated capability objectives upfront that may need to be translated into formal requirements. Requires careful planning of acquisition management activities across multiple authorities.
Test and Evaluation	Test and evaluation of the system is generally possible.	Testing is more challenging due to the difficulty of synchronizing across multiple systems' life cycles; given the complexity of all the moving parts and potential for unintended consequences.
Mission Engineering	Platform focused analysis of core capabilities.	Identifies capability gaps for a set of missions with SoS that can then be handed off to a systems engineering process.
Boundaries and Interfaces	Focuses on boundaries and interfaces for the single system.	Focus on identifying the systems that contribute to the SoS objectives and enabling the flow of data, control and functionality across the SoS while balancing needs of the systems.
Performance and Behavior	Performance of the system to meet specified objectives.	Performance across the SoS that satisfies SoS user capability needs while balancing needs of the constituent systems.
Requirements	Requirements are defined early and precisely in the life cycle, with formal verification against specifications.	Requirements are defined collaboratively and often evolve, focusing on capabilities rather than specific technical specs. Trade-offs and prioritization must occur across constituent systems.

Aspect of Environment	System	System of Systems
Architecture and Design	Primarily focus on defining the internal structure and interfaces of a single, cohesive system and its components. Architecture is under a single, central design authority with complete control over all system elements. The architecture is designed for a specific life cycle, with upgrades managed by the single authority. The process is relatively predictable. Typically uses a single, integrated toolset (e.g., model-based systems engineering tools) for a system's full life cycle. Relies on controlled, internal specifications and standards to ensure component compatibility. Emphasis on a modular open systems approach to ensure new components can be integrated and to reduce vendor lock-in.	Primarily focus on defining the interconnections, interactions, and emergent behaviors that arise from the collaboration of multiple independent systems. The SoS architect influences, rather than controls, the designs of the constituent systems. Control is collaborative and decentralized. The architecture evolves continuously as new constituent systems are added or removed. The architecture must accommodate this constant change. Requires tools that can model dependencies and behavior across multiple, different systems. Tools must analyze cross-cutting concerns like performance and dependencies, not just internal function. Emphasizes modular open systems approach to ensure new systems can be integrated and to reduce vendor lock-in.

2.4 Types of SoS

There are four types of SoS (Maier 1998) of significance in the DoW: virtual, collaborative, acknowledged, and directed. These are described in the following paragraphs.

2.4.1 Virtual

In a virtual SoS, the systems are loosely coupled, may operate independently, and lack a centrally agreed-upon purpose or goal. The constituent systems do not have direct control over or may not be aware of one another; instead, they collaborate through a shared infrastructure or framework, often using standards or protocols to interoperate. This type of SoS has no central management authority or a centrally agreed-upon purpose for the SoS. Large-scale behavior emerges—and may be desirable—but this type of SoS must rely upon relatively invisible mechanisms to maintain it.

- Key Characteristics:
  - There is no central control.
  - The SoS may have a central manager that facilitates interaction but does not control operations.
  - Systems may have their own independent purpose and operations.
  - Coordination is negligible. The SoS is emergent.
  - Often seen in ad-hoc or loosely integrated systems, such as certain cloud computing networks or federated databases.

Example: The U.S. Global Positioning System (GPS) has many users and applications, and users are not constrained within any scenario. While GPS is centrally managed by the U.S. Space Force, its ubiquitous use is not constrained.

Another example is the World Wide Web (WWW), which consists of independent systems (websites, browsers, applications, and services) that interoperate through shared standards. The internet serves as the underlying infrastructure, while the WWW operates on the application layer. Systems that participate in the WWW as an SoS may not be aware of the existence and/or participation of other systems, hence making it a virtual SoS.

2.4.2 Collaborative

In a collaborative SoS, the constituent systems work together actively, sharing information and collaborating toward a common goal. While there may not be a single controlling entity, they interact voluntarily to fulfill agreed-upon central purposes by coordinating their actions, share data, and work together to solve problems or optimize outcomes.

- Key Characteristics:
  - No single entity is in charge, and there is no designated manager.
  - Constituent systems may act autonomously but cooperate in decision making.
  - Constituent systems have a high degree of interaction and coordination.
  - Collaboration is often motivated by shared objectives or mutual dependencies.
  - Each constituent system may contribute resources or information to the overall success of the SoS.

Example: The internet is a collaborative system. The Internet Engineering Task Force, a global cooperative network, develops standards but has no power to enforce them. The central players collectively decide how to provide or deny service, thereby providing some means of enforcing and maintaining standards.

2.4.3 Acknowledged

An acknowledged SoS has recognized objectives, a designated manager, and resources for the SoS; however, the constituent systems retain their independent ownership, objectives, funding, and development and sustainment approaches. Changes in the systems are based on collaboration between the SoS and the system. Communication is often in the form of status updates or signals that indicate a constituent system health or readiness.

- Key Characteristics:
  - Recognized objectives, a designated manager, and resources for the SoS.
  - No central entity controls or directs operations on constituent systems.
  - Constituent systems are aware of each other, but the interactions are limited to acknowledgment of one another and acknowledgment of information or data that could influence actions.
  - Typically, constituent systems do not share control or decision-making responsibilities.

Example: The Link 16 combat communication system is managed by the Defense Information Systems Agency, but the radios that use it are system specific with each Military Service acquiring their own fit-for-purpose solutions.

2.4.4 Directed

A directed SoS is built and managed to fulfill specific purposes. A directed SoS is centrally controlled and managed during long-term operation to continue to fulfill those purposes as well as any new ones the constituent system owners might wish to address. The constituent systems are coordinated and directed to achieve a common goal, with the central entity assigning tasks, providing instructions, or overseeing the execution of specific actions. The constituent systems maintain an ability to operate independently, but their normal operational mode is subordinated to the central managed purpose.

- Key Characteristics:
  - A central entity (or coordinating SoS) controls the operation of constituent systems but not the management.

- Constituent systems are typically tightly integrated and must follow centralized instructions or directions.
- The SoS works toward a clearly defined objective, with central management ensuring alignment.

Example: An aircraft carrier is a directed SoS.

2.4.5 Additional Considerations

Table 2-2 summarizes a comparison of the types of SoS.

Table 2-2. Summary of Types of SoS

Types of SoS	Degree of Control	Interaction/Coordination	Centralized Authority
Virtual SoS	Low	Loosely coordinated, independent	None
Collaborative SoS	Low	Voluntary collaboration and data sharing	No single entity in control
Acknowledged SoS	Medium	Acknowledge of state or status	Constituent systems agree to work together and an organization is designated to manage the SoS. (constituent systems can have their own authority to manage their designated systems but collaborate with acknowledged SoS manager)
Directed SoS	High	Centralized control and direction	Centralized authority

Most DoW SoS are combinations of the four types. For example, the Global Navigation Satellite Systems (GNSS) SoS is a collaborative SoS that includes U.S. GPS, Russian GLONASS, EU Galileo, and China BeiDou. Each of these systems has volunteered to provide free use of their systems in support of the global community. These systems interoperate through International Civil Aviation Organization (ICAO) standards and recommended practices.

It may be necessary to define a higher-level SoS as one type while categorizing segments into multiple types. It is important to apply the correct SoS management approach to each of the segments of the SoS.

The U.S. GPS, which collaborates in the GNSS, is an example. A host of applications worldwide use GPS as part of the virtual SoS, standardized around its three basic services: absolute location, relative movement, and time transfer. Each of these domains define additional standards for use of these services in their domain. When new satellites are launched, the GPS PM (U.S. Space Force) and launch agency work together in an acknowledged SoS. The launch systems and payloads are independent but interact in accordance with contractual agreements.

The U.S. GPS also includes a directed SoS that contains multiple satellite versions, several ground stations, and the satellite launch payload system. U.S. GPS consists of three segments: space, control, and user. U.S. Space Force develops, operates, and maintains the space and control segments.

In the GPS user segment, the relationship between GPS satellites and antenna/receivers/clocks forms another collaborative SoS. The user segment includes many types of systems that collaborate, standardized across domains like aviation, maritime, civilian, etc.

Identifying an SoS by its type supports the selection and application of the appropriate systems engineering and management methods that must be implemented and followed at the SoS level. DoD Instruction 8410.02, "Support to DoD Information Network Operations" (2021), provides DoW-wide operational, organizational, and technical capabilities for securing, operating, and defending the DoW information network in the absence of explicit shared objectives or management.

The DoW sees a growing number of acknowledged SoS. Like directed SoS, acknowledged SoS have recognized authorities and resources (that may be temporary) at the SoS level; however, because an acknowledged SoS comprises systems that maintain independent objectives, management, and resources, along with independent development processes, these SoS are largely collaborative in practice.

For systems in an acknowledged SoS, their normal operational mode is not subordinate to a central managed purpose. Defense acquisition and funding are still largely platform-focused, so many SoS managers do not have authority over other constituent systems. They typically try to address SoS objectives by leveraging the development of other constituent systems, which are normally more long-standing and better supported than the SoS. As with collaborative SoS, changes in systems to meet SoS needs are based on agreement and collaboration founded on mutually beneficial outcomes, not top-down authority from the SoS manager.

The DoW is creating structures or standing organizations to address higher-level capability needs and investments. As more systems are integrated into the DoW's data-centric IT, systems are evolving from sets of individual systems to sets of services that work in different combinations to meet different user needs. The DoW continues to address the issues of systems engineering in data-centric enterprise systems.

2.5 Purpose of SoS Engineering

The concept of SoS engineering has evolved significantly over the years, particularly within the DoW, reflecting a shift from managing stand-alone systems to a holistic management of interconnected systems comprising various technologies, disciplines, and components. This evolution was documented in the SE Guide for SoS (2008), which offered insights into integrating complex systems effectively to enhance military capabilities.

As with all engineering, the goal of SoS engineering is to enable mission success, which, in a military context, often hinges on the ability to deploy capable and reliable systems in a coordinated manner. This guidebook underscores the importance of SoS engineering in modern warfare and warfighting strategies, where technological superiority and integration can significantly influence the outcome of operations. The foundational principles that influence SoS engineering practices are:

- Interoperability and Integration: Emphasizing the ability of different systems to work together seamlessly, which is crucial for joint operations and multi-national warfighting activities.
- Flexibility and Adaptability: Systems must be designed to be flexible and adaptable, allowing for modifications in response to changing technologies or operational needs without extensive overhauls.
- Enhanced Decision-Making: Through the integration of systems, decision-making processes are enhanced, providing commanders and operators with comprehensive situational awareness and response options.
- Establishing and Sustaining Collaboration: Since non-directed SoS rely on systems not subject to centralized control, providing an environment where collaboration can be achieved and sustained is critical for SoS success.

2.6 SoS Challenges

Over the past two decades, SoS research and development have identified unique challenges that a systems engineer may encounter when applying SE throughout the life cycle of an SoS (Dahmann 2014, Honour 2013, Madni and Sievers 2014, McDermott 2025). These challenges largely arise from the unique SoS characteristics and the different SoS types.

- Authorities: Each constituent system within the SoS may have its own owners and stakeholders. As a result, multiple system owners need to decide on a collaboration pattern to contribute to the SoS purposes. The chosen SoS collaboration pattern determines the type of SoS (discussed in subsection 2.4) and how high-level concerns, such as SoS management, coordination style, and funding, are handled.

- Leadership: The stakeholders of each constituent system within the SoS may have its own interests and objectives that can differ from those of the SoS. The SoS systems engineers need to create mechanisms to align these differences and find an SoS solution while maintaining the independent interests of constituent systems.

- Integration: An SoS-level integration of a constituent system may not have been considered during the system's architectural design and development. In some cases, the architectural constraints may inhibit such integration, leading to interoperability concerns within the SoS. These concerns can arise due to technical incompatibility between interfaces, communication protocols, data formats, etc.

- Funding: Funding is frequently a challenge for proposing changes to existing systems or adding new systems to the SoS.

- Capabilities and Requirements: Each constituent system has its own requirements, and meeting these requirements may not align with the SoS capability needs. In this case, the SoS team may need to propose changes to the current constituent systems or add new systems to address SoS capabilities. For a directed SoS, part of the engineering process can be explicitly aligning and bi-directionally tracking constituent requirements with SoS requirements.

- Evolutionary Behavior in SoS vs. Constituent Systems: Changes to SoS capabilities usually occur in the constituent systems, which makes SoS evolution more complicated to manage than individual systems. Individual constituent systems also can change independent of SoS capability needs, creating unanticipated effects at the SoS level. An SoS could have unique inputs and outputs beyond those provided by individual constituent systems. In SoS evolution, one must take care that SoS unique capabilities, inputs, and outputs are attained, preserved, and appropriately managed at the SoS level. As a result, SoS evolution must be coordinated at the SoS level and managed across constituent systems and perhaps other SoS, even though this management may be indirect.

- Emergence: Constituent systems within the SoS are interdependent. Thus, the independent evolutionary behavior of a constituent system (e.g., technological advances) can affect other constituent systems, leading to a desired or undesired emergent behavior at the SoS level. Desired emergent behaviors are the main reason for constructing SoS, while undesirable ones need to be evaluated for preventing adverse consequences during SoS operation. This should also include adaptability to emerging or changing operational environments.

- Test and Evaluation: The evolutionary behavior of SoS and constituent systems, interdependencies, and SoS emergence can make the T&E challenging. Evaluating SoS is more complicated than just combining individual system tests and should include higher-level evaluations of interactions among systems and between systems and the operational environment. The evolutionary behavior at both the system and SoS levels necessitates a continuous evaluation, making the traditional end-to-end testing costly.

- Architecture and Design: Defining an SoS architecture and design can be a challenge as it may involve bringing together constituent systems with dissimilar interfaces and data requirements. SoS design challenges involve navigating inherent complexity, varied technologies, and evolving constituent systems. These factors create significant problems for successful integration, interoperability, testing, and security. SoS architectural challenges present difficulty in defining and enforcing quality attributes, ensuring consistency across disparate systems, integrating legacy components, and adapting to technological advancements. To help alleviate this challenge and minimize integration risk, it is essential that constituent systems be designed and developed using a modular open systems approach (MOSA) to the maximum extent practicable.

SoS engineering helps to shift the focus of systems engineering efforts from independent systems to addressing the SoS challenges discussed above. Implementing SoS engineering within the warfighting context presents unique issues, primarily because of the scale and critical nature of military operations. These issues include technological disparities, alignment of different system life cycles, and maintaining security and robustness. This guidebook discusses strategies to mitigate these issues, such as:

- Stakeholder Engagement: Involving all stakeholders early and throughout the system development process to ensure alignment with strategic objectives and operational requirements.
- Life Cycle Management: Implementing rigorous life cycle management practices to synchronize the introduction, support, maintenance, and phasing out of diverse systems.

- Regular Reviews and Assessments: Conducting periodic reviews and assessments to ensure that the SoS engineering is on track to meet its goals and adapting strategies as necessary.

A significant challenge is the establishment of an SoS model that addresses the aspects of the constituent systems. A model-based approach for SoS provides a unified, visual, and interactive platform for managing the inherent complexity, diverse components, and asynchronous timelines of large-scale integrated systems. It also improves communication among various stakeholders, facilitates and allows parallel development, reduces project risks by enabling early identification of issues, and ensures consistency and accuracy.

If the constituent systems of an SoS have instituted MBSE processes, then a federation of MBSE models can be established if those models have sufficient interoperability to do so. The authoritative source of truth for a model-based system or SoS is a digital engineering environment that contains a validated, integrated set of system models and data, stored in a central database that ensures consistency, traceability, and accuracy across disciplines and life cycle phases by acting as the single, reliable point of reference for all system information. The SoS engineering team should consider applying MBSE to understand the end-to-end performance of the overall SoS. If early models of the systems forming the SoS can be constructed and validated, better identification of potential problems can be understood at early stages of the life cycle. More information on MBSE can be found in Appendix A of this guide.

2.7 Applicable Standards

Standardization supports the achievement of compatibility, interchangeability, commonality, and interoperability of parts and processes. The implementation of consensus-based standards is extremely important for an SoS to achieve interoperability and integration of individual systems. Standards help ensure compliance with regulatory requirements, interoperability, and achieving strategic objectives.

While there are no government standards specific to SoS engineering, the following non-government standards provide relevant guidance:

- "System of systems (SoS) Considerations in Life Cycle Stages of a System"–ISO/IEC/IEEE 21839 (2019) provides guidance on a set of critical SoS considerations to be addressed that apply to systems at key points of their respective life cycles of the system of interest (SoI).
- "Guidelines for the utilization of ISO/IEC/IEEE 15288 in the context of system of systems (SoS)"–ISO/IEC/IEEE 21840 (2019) provides guidance on the application of processes in ISO/IEC/IEEE 15288 to SoS.

- "Taxonomy of SoS" – ISO/IEC/IEEE 21841 (2019) outlines the taxonomy of SoS based on the authority of relationships.
- SysML: As the universally adopted standard for MBSE, SysML provides the necessary notations and semantics to rigorously capture system requirements, structure, behavior, and parametric. Its integrated diagram set is essential for building a holistic model of a complex SoS.
- OMG Unified Architecture Framework (UAF): The UAF is beneficial in SoS environments. It provides a structured framework of views (e.g., Capability, Operational, Systems, Services) that directly maps the architectural analysis required for SoS and aligns with the "DoW Architecture Framework viewpoints" mentioned in Appendix C. Using UAF provides a clear path for modernizing DoD Architecture Framework based practices.

DoDM 4120.24, "Defense Standardization Program (DSP) Procedures," provides processes on standardization considerations, how to document standardization decisions, and a discussion of the tailoring of standardization documents. It also provides references to resources for the standardization process. Defense standardization products are published in ASSIST (https://assist.dla.mil), including U.S. ratified international standardization agreements and adoption notices for non-government standards.

2.8 SoS Engineering Core Elements

SE in DoW SoS environments can be described in terms of a set of seven core elements. In the context of SoS engineering, "core element" is defined as engineering activities that are essential for managing the integration and life cycle of an SoS. To provide a structured framework for applying these augmentations, this guidebook recommends aligning them with a comprehensive MBSE methodology. A prime example is the Object-Oriented Systems Engineering Method (OOSEM), which provides a logical, step-by-step process that integrates the core elements with the standard SE processes throughout the system life cycle. This gives engineering teams a clear and repeatable path for execution. These core elements are activities, conducted by SoS systems engineers and the SoS engineering team, that leverage the SE processes to assess and refine operational environments, boundaries, and interfaces specified by stakeholders in order to achieve higher-level mission capabilities that otherwise cannot be achieved by a single system. These seven SoS engineering core elements are:

- Translating capability objectives
- Understanding systems and relationships
- Assessing performance against capability objectives

- Developing and evolving an SoS architecture
- Monitoring and assessing changes
- Addressing requirements and solution options
- Orchestrating upgrades to SoS

Understanding the tasks facing the SoS systems engineer leads to better appreciation of how basic SE processes are applied in an SoS environment and suggests some emerging principles for SoS engineering. The core elements of SoS are intended to augment current DoW SE practices to account for the SoS challenges. Table 2-3 demonstrates a mapping between areas of SoS challenges discussed in section 2.6 and SoS engineering core elements listed in section 2.8.

Table 2-3. SoS Challenges Addressed by SoS Engineering Core Elements

	Challenges							
Core Element	Authorities	Leadership	Integration	Capabilities and Requirements	Evolutionary Behavior	Emergence	Test and Evaluation	Architecture and Design
Translating capability objectives					✓			
Understanding systems and relationships	✓	✓	✓	✓	✓	✓		✓
Assessing performance against capability objectives					✓	✓	✓	
Developing and evolving an SoS architecture			✓	✓	✓	✓	✓	✓
Monitoring and assessing changes			✓	✓	✓	✓	✓	✓
Addressing requirements and solution options	✓	✓	✓	✓	✓			✓
Orchestrating upgrades to SoS		✓					✓	

2.9 SoS Engineering Management

SoS engineering management is frequently complicated by a lack of central governance. SoS governance typically includes multiple organizations and decision authorities through which extensive collaboration must take place to allocate resources and coordinate activities. Effective SoS governance is critical to effective SoS engineering management and the integration of efforts across multiple independent systems in an SoS. These issues are reflected in the authorities and leadership challenges described in section 2.6.

While the SoS may have a manager and resources devoted to the SoS objectives, the constituent systems in the SoS have their own objectives, PMs, sponsors, funding, systems engineers, and independent acquisition and sustainment programs. In addition, some constituent systems will participate in multiple SoS. Frequently, leadership and stakeholders focus on management issues that result from the overlapping decision authority rather than SE technical implications. SoS increase the complexity, scope, and cost of both the planning (management) process and systems engineering process.

Successful SoS engineering management requires reaching across organizational boundaries to establish a shared set of objectives and a resourced plan. Experienced managers are needed in an SoS environment, which requires considerable flexibility and collaboration to negotiate among the competing interests. For virtual, collaborative, and acknowledged SoS, the systems engineering activities will be more collaborative than for directed SoS.

Securing the level of participation required from the constituent systems, and their systems engineers, can be challenging. Principally during the early, formative stage of an SoS, the tendency can be to ask the systems engineers of the constituent systems to participate in all aspects of the SoS engineering process. Given the potential system-level workload of these systems engineers, this amount of support may not be sustainable in the long run. A successful SoS technical management approach must identify a sustainable pattern of participation of the various stakeholder engineers.

2.9.1 Collaboration Forum and Hierarchy

To foster reaching across organizational boundaries and relieve burden from the constituent systems engineers, some SoS may establish an intermediary systems engineering collaboration forum. This forum acts as a bridge between the SoS systems engineers and the constituent systems engineers, supporting SoS technical planning and resolving engineering issues that span multiple systems. It can also determine the necessity and structure of SoS-level technical reviews. Such a forum may be particularly effective for collaborative or acknowledged SoS, and in complex, joint, or large SoS where it is impractical for the SoS systems engineers to work directly with many constituent systems engineers. It also allows organizations (Services, Agencies, etc.) to retain more control on how the constituent systems under their authority are directed to integrate into the SoS. Figure 2-2 shows how an engineering collaboration forum might act as a broker between the SoS level and the organizations that have authority over the constituent systems that make up the SoS.

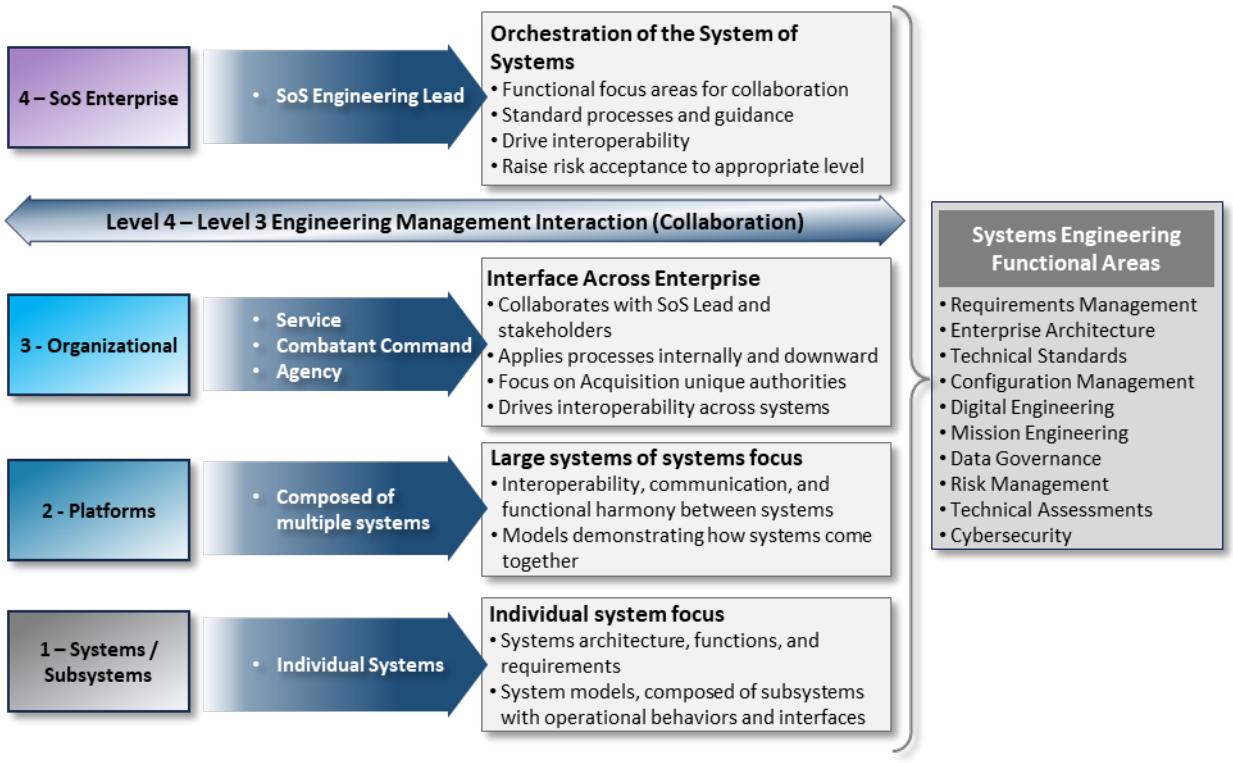


Figure 2-2. SoS Hierarchy and Collaboration

SoS managers and systems engineers should focus on those areas that are critical to the SoS success and leave system-level issues to their constituent PMs and systems engineers. The systems engineers at the constituent system level have the knowledge and responsibility and are in the best position to address implementation details. The biggest challenges are determining the areas that need to be addressed at the SoS level to enable SoS systems engineers to focus on them. SoS systems engineers typically concentrate on risk, architecture, CM, and data as they apply across the SoS. For SoS, a key area of concern is the synchronization across development cycles of the systems. The SoS manager should develop an Integrated Master Plan and Integrated Master Schedule (IMS) that focus on key intersection points and dependencies across the SoS rather than on individual systems' details. For more information on developing an Integrated Master Plan and IMS, see the SE&A policy and guidance web page (https://www.cto.mil/sea/pg).

A governance and management structure that addresses these realities enables SoS engineering to be more effective in all phases of the processes outlined in this document. Without a solid governance and management approach for an SoS, independent authorities are unlikely to accept guidance from an SoS systems engineer they do not manage, placing the constituent systems engineer in an untenable position in attempting to support an SoS.

2.9.2 Stakeholder Agreements for SoS

The operational and managerial independence of constituent systems leads to independent stakeholders and objectives that may only partially overlap with a wider set of SoS stakeholders and SoS-specific objectives. Foreseeably, some aspects of constituent systems may need to evolve or adapt to the SoS needs. Furthermore, the constituent systems may become part of multiple SoS, and contention may arise among different SoS.

In addition, the evolution of constituent systems may occur independent of the SoS without prior engagement of the SoS stakeholders or evaluation of SoS capability impact. These independent and siloed evolutions of constituent systems can have significant impact on the SoS behavior because of interdependency and interactions with other systems. The SoS manager and the SoS team need to align independent evolution of individual constituent systems with the SoS-level desired capabilities.

This alignment of stakeholder and objectives can be facilitated through stakeholder agreements. Agreements in the SoS context span a wide spectrum of applications. An SoS can use working agreements to share what it needs in terms of constituent systems and their required relationships to achieve SoS objectives. Organizations owning the constituent systems can use them and assess whether they are able to supply the SoS and comply with the necessary relationships with other organizations. This can significantly help with aligning the interests of independent organizations with the SoS needs.

These working agreements specify responsibilities of the SoS and constituent systems. Examples include a Memorandum of Agreement (MOA) for addressing SoS data management and a Memorandum of Understanding (MOU) for formalizing relationships between the SoS and constituent systems. Furthermore, the agreements can be achieved through various working groups and boards whose major objective is to ensure successful SoS development and evolution by helping with the SoS management and updates. Examples include Interface Working Group for interface management and update, a Capability Trade Council (CTC)[1] and a Configuration Control Board (CCB) for configuration management and update, and a Risk Management Board (RMB) for identifying and assessing risks to the SoS and potential mitigation plans. A collaboration forum and structure such as that discussed in 2.9.1 and depicted in Figure 2-2 could be established through a stakeholder agreement. Members from individual systems should be encouraged to participate in these boards and working groups to minimize the gap between their interests and those of the SoS and to align their systems with SoS needs and capabilities.

[1] November 7, 2025 SecWar memorandum, “Transforming the Defense Acquisition System into the Warfighting Acquisition System to Accelerate Fielding of Urgently Needed Capabilities to Our Warriors”, replaces Configuration Steering Boards with Capability Trade Councils.

SoS managers may use formal contracts to manage SoS relationships with constituent systems. These contracts are most likely to be outcomes of working agreements. Finally, SoS agreements may be encoded into policy, regulation, or laws. These would apply to all constituent systems of an SoS. For example, telecommunication SoS operate in a heavily regulated environment, and much of their interdependence is defined by regulations.

The ISO/IEC/IEEE 15288 (2023) introduces two SE agreement processes: the Acquisition process and the Supply process. The ISO/IEC/IEEE 21840 (2019) provides additional guidance on these agreement processes as applied to SoS. From the SoS perspective, the SoS is the acquirer, and the organizations owning the constituent systems are the suppliers. The acquirer SoS leverages supplier constituent systems to provide new capabilities that go beyond the simple accumulation of each system's capabilities. Stakeholder agreements improve the coordination and cooperation of constituent systems toward addressing the desired SoS capabilities.

2.9.3 Enterprise Engineering Management Plan

PMs and systems engineers develop a systems engineering approach to guide the systems engineering activities of an individual acquisition program. This engineering management tool is typically documented in a Systems Engineering Plan (SEP) (DoD SEP Outline, Version 4.1, May 2023), or equivalent, but may be captured in the program's digital ecosystem, or a combination of both. For a directed SoS, where a central organization holds clear authority, the SoS may use a SEP as the document of choice for technical planning. For a collaborative, acknowledged, or virtual SoS, where no single organization holds clear authority, the SoS manager and SoS systems engineer should consider using an Enterprise Engineering Management Plan (EEMP) to guide the overall systems engineering activities across the SoS enterprise. The following paragraphs provide guidance in developing an EEMP.

Note: An EEMP is a new approach the DoW has developed as a pathfinder in an existing SoS effort; at this time, using an EEMP is a recommendation only. SoS may use a tailored traditional SEP if desired.

An EEMP establishes an enterprise approach to communicate, align, identify, track, and guide key enterprise-level activities among organizations responsible for oversight of the SoS components. Figure 2-3 depicts the parallels between a single system SEP and an SoS EEMP. The EEMP addresses the complexity and unique challenges of enterprise-level SoS systems engineering, serving as a critical tool to manage the structure, operation, information, processes, and technology changes necessary to achieve SoS objectives. By developing and agreeing to an EEMP, stakeholders can overcome some of the authorities and leadership challenges outlined in section 2.6. A pre-established agreement with defined processes enables more effective and timely engineering collaboration and decision making across stakeholders.

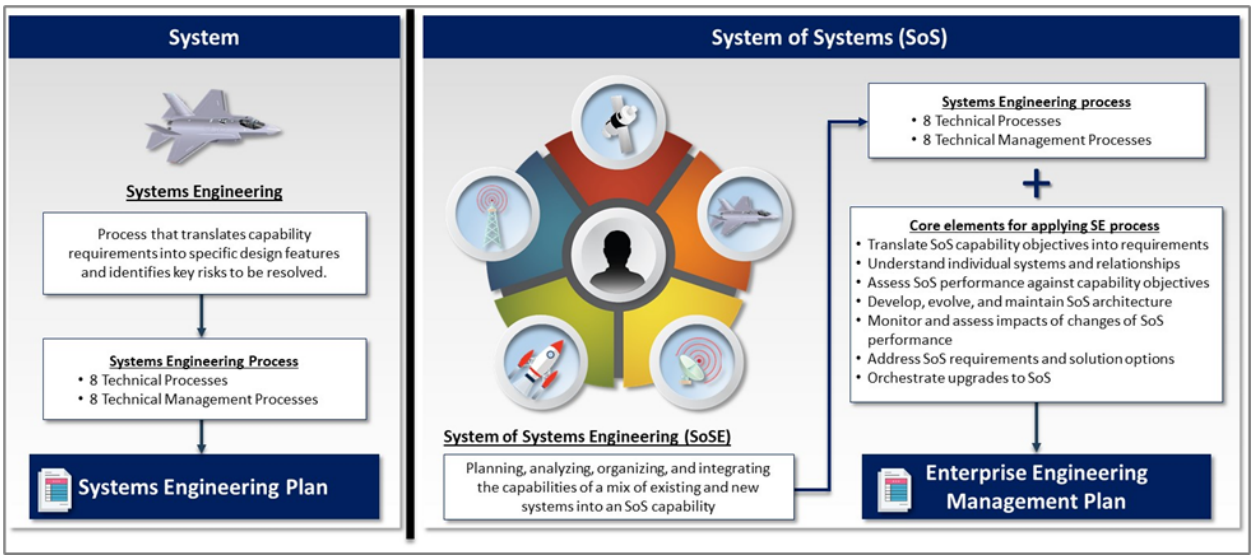


Figure 2-3. SEP and EEMP Comparison

The EEMP should describe, as necessary, how the SE processes (eight technical and eight technical management) and stakeholder agreements will be executed to support the seven SoS engineering core elements. The SoS engineering core elements augment current DoW SE practice to account for the challenges of dealing with the planning, analyzing, organizing, and integrating of capabilities or systems into a larger SoS. The EEMP should communicate to the constituent systems engineers how the SE processes will be executed and managed across the SoS enterprise (e.g., how configuration management function within the SoS).

Figure 2-4 places the EEMP and SEP within the hierarchy of the organizations and systems involved in an SoS. Given the number and diversity of systems and organizations involved, as well as the varying lines of authority, it is neither practical nor advisable for SoS systems engineers to provide direction or guidance to individual systems. Instead, the EEMP functions as an agreement between Level 4 (SoS Enterprise) and Level 3 (Organizational) organizations, designed to establish approaches for executing systems engineering activities across the SoS levels. It ensures that all stakeholders within the enterprise understand SoS engineering processes and are aligned toward achieving the same objectives.

The EEMP (or tailored SEP) should be drafted by the SoS systems engineer in coordination with the DoW Components and Agencies overseeing the constituent systems that compose the SoS. The EEMP and individual system SEPs (each system should develop a SEP) should be coordinated and aligned to the maximum extent possible. Specific approval and concurrence on the EEMP must be determined on a case-by-case basis.

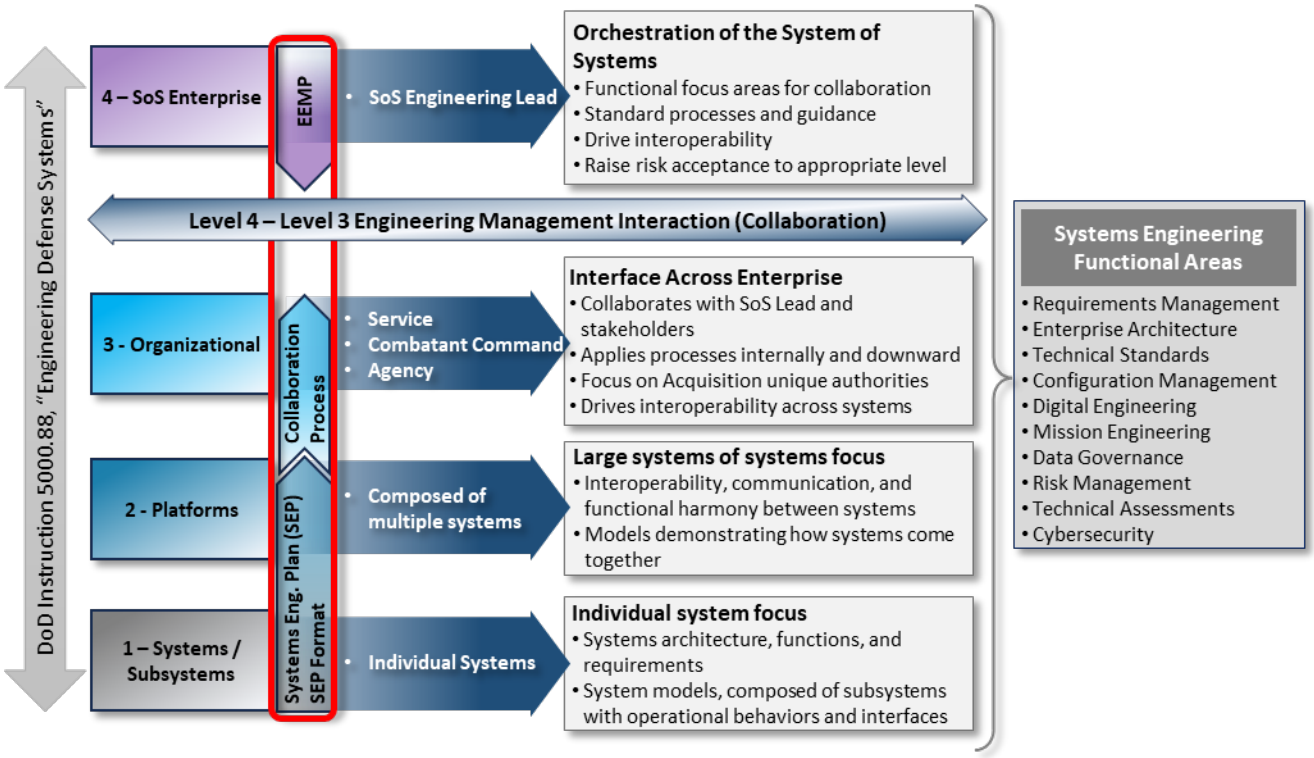
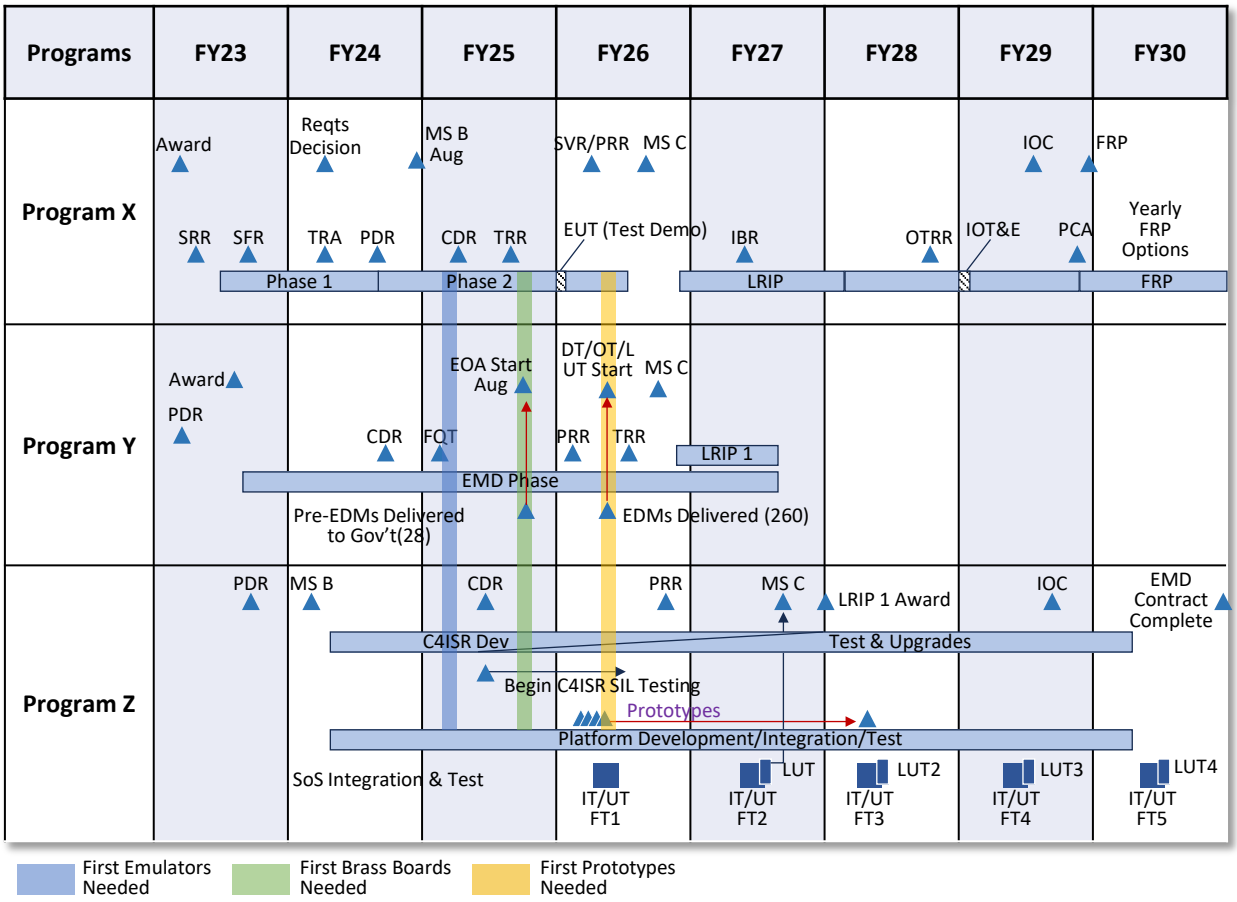


Figure 2-4. SoS EEMP and System SEPs

Beyond describing how the SE engineering processes (requirements, architecture, configuration management, etc.) will be executed and managed across the SoS enterprise, the EEMP should address (but is not limited to):

- Enterprise engineering governance, including:
  - SoS manager, DoW Component, and constituent systems organizations, roles, responsibilities, and relationships. Roles and responsibilities include aligning technology initiatives with business strategy and ensuring that engineering practices are consistent, efficient, and effective across an organization
  - Associated decision boards.
  - SoS systems engineering technical reviews.
  - Resolution of enterprise engineering issues that cross the interdependent programs, program executive offices, DoW Components, and SoS lines.
  - "Triggers" that require an SoS member to inform the others if there is a cost, schedule, or performance deviation.
- Establishment of a technical baseline approach (functional, allocated, product).
- Alignment of schedules across the interdependent programs and who maintains those schedules as depicted in Figure 2-5.

- Data that needs to be shared by the SoS and constituent systems.
- SoS and constituent SE deliverables.
- Links to other management efforts, including the Acquisition Strategy.



Source: DoD SEP Outline, May 2023.

Figure 2-5. SoS Schedule Example

3 SoS Engineering Implementation

Section 2.8 introduced the seven SoS engineering core elements. Section 3.1 and Appendix B address the implementation of the eight technical management and eight technical SE processes in an SoS environment. Section 3.2 provides a detailed description of the SoS engineering core elements and provides recommended activities for the SE processes as they apply to each of the core elements.

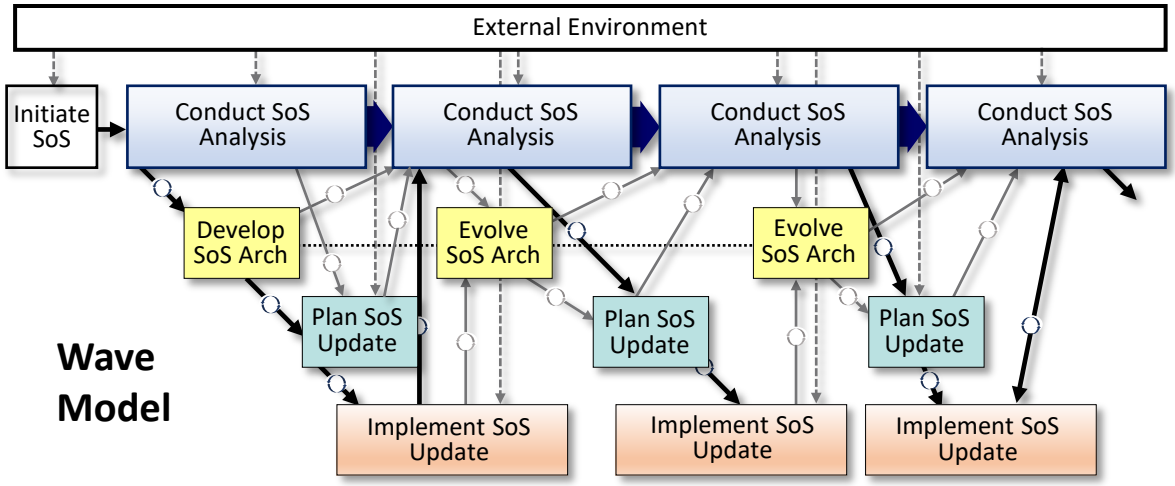
3.1 SE Processes in an SoS Environment

The SE processes are used by Government and contractor organizations to provide a framework and methodology to plan, manage, and implement activities throughout the acquisition life cycle. SE planning and execution should focus on applying the processes and tools in a rigorous, integrated, and disciplined manner to achieve a system solution that balances performance, cost, schedule, and risk. The eight technical management processes, applicable to all acquisition pathways, provide a consistent framework for managing technical activities and identifying technical information and events critical to the success of the program. The eight technical processes ensure the system design and the delivered capability reflect the requirements that the stakeholders have expressed.

The eight technical management and eight technical processes apply to a varying degree to SoS engineering. The difference for SoS is the context or environment in which these processes are conducted or applied. The SoS engineering team implements the seven core elements, introduced in section 2.8, largely drawing from the eight technical management and eight technical processes and tailoring them to the particulars of the SoS context and environment. Appendix B provides a more detailed discussion of the processes and their importance in an SoS environment.

3.2 Implementing SoS Engineering Core Elements

SoS engineering is an ongoing iterative process for developing an evolving SoS architecture as shown in Figure 3-1. The SoS architecture provides an integrated view of the ensemble of systems within the SoS. The development of the architecture of an SoS is an important core element for SoS engineering because it frames and supports design changes to the SoS over time. As in SE of new systems, the systems engineer in an SoS addresses requirements and implementation approaches; monitors development, integration, and test; and assesses the impact of the changes to the end user’s capability needs.



Source: Adapted from "Wave Process Model" presented in Dahmann et al. 2011.

Figure 3-1. SoS Engineering Implementers' View

The SoS engineering core elements are implemented under the auspices of the SoS systems engineer in partnership with the systems engineers of the constituent systems. The SoS systems engineer invests substantial time understanding changes that are outside his span of control but that could potentially affect the SoS. The SoS engineering team tracks this information as well as outputs from the other SoS engineering core elements and assesses feedback on the SoS from the field. The SoS systems engineer focuses on monitoring and assessing changes and, in fact, anticipating change as a core element of the SE for SoS. Using a MOSA in an SoS architecture can support the SoS systems engineer's goal to address user needs and feedback through rapid and evolutionary system upgrades.

Well-founded and thorough SE practices in an SoS environment rely on identifying and implementing the eight technical management and eight technical processes in the seven core elements of SoS (Figure 3-2) within the constraints and direction of the stakeholder agreements. This results in constituent systems operating effectively to achieve high-level warfighting capabilities.

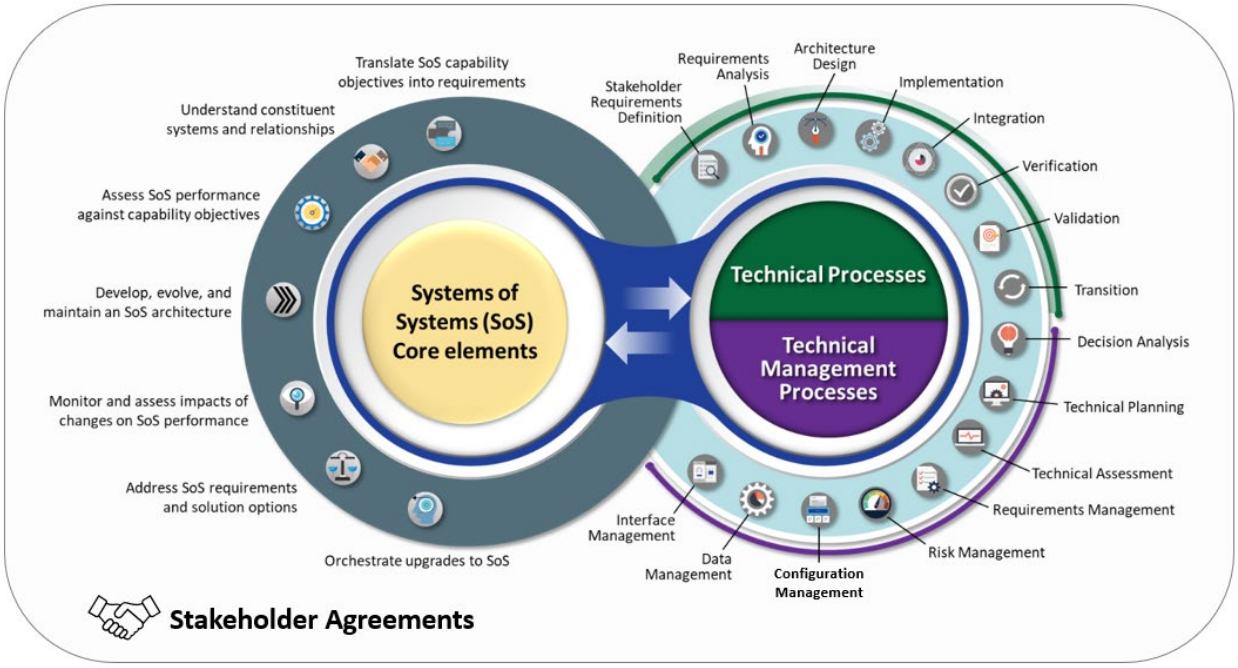


Figure 3-2. Seven Core Elements Provide the Context for Applying the Systems Engineering Processes

The core elements are conducted on an ongoing basis. Whereas a systems engineer of a single system typically implements the 16 SE processes using a waterfall, incremental, or iterative approach, there is less structure in timing or sequencing these SoS SE core elements. They may be conducted by members of single or multiple SoS engineering teams depending on the size or scope of the SoS.

Three of the core elements reflect areas critical to SoS engineering: translating capability objectives, understanding systems and relationships, and monitoring and assessing changes. These elements also apply to individual system SE, but because external influences play such a significant part in the SoS environment, they are emphasized for SoS.

The SoS engineering core elements are arranged in Table 3-1 in what would typically be a starting order of operations. This is an iterative process that will vary in the order and use of the SE processes over time. The SoS engineering core elements are not necessarily executed in a predetermined order.

As the SoS capability is developed, certain elements need to be addressed at the time an SoS is acknowledged. These include translating capability objectives, understanding systems and relationships, and assessing performance against capability objectives. The results of these elements provide the basis for the others; however, these elements are revisited as the SoS evolves and may be implemented concurrently. While Figure 3-2 illustrates the relationship between the core elements and the standard SE processes, it does not prescribe a specific workflow. To execute this in a repeatable and rigorous manner, teams should adopt a formal systems engineering method. The Object-Oriented Systems Engineering Method (OOSEM), for instance, provides a proven life cycle framework that orchestrates these interactions. For example:

- The OOSEM activity of "Analyze Stakeholder Needs and Requirements" directly implements the "Translating Capability Objectives" core element, leveraging the Stakeholder Requirements Definition and Requirements Analysis processes shown in the figure.

- The OOSEM activities of "Define Logical Architecture" and "Synthesize Physical Architecture" are the direct implementation of the "Developing and Evolving an SoS Architecture" core element, using processes like Architecture Design and Decision Analysis.

Adopting such a method provides the necessary procedural glue to effectively put the concepts shown in Figure 3-2 into practice.

Table 3-1. SE Technical and Technical Management Processes Applied to the SoS Engineering Core Elements

	Technical Processes								Technical Management Processes							
SoS Engineering Core Elements	Stakeholder Requirements Definition	Requirements Analysis	Architecture Design	Implement	Integrate	Verify	Validate	Transition	Technical Planning	Decision Analysis	Technical Assessment	Requirements Management	Risk Management	Configuration Management	Data Management	Interface Management
Translating Capability Objectives	X								X			X	X	X	X	
Understanding Systems and Relationships		X											X	X	X	X
Assessing Performance against Capability Objectives							X			X	X	X	X		X	
Developing and Evolving an SoS Architecture	X	X	X						X	X		X	X	X	X	X
Monitoring and Assessing Changes	X	X	X				X			X	X	X	X	X	X	X
Addressing Requirements and Solution Options	X		X						X	X		X	X	X	X	X
Orchestrating Upgrades to SoS				X	X	X	X	X		X	X	X	X	X	X	X

The SE activities marked by "x" in Table 3-1 are not all inclusive, and depending on the specific SoS and constituent systems, the SoS team should tailor their SE processes to their SoS needs for each SoS engineering core element.

In general, the technical management processes are more heavily represented in the SoS engineering core elements, reflecting the SoS systems engineer's role of coordination and orchestration across systems, with detailed engineering implementation taking place primarily at the system level.

The mappings in Table 3-1 are based on a close reading of the technical management process definitions in the SE Guidebook as they apply to SoS engineering. Not all SoS are the same, and it is impossible to list all SE actionable activities to implement in a core element and in sequential order. Each SoS, composed of unique constituent systems, requires its own approach to management and integration of the constituent systems. Each of the seven core elements are further described in sections 3.2.1 through 3.2.7 along with their typical recommended SE process activities.

3.2.1 Translating Capability Objectives

When an SoS need is first identified, the SoS engineering team is called upon to understand and articulate the technical-level expectations for the SoS. SoS objectives are typically couched in terms of needed capabilities, and the SoS engineering team is responsible for working with the SoS manager and users to translate these into high-level requirements that can provide the foundation for technical planning.

To accomplish the SoS objectives, the SoS engineering team needs to understand the nature and the dynamics of the SoS both to appreciate the context for SoS expectations and to anticipate areas of the SoS that are most likely to vary in implementation and change over time. The SoS engineering team has a continuous active role in this ongoing process of translating capability needs into technical requirements and identifying new needs as the situation changes and the SoS evolves.

Because objectives are often stated in terms of broad capabilities, the SoS systems engineer and manager review objectives and expectations on a regular basis as the SoS evolves and changes occur in user needs, the technical and threat environments, and other areas. This activity should be captured through MBSE so the SoS engineering team can have a common structured basis for developing all aspects of the SoS. This approach is consistent with DoW digital engineering efforts to modernize SE for automation and reuse of system models. The most desired aspect of this process would be to have a federated set of MBSE models representing the constituent systems overlayed by the SoS integrated MBSE activity.

Once the SoS establishes the capability objective (often based upon desired operational tasks and missions), the SoS engineering team defines the functions required to provide the capability and the variability in the user environment that will impact the different ways these functions will be executed. The articulation of objectives may be somewhat general at first, but as the SoS engineering processes mature, the objectives become more focused, and they may change. This activity provides the SoS engineering team with a broader understanding of priority and relationship of objectives, which will be useful in the further development and management of requirements.

SoS and their constituent systems must operate in a changing requirements and acquisition landscape. The November 7, 2025 SecWar memorandum, "Reforming the Joint Requirements Process to Accelerate Fielding of Warfighting Capabilities", dissolves the Joint Capabilities Development and Integration System and pushes validation of Component-level requirements back to the Military Services. At the joint level, the Joint Requirements Oversight Council will focus on identifying and annually ranking Joint Force Key Operational Problems and associated capability gaps that flow from the National Defense Strategy. The November 7, 2025 SecWar memorandum, "Transforming the Defense Acquisition System into the Warfighting Acquisition System to Accelerate Fielding of Urgently Needed Capabilities to Our Warriors", emphasizes the timely and urgent delivery of operational capabilities to the Warfighter – "prioritizing flexible requirements and resource trades to enable timely delivery at the speed of relevance". Acquisition reform also empowers Program Acquisition Executives to operate with expanded authorities for resourcing, requirements, and acquisition decisions within prescribed portfolios, allowing for more rapid trades and priority shifts within assigned portfolios. In keeping with these mandates for rapid development and greater requirements flexibility (trades), SoS managers and engineers will need to be particularly attentive to changing capability objectives, requirements management, and requirements analysis.

Figure 3-3 depicts the core element *Translating Capability Objectives* input/output exchanges with external sources and other core elements. The SoS engineering team defines SoS capability objectives based on inputs from:

- External Sources – stakeholder needs, threat conditions, and national priorities.
- *Developing and Evolving SoS Architecture* core element – design feasibility.
- *Assessing Performance Against Capability Objectives* core element – user needs based on operational feedback.
- *Understanding Systems and Relationships* core element – status of systems, relationships, and functionality.

Output: The *Translating Capability Objectives* core element provides information about the first-order goals and expectations of the SoS to the *Assessing Performance against Capability Objectives*, *Developing & Evolving SoS Architecture, Understanding Systems and Relationships,* and *Monitoring and Assessing Changes* core elements. The information helps the SoS engineering team ensure that constituent system requirements align with SoS capabilities. This information also gives the team a broader understanding of objectives, priorities, and relationships, which will be useful in the further development and management of requirements.

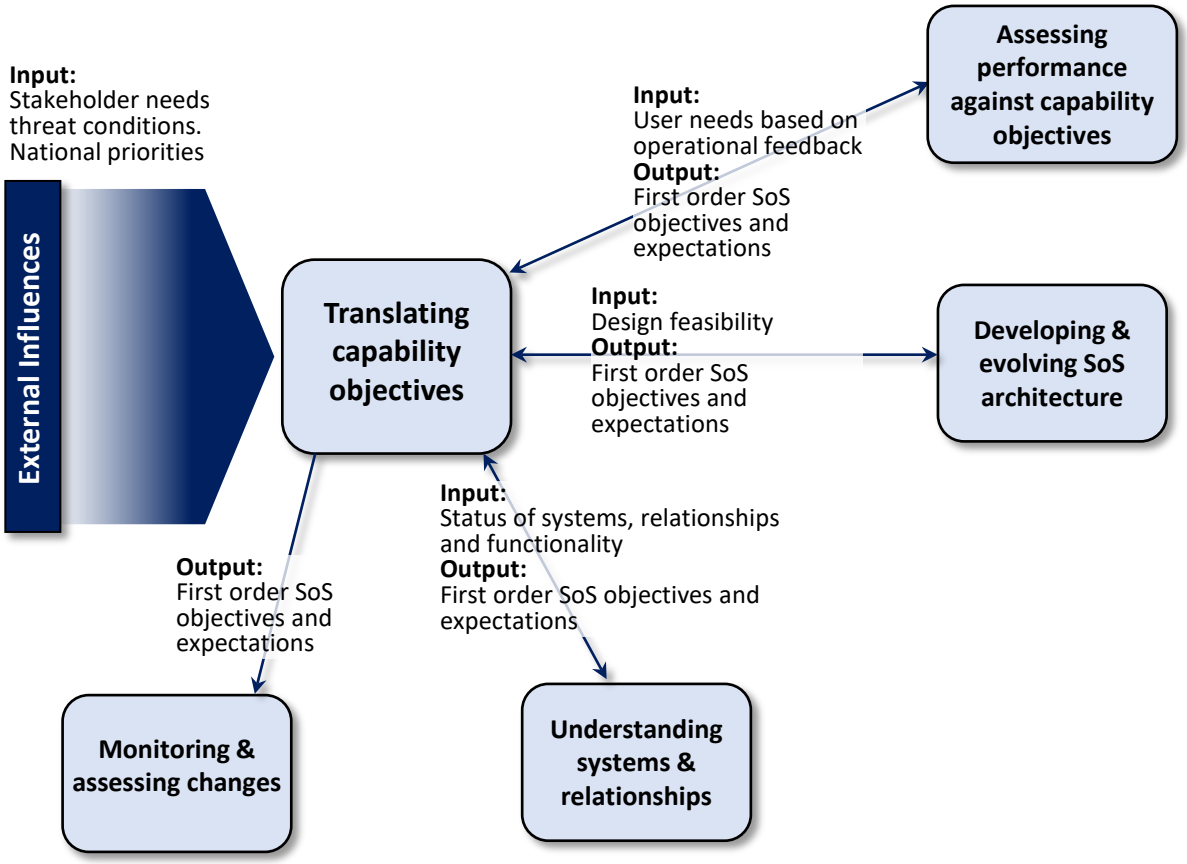


Figure 3-3. Relationship between Translating Capability Objectives and Other SoS Engineering Core Elements

In this element, there should be no explicit consideration of a specific solution (systems or architectures). The capability objectives should be independent of solutions. In practice, existing systems are typically present at the time the SoS is recognized, potentially narrowing the scope of perceived solutions; however, it is important to clarify the capability needs and expectations through SE technical and technical management processes so that over time the SoS systems engineer can consider a range of options to meet capability needs. This consideration may include ways of meeting the needs with a single system or multiple systems. Once the SoS systems engineer begins to review how the objectives can be met by applying SE technical and technical management processes, the program may adjust the objectives given the feasible implementation options.

SE Process Activities in Translating Capability Objectives

The following are recommended SoS engineering team activities for each of the SE processes as they pertain to the *Translating Capabilities Objectives* core element. These are not all inclusive, and depending on the specific SoS and constituent systems, the SoS team should tailor their activities to their needs.

For **Stakeholder Requirements Definition,** the SoS engineering team:

- Translates the SoS stakeholders' capability objectives into high-level SoS technical requirements.
- Conducts trade-offs and negotiates, prioritizes, and develops requirements with SoS stakeholders to balance competing interests and ensure the SoS capability objectives will be met.
- With the SoS stakeholders, reviews and confirms that the SoS capability objectives are correctly translated into high-level requirements and can be further translated to constituent systems, and captures these requirements in the SoS functional baseline.
- With the SoS stakeholders, reviews and updates the SoS requirements regularly to ensure they remain aligned with changing capability objectives.

For **Technical Planning,** the SoS engineering team:

- Assigns a lead SoS systems engineer who is responsible for working with systems engineers of constituent systems to find an SoS solution that provides the needed capability objectives while respecting the different and sometimes conflicting interests of constituent systems.

- Develops a technical plan (SEP or EEMP) to describe the scope of the technical effort required for SoS development. The plan should describe how SE processes will be implemented to support the seven SoS engineering core elements.

For **Requirements Management,** the SoS engineering team:

- Uses established requirements management processes that accommodate traceability and allocation of SoS requirements to constituent systems. The processes should enable the creation of requirements baselines and accommodate bi-directional tracking of requirements information – SoS to constituent systems and constituent systems to SoS. However, constituent systems engineers manage individual systems' requirements.
- Initiates documentation and tracking of requirements through requirements management processes.
- Uses Requirements Traceability Matrix (RTM) to maintain requirements traceability and ensure capability objectives are aligned with all critical stakeholders' needs.
- Conducts requirements reviews with the CTC as needed and updates requirements regularly to ensure they remain aligned with changing capability objectives.

For **Risk Management,** the SoS engineering team:

- Develops an SoS risk management plan that incorporates risks associated with SoS capability objectives into risk management.
- Identifies potential risks that could impede the achievement of capability objectives and incorporates them into the risk management plan. This involves analyzing the complexity of interactions in an SoS environment, where risks may arise from system failures, operational inconsistencies, misalignment of objectives across constituent systems, or unforeseen interdependencies.
- Assesses the identified risks in terms of their likelihood and impact on achieving the capability objectives. In SoS engineering, this might involve evaluating how the failure of one system or component could affect other systems and overall SoS performance.
- Develops strategies (e.g., accepting, avoiding, transferring, controlling) to mitigate identified risks to reduce the probability of their occurrence or minimize their impact on SoS capability objectives. Mitigations might include building in redundancies or fail-safes, or designing systems with the flexibility to adapt to changing conditions.
- Monitors the risk mitigation plan to track and evaluate SoS performance against established SoS metrics.

For **Configuration Management,** the SoS engineering team:

- Initiates SoS Configuration Management activities. As objectives are captured and high-level requirements for the SoS are defined and evolved, these high-level requirements must be captured and managed because eventually they may be incorporated into SoS baselines.
- Evaluates any modifications or changes in system components or configurations for their impact on the broader capability objectives. This ensures that new capabilities or adjustments do not inadvertently disrupt other parts of the SoS.

For **Data Management**, the SoS engineering team:

- Initiates a data management approach and knowledge base to support SoS development.
- Identifies data and information required to accomplish SoS objectives between systems.
- Captures data on SoS objectives, expectations, and high-level requirements.

3.2.2 Understanding Systems and Relationships

One of the most important aspects of the SoS systems engineer role is developing an understanding of the systems involved in providing the needed SoS capabilities and their relationships and interdependencies as part of the SoS. In an individual system acquisition, the systems engineer is typically able to clearly establish boundaries and interfaces for a new system.

In an SoS, understanding the functionality of each system is the basis for understanding (1) how the systems support the SoS objectives, (2) technical details of the systems pertinent to the SoS (e.g., approaches to sharing or exchanging mission information), and (3) the current system development plans including timing and synchronization considerations. An SoS systems engineer also needs to identify the stakeholder and users, their relationships and organizational contexts, and their drivers so that options for addressing SoS objectives can be identified and evaluated and impacts of changes over time can be anticipated and addressed. Also, key systems can be outside of the direct control of the SoS management but have large impacts on the SoS objectives, and it may not be possible to identify them all.

Boundaries and interfaces in an individual system acquisition are less subject to change during an increment of system development, and these are typically defined and documented in a relationship document (e.g., Interface Control Document, Interface Control Standard, etc.). In an SoS, the boundaries and interfaces are more dynamic; the constituent systems may interact with one or more of the other systems at different times to achieve SoS capabilities. In addition, constituent systems may undergo changes that affect SoS boundaries and interfaces. The importance of interfaces in an SoS is that they enable access to SoS behavior. Within an SoS, the behavior involves the combined interactions, including processes and data flow, within and across the constituent systems that create the behavior and performance of the SoS. Interfaces are therefore critical to successful SoS systems engineering.

Understanding Systems and Relationships involves addressing several different types of relationships. Typically, in this area, a first step is defining the functionality of the systems and how they share data and resources during operations. These are certainly major areas of concern for the SoS systems engineer; however, because of the characteristics of an SoS, other relationships are also important. Each of the following examples illustrates some of these relationships:

- Organizational relationships among the systems (who is responsible for management and oversight of the systems?)
- Stakeholders, including users of SoS and constituent systems, including their organizational context as a foundation, for their role in the SoS
- Resourcing relationships (Who is responsible for funding which aspects of the systems and how are they related to the SoS funding authorities?)
- Requirements (What is the relationship between the requirements of the constituent systems and SoS?)
- Relationships among the development processes and plans of the constituent systems and the SoS (waterfall, incremental, agile development approaches, timing, and scheduled events). This also includes the SoS architecture and the strategy of migration to that architecture over time should consist of a set of as-is and to-be models along with transition plans.

Understanding Systems and Relationships also provides integrated knowledge and data within the SoS environment, including linkages to data maintained by the systems relevant to the SoS. This element considers both those systems under direct responsibility of the SoS manager and those that are outside the manager's immediate span of control, which will require influence through collaboration and the establishment of common goals.

Importantly, *Understanding Systems and Relationships* provides the basis for identifying where formal and informal working agreements are required. They are the basis for understanding primary areas of focus, that is, instances in which SoS functionality and performance are affected by changes in systems. Because SoS in the DoW today are not typically supported by standards, basic organizational structures, and processes, the SoS manager and systems engineer must assess when specific working agreements need to be established for the SoS. Some SoS have created types of MOAs or MOUs that formalize the relationships between the SoS and the constituent systems. The MOA or MOU specifies the responsibilities of SoS, system management, SE, and other aspects of their SoS related working relationships. Moreover, if SoS adopts a service-oriented architecture (SOA) approach, they will adopt service-level agreements (SLAs) to specify agreed-upon support to the SoS.

Figure 3-4 depicts *Understanding Systems and Relationships* core element input/output exchanges with other core elements. The SoS engineering team identifies constituent systems functions and establishes data exchange requirements based on *inputs* from:

- *Translating Capability Objectives* core element – first order SoS objectives and expectations.
- *Developing and Evolving SoS Architecture* core element – updated architecture information.
- *Monitoring and Assessing Changes* core element – changes that impact systems and relationships, could be internal (e.g., changes to constituent systems or interfaces) or external (e.g. technology, threat).
- *Orchestrating Upgrades to SoS* engineering core element – upgrades that impact systems and relationships.

Output: The *Understanding Systems and Relationships* core element provides status of systems, relationships and functionality to the *Translating Capability Objectives*, *Developing and Evolving SoS Architecture*, *Addressing Requirements and Solution Options*, and *Monitoring and Assessing Changes* core elements. From the output of this core element, the SoS engineering team can gain a high-level understanding of the ensemble of systems that affect SoS capabilities, and the way they interact and contribute to the overall SoS mission objectives. With this understanding they can develop working agreements between the SoS and constituent systems.

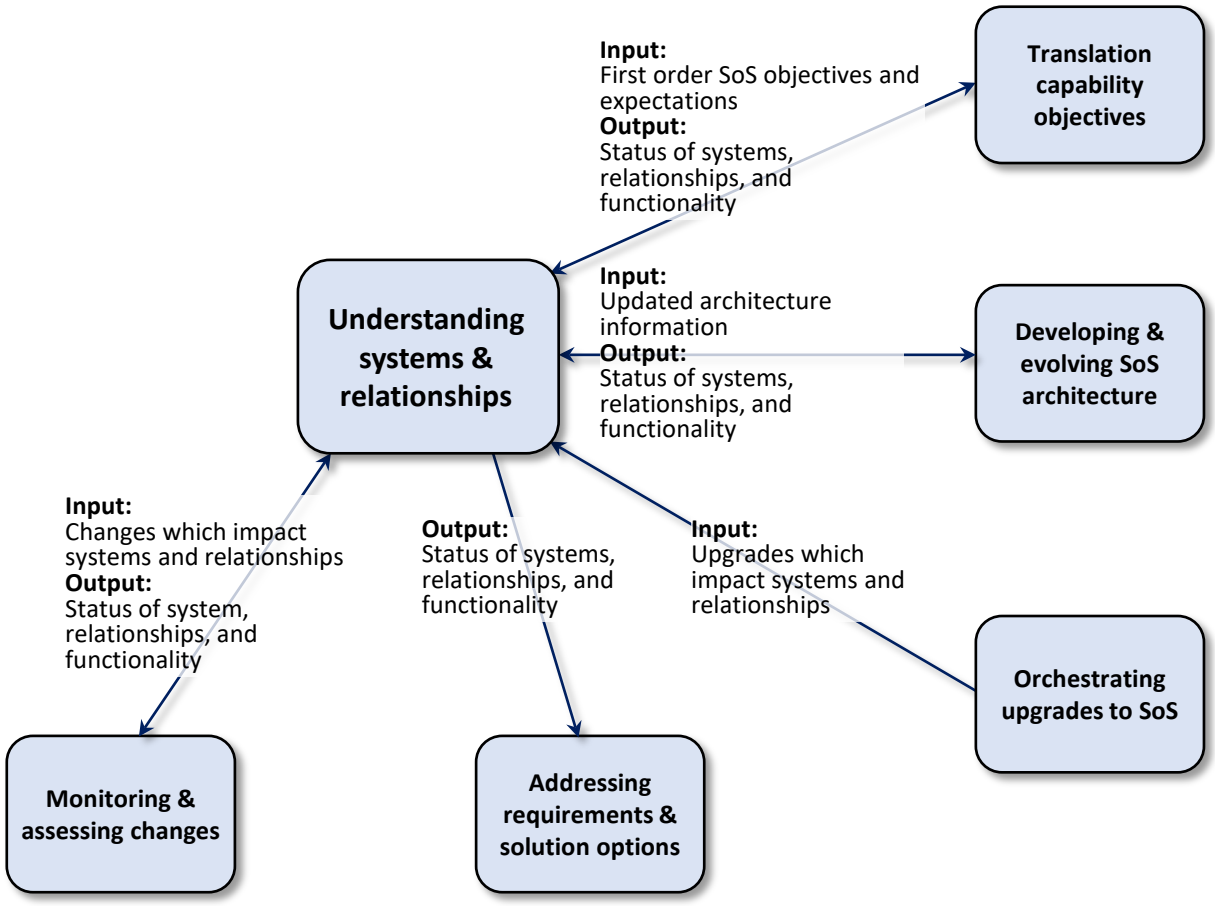


Figure 3-4. Relationship between Understanding Systems and Relationships and Other SoS Engineering Core Elements

SE Process Activities in Understanding Systems and Relationships

The following are recommended SoS engineering team activities for each of the SE processes as they pertain to the *Understanding Systems and Relationships* core element. These are not all inclusive, and depending on the specific SoS and constituent systems, the SoS team should tailor their activities to their SoS needs.

For **Requirements Analysis**, the SoS engineering team:

- Identifies the constituent systems that (currently) make up the SoS.
- Understands:
  - How each constituent system supports SoS objectives and requirements.
  - The technical details of each constituent system.
  - Current development plans of each constituent system.
  - The stakeholders and users of each constituent system.

- Analyzes constituent systems requirements, assesses their impacts on the overall capability objectives of the SoS, and translates the results into specific SoS design requirements and constraints.
- Monitors relationship changes between constituent systems in the SoS (relationships between constituent systems and an SoS are often dynamic and can evolve) and conducts requirements analysis as needed.
- Identifies functionality and attributes which are common across the SoS and assesses the current state of the SoS with respect to these cross-cutting requirements.

For **Risk Management**, the SoS engineering team:

- Manages risks including:
  - Unanticipated effects of different implementations of functionality needed in a core thread for the SoS.
  - Changes in functionality in constituent systems due to new and conflicting needs of the system users.
  - Limited capacity in systems in view of unknown SoS demand.
  - Technical constraints within systems that affect their ability to adapt to changes needed by the SoS.
  - System owners' resistance to implementing the changes needed by the SoS, because of competing priorities for funds, development time, or technical staff.
- Identifies and analyzes potential risks early in the SoS design phase by understanding the complexities of constituent systems interactions that arise from the diversity of system types (e.g., hardware, software, human-in-the-loop), operational environments, and the need for interoperability.
- Develops a risk mitigation plan and monitors and tracks the burn-down history of the risks to constituent systems and their relationships to support the SoS capability objectives. In coordination with the established RMB, they determine whether the risk management process is having the desired effect reducing the likelihood or impact of the risks on the SoS and constituent systems.

For **Configuration Management**, the SoS engineering team:

- Implements CM process for the "as is" SoS, which is maintained as the SoS product baseline.
  - In an SoS, the ensemble of systems, their functionality, and their relationships (e.g., data exchanges) is the "product" to be put under configuration management at the

SoS level. SoS CM also addresses any cross-cutting attributes that pertain to the SoS but are not addressed by the CM of any constituent system.

  - The constituent systems maintain detailed CM of the systems. SoS configuration managers have access to constituent system CM information and link to it as necessary.

- Uses CM tools to map the SoS product baseline.
- Tracks and ensures, through CM version control processes, that changes to one system's configuration do not disrupt the overall SoS. This activity provides traceability of changes and supports rollback or recovery when needed.
- Reviews the status of configuration controls with the established CCB to determine if the controls put in place by the SoS CM process are having the desired effect.

For **Data Management**, the SoS engineering team:

- Implements data management process to identify data about:
  - Functionality in systems.
  - Relationships among systems, including interfaces for real-time data exchange, organizational relationships, development plans, etc., to which common or cross-cutting attributes are present across system.
- Ensures that the data management process enables the capture of comprehensive data (both technical and operational) from all constituent systems, and ensures the data's accuracy, integrity, integration, and availability across the SoS. Without robust data management, there would be gaps in understanding the current state and progress toward achieving effective relationships.
- Ensures interoperability between systems by establishing common data standards and exchange protocols. An SoS typically consists of different systems that may be from different domains or organizations. Thus, interoperability is essential for the timely exchange of data required for decision making and tracking of capability objectives. Interoperability between SoS and constituent systems must be tested and verified and corrective action taken.
- Ensures seamless access and exchange of data between constituent systems. Implements strategic use of application programming interfaces (APIs), which allow constituent systems to exchange information in a data-centric mode (Appendix C) instead of the less dynamic, net-centric structures of the past. Use of APIs negate the cumbersome, net-centric point-to-point communication structure, where each data exchange point required unique design activities.

- Data is a product as much as any piece of hardware. Emphasis should be equally placed on its creation, use, maintenance, and disposal.

For **Interface Management**, the SoS engineering team:

- Identifies, defines, and documents the internal and external interfaces and dependencies between constituent systems and the SoS to clearly understand the relationships in terms of the data exchanges among systems.
- Defines interface requirement specifications that fully describe the physical and functional interfaces between the SoS and constituent systems to ensure compatibility between interfacing components, systems, and platforms.
- Facilitates cooperation and agreement among all stakeholders to establish interfaces and boundaries for constituent systems and the SoS.
- Monitors changes on internal and external interfaces and the dependencies between constituent systems. The team reviews changes with SoS managers, stakeholders, and constituent system owners through an established interface management process or review board.

3.2.3 Assessing SoS Performance Against Capability Objectives

In an SoS, T&E may be conducted at two levels: Developmental Test and Evaluation (DT&E) and Operational Test and Evaluation (OT&E). DT&E is an early integrator and must be planned as the SoS is designed and engineered. *Assessing SoS Performance Against Capability Objectives* relies on OT&E and focuses on developing metrics and collecting data from a variety of settings over time to monitor the performance of the SoS with respect to the user objectives. The assessments can take a variety of forms, including analysis, demonstration, and inspection. From an SE perspective, the results inform different elements of SoS engineering on changes in performance and in other areas, which may affect the SoS. DT&E activities occur during the development and implementation of constituent system upgrades and can be conducted during large-scale exercises and experiments to verify the technical performance of the SoS. Assessments should include assessing performance in a representative, contested cyber environment to validate operational resilience.

In the *Assessing Performance Against Capability Objectives* core element, the SoS engineering team works with the T&E community to establish technical performance measures and methods for assessing the overall performance of the SoS. At the SoS level, performance is measured in terms of capability objectives, focusing on the utility of the SoS capability to the user. Therefore, these metrics should measure the intended integrated behavior and performance of the SoS in actual operations, rather than just SoS development progress. Furthermore, these "external" user-oriented measures of SoS ("Is it meeting the capability objectives?") should be applicable across implementation or operational environments. Because the SoS is typically operating in the field or in exercise environments, these offer opportunities to collect and analyze data on SoS performance to support SoS-level SE as well as other management decisions. These metrics are akin to the user-oriented Key Performance Parameters (KPPs) for an acquisition program as applied at the SoS level. Often when an SoS is first acknowledged, one of the first actions by the SoS engineering team is to collect performance information as the starting point for identifying areas of the SoS that need attention.

Because SoS typically comprise existing systems, data from operations is an important source of understanding the state of the SoS. Since the SoS will typically evolve based on incremental changes in individual systems, it is important to have a set of user-oriented metrics that can be applied in different settings over time. The SoS engineering team uses data from these settings to analyze SoS performance and behavior; hence, the metrics should include measures that use data from operations.

These SoS metrics should be traceable to the capability objectives established for the SoS, and the program needs to rank the metrics by importance. These metrics should not change as the capability of the SoS matures unless the capability objectives themselves change. They must remain applicable as the SoS matures to assess whether the changes made are translating into better user support.

When captured in an operational environment, metrics enable independent assessments of SoS performance from the user's perspective. Metrics also allow assessment of the impact of external factors on capability objectives. These operational user-based performance assessments, focused on SoS capability objectives, do not substitute for the technical reviews and assessments performed during the process of upgrading the individual systems in the SoS. These activities are discussed in section 3.2.7, Orchestrating Upgrades to SoS.

Data from these operational venues also can be used to identify unanticipated external changes that affect SoS performance and therefore need to be factored into SoS engineering. Because of the complexity of many SoS, when changes made to support SoS objectives are introduced into an operational environment, unexpected interactions are not uncommon. Therefore, SoS T&E needs to identify and assess the additional capability provided by these unexpected interactions. This includes the need to consider potential harmful interactions between systems, and how those interactions can be identified and managed. Just as important, T&E can address new and unexpected capabilities that result from SoS. Users often find new ways to employ new functionality; by watching those creative adaptations, systems engineers can often discover new categories of functionality that will further aid users. In short, systems and their users evolve synchronously, and it is important to be aware and adapt the SoS to leverage these changes.

Figure 3-5 depicts the *Assessing Performance Against Capability Objectives* core element input/output exchanges with other core elements. The SoS engineering team establishes metrics and methods for assessing performance of the SoS capabilities by identifying the most important mission threads and assessing end-to-end performance in an operational environment. The SoS engineering team accomplishes this, based on inputs from:

- *External Influences* – external factors that may affect the performance of the SoS.
- *Translating Capability Objectives* core element – first-order goals and objectives, which serve as the basis for the metrics and assessment approach.
- *Orchestrating Upgrades to SoS* core element – changes to SoS with objectives of improved capabilities for the user.

Output: The *Assessing Performance Against Capability Objectives* core element provides feedback on implementation variability to the *Developing and Evolving SoS Architecture* core element, user needs based on operational feedback to the *Translating Capability Objectives* core element, and factors affecting capability; and feedback on user behavior to the *Monitoring and Assessing Changes* core element. This feedback also includes unanticipated changes in the way that users employ the SoS, which may need to be considered in planning for SoS evolution.

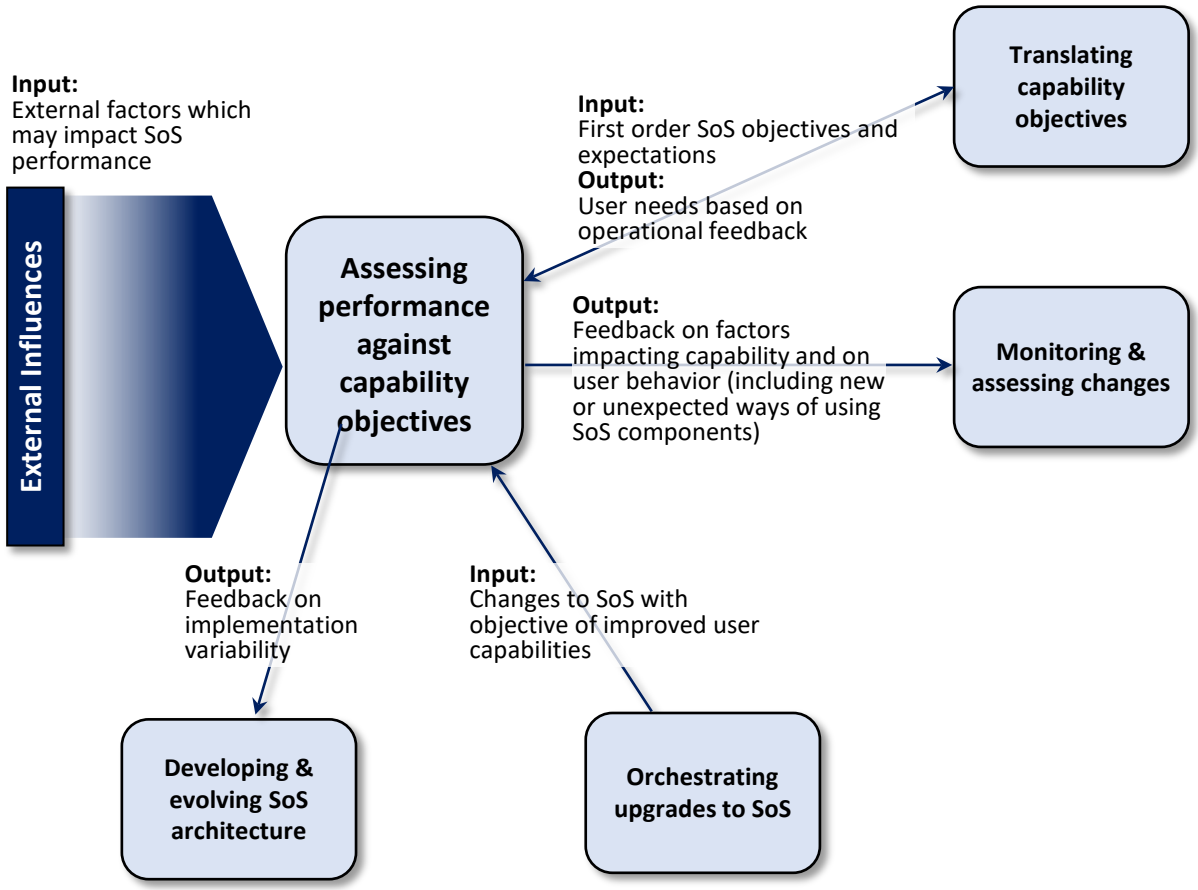


Figure 3-5. Relationship between Assessing Performance Against Capability Objectives and Other SoS Engineering Core Elements

SE Process Activities in Assessing Performance Against Capability Objectives

The following are recommended SoS engineering team activities for each of the SE processes as they pertain to *Assessing Performance Against Capability Objectives* core element. These are not all inclusive, and depending on the specific SoS and constituent systems, the SoS team should tailor their activities to their SoS needs.

For **Validation,** the SoS engineering team:

- Defines the scope of what will be validated and the validation actions (strategies, methods, and success criteria) to assess performance against capability objectives of an SoS.
- Executes validation of SoS performance against capability objectives using established metrics and procedures to ensure constituent systems and the SoS operational functionalities meet stakeholder and user needs.

- Analyzes validation results against established validation success criteria to ensure SoS capabilities meet stakeholder and user needs.

For **Decision Analysis,** the SoS engineering team:

- Identifies decisions relative to SoS performance that need to be made (e.g., major technical alternative selection).
- Develops SoS decision criteria, criteria weight, and measures to evolve the ensemble of systems to meet new needs while accommodating the independently owned and funded constituent systems.
- Conducts analysis to address the following questions relative to the decisions that need to be made: Are the right metrics/indicators being collected? In the right venues? At the right points? Will you have the right information available to inform the decisions?
- Identifies, records, and tracks any assumptions made during decision analysis.
- Performs sensitivity analysis and uses tools to conduct decision analysis on SoS performance against its capability objectives to generate alternatives, assess alternatives, identify uncertainties, assess the impact of uncertainties, and communicate trade-offs.
- Presents analysis, recommendations, and action plans to decision makers.

For **Technical Assessment**, the SoS engineering team:

- Identifies measures and metrics for assessing performance of the SoS capabilities that are independent of alternative SoS implementation approaches.
- At a minimum, focuses assessment on the end-to-end performance of the most important SoS mission threads.
- Conducts a technical assessment of the SoS performance against its capability objectives to ensure implemented changes in the constituent systems directed at improving SoS performance meet their intended high-level goals and mission objectives and documents the assessment results. This includes identifying and verifying core capability objectives such as situational awareness, operational effectiveness, or strategic agility, as well as changes proposed to improve SoS performance.
- Includes operational feedback in assessments.
- Continuously assesses the progress of implemented changes aimed toward improving SoS performance to achieve desired capability objectives.
- Reviews the results of these assessments with decision makers, SoS managers and stakeholders, systems engineers for the constituent systems, and SMEs.

For **Requirements Management,** the SoS engineering team:

- Uses the traceability of requirements information between the SoS and constituent systems to track validation results to constituent systems in case the SoS performance fails the validation.
- Regularly reviews requirements to ensure they remain aligned with changing capability objectives for the purpose of validation traceability.

For **Risk Management,** the SoS engineering team:

- Develops a risk management plan to identify, analyze, mitigate, and monitor those risks related to the ability of the SoS to achieve its desired performance against its capability objectives.
- Identifies potential risks in terms of their likelihood and impact on SoS performance to support its capability objectives. This involves analyzing the interactions in a complex SoS environment, where risks may arise from misalignment of capability objectives across constituent systems.
    - Includes, where possible, risk assessment in validation and technical assessment activities to provide more definitive information on risk likelihood and impact.
- Develops strategies (accepting, avoiding, transferring, controlling) to mitigate identified risks to reduce the probability of their occurrence or minimize their negative impact on SoS performance. The team also monitors and tracks the risk mitigation plan and evaluates the resulting SoS performance toward achieving its capability objectives.
- Documents results of the SoS performance risk assessment, which includes implemented changes in the constituent systems directed at improving the SoS performance to achieve its capability objectives. These results are reviewed with and agreed upon by SoS and constituent system managers.

For **Data Management,** the SoS engineering team:

- Identifies the SoS data requirements, which include the characteristics of the assessment venue (the players, the scenarios, the state of the systems and SoS at the time of the event), interfaces for real-time data exchange, organizational relationships, and development plans. The data management process includes the measurement and analysis approaches for data collected and results obtained.
- Ensures that the established data management process enables the capture of comprehensive data (both technical and operational) from all constituent systems, and ensures the data's accuracy, integrity, integration, and availability across the SoS.

Without robust data management, there would be gaps in understanding the current state of SoS performance and progress toward achieving its capability objectives.

- Uses an API strategy that allows constituent systems to exchange information in a data-centric mode instead of the less dynamic net-centric structures of the past.
- Assesses that constituent systems within an SoS are interoperable by using common data standards and exchange protocol diagnostic tools to ensure SoS performance supports capability objectives.
- Analyzes available operational data in real time to identify SoS performance gaps in achieving desired SoS capability objectives. Effective data management consists of an infrastructure for data analytics, which can be used to evaluate whether the SoS performance is achieving its capability objectives.

3.2.4 Developing and Evolving an SoS Architecture

Once an SoS systems engineer has clarified the high-level technical objectives of the SoS, identified the systems that are key to SoS objectives, and defined the current performance of the SoS, an architecture overlay for the SoS is developed, beginning with the existing or de facto architecture of the SoS. The architecture of an SoS addresses the Concept of Operations (CONOPS) for the SoS and encompasses the functions, relationships, and dependencies of constituent systems, both internal and external. This includes end-to-end functionality, data flow, and communications.

The architecture of the SoS provides the technical framework for assessing changes needed in systems or other options for addressing requirements. In the case of a new system development, the SE team can begin with a fresh, unencumbered approach to architecture; however, in an SoS, the systems contributing to the SoS objectives are typically in place when the SoS is established, and the SoS engineering team needs to consider the current state and plans of the individual systems as important factors in developing an architecture for the SoS. In developing the architecture, the SoS engineering team identifies options and trades and provides feedback when there are barriers to achieving balance between the SoS and constituent system's needs and constraints.

To develop this architecture in a manner that is rigorous, analyzable, and maintainable, the SoS engineering team must move beyond simple drawings and employ a formal modeling language. SysML provides the standard, integrated toolset required for this task. The architecture should be captured in a cohesive model using a combination of diagrams. Using a multi-diagram approach ensures that the SoS architecture is a living, digital model that can be evolved and analyzed over time, rather than a static set of documents.

Developing an architecture requires analysis and assessments of trades among different options. Architecture analysis may be supported by different assessment approaches. For long-term viability, architecture developments are served well by up-front analyses to explore sensitivities through modeling, simulation, analysis, and/or lab experimentation to identify scalability issues or knees in the curve (e.g., concerning requirements or usage assumptions, assumed network bandwidth, or others) beyond which performance starts to break down. This type of analysis provides a basis for the architecture decisions. Similarly, development of metrics for assessment of SoS performance and maturity provides a basis for both developing an architecture and assessing it over time.

Focused investigations of functionality and relationships may be conducted to address core issues. For example, it may be important to assess the effect of multiple systems working together under controlled conditions to understand underlying processes that will affect the SoS behavior. These analyses often employ modeling and simulation (M&S).

The architecture of an SoS is somewhat constrained by the structure and content of the individual systems, particularly the extent to which changes in those systems are affordable and feasible, since systems will typically need to continue to function in other settings in parallel with participation in the SoS. The functionality that the individual systems contribute to the SoS can be described in a functional architecture that puts the key functions in order, thereby sequencing the SoS tasks. The functional architecture provides a functional "picture" of the SoS. It details the complete set of functions to be performed within the SoS as well as the relationships among the functions. The variability in the execution of these functions in the field also needs to be understood and factored into the SoS architecture (Boxer, 2008).

Ideally, the architecture of an SoS will persist over multiple increments of SoS development, allowing for change in some areas while providing stability in others. This ability to persist and provide a useful framework considering changes is a core characteristic of a good architecture. Over time, the SoS will face changes from several sources (e.g., capability objectives, actual user experience, changing CONOPS and technology, and unanticipated changes in systems), which may all affect the viability of the architecture and may call for changes. Consequently, the SoS engineering team needs to regularly assess the architecture to ensure that it supports the SoS evolution.

In most cases, because of the nature of SoS as an overlay on multiple existing systems, the development of an architecture of an SoS will be incremental. In some situations, the first step in an SoS evolution is to improve the way the SoS functions without making any explicit architecture changes. Only then, based on this experience, will the SoS engineering team develop an architecture that can be implemented over time. The architecture of an SoS will evolve and mature over time through the result of technical reviews at the SoS level and through changes to specific systems, aimed at increasing the capability of the SoS.

The development and implementation of an SoS architecture may be significantly constrained by a reluctance to invest in the constituent systems, which in many cases are very mature (e.g., in sustainment), to support the SoS. In this case, approaches such as gateways and "wrapping" (e.g., APIs) may be used to incorporate these systems into the SoS without making significant changes in these systems.

The constituent systems that compose an SoS will continue to face evolving functional requirements and the need for technology upgrades independent of the SoS. This creates inherent tension between the needs of individual systems and the demands of the SoS. To address this, an SoS architecture based on a MOSA and loose coupling offers significant advantages. MOSA provides constituent systems with the flexibility to adapt to changing user needs and incorporate the most suitable technologies without disrupting the SoS.

From the SoS perspective, architectural decisions should prioritize extensibility, flexibility, and persistence over time, enabling the addition, deletion, or modification of systems without adversely affecting other components or the SoS as a whole. MOSA implementations leverage open architectures with severable modules and well-defined interfaces. These modules are designed for specific functions, allowing one module (or system) to replace another while maintaining functionality. Interfaces, based on open, consensus-driven standards, define the protocols for data, signal, and command exchange and resource, facilitating interoperability across modules or systems, even when modified or replaced. Where interfaces differ, efficient translators, adapters, mappings, or other objective integration approaches may be more appropriate. SoS engineers should develop and share reference architectures that incorporate severable modules and well-defined interfaces. These reference architectures provide constituent systems or potential constituent systems with a blueprint for integrating into the SoS.

By carefully defining standard interfaces and modular components, organizations can create a cohesive, scalable, and adaptable SoS architecture. It is particularly valuable in large, multi-system SoS in which constituent systems are managed by different authorities and compliance with modularity can ensure continued SoS functionality. While not all constituent systems may initially support open architectures, transitioning to modular, open designs over time is both a typical and desirable strategy for SoS development. More engineering guidance on MOSA can be found on the SE&A website (https://www.cto.mil/sea/) library and MOSA pages. MOSA-enabling standards information can be found in ASSIST (https://assist.dla.mil/).

Figure 3-6 depicts the *Developing and Evolving SoS Architecture* core element input/output exchanges with the other core elements. The SoS engineering team develops the architecture of an SoS based on CONOPS for the SoS that include functions, relationships, and internal/external dependencies of constituent systems based on inputs from:

- *Translating Capability Objectives* core element – capability objectives for the SoS, including changes.
- *Understanding Systems and Relationships* core element – current systems functionality and technical interfaces.
- *Assessing Performance against Capability Objectives* core element – performance measures/goals related to the SoS architecture, end-to-end functionality, data flow, and communications.
- *Addressing Requirements and Solution Options* core element – feedback from the implementation on aspects of the architecture that may need to be adjusted.

Output: The *Developing and Evolving SoS Architecture* core element provides the persistent framework for assessing options for meeting SoS requirements to the *Addressing Requirements and Solution Options* core element, feedback on the design feasibility of achieving the desire SoS objectives to the *Translating Capability Objectives* core element, and feedback on SoS architecture implementation variability to the *Assessing Performance against Capability Objectives* core element.

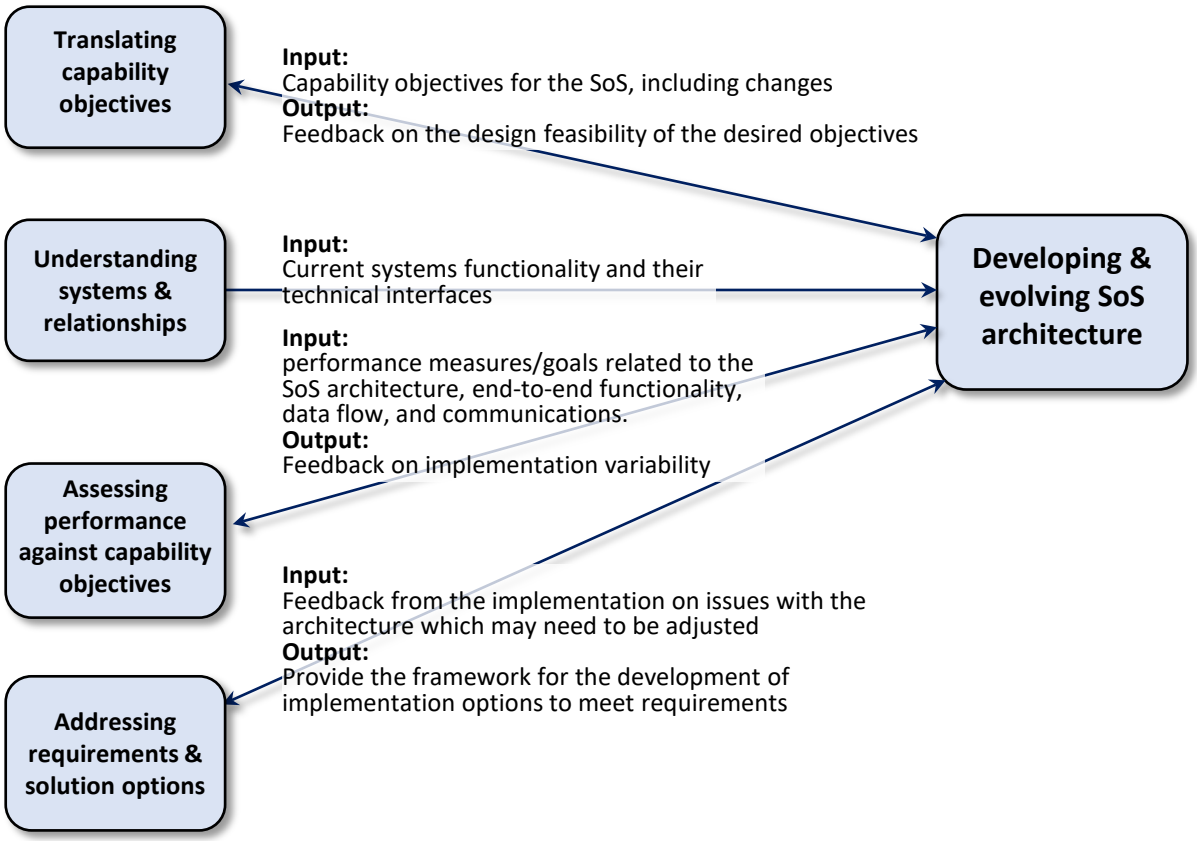


Figure 3-6. Relationship between Developing an SoS Architecture and Other SoS Engineering Core Elements

SE Process Activities in Developing and Evolving an SoS Architecture

The following are recommended SoS engineering team activities for each of the SE processes as they pertain to the *Developing and Evolving an SoS Architecture* core element. These activities are not all inclusive, and depending on the specific SoS and constituent systems, the SoS team should tailor their activities to their SoS needs.

For **Stakeholder Requirements Definition**, the SoS engineering team:

- Uses the SoS stakeholders' needs and defines requirements for the development of an SoS architecture to ensure the SoS capability objectives are achieved. Defining SoS stakeholders' requirements include specifying constituent systems boundaries to reflect operational scenarios and expected system behaviors in the SoS architecture.
- Negotiates, prioritizes, and develops requirements trade-offs with SoS stakeholders to balance competing interests and ensures the SoS architecture supports critical mission objectives.

- Reviews with SoS stakeholders' and confirms that the SoS capability objectives are correctly defined into SoS high-level requirements for the SoS architecture, which can be incorporated into an SoS functional baseline.
- Updates the SoS requirements to ensure they remain aligned with the SoS architecture mission objectives. The SoS architecture may change as the SoS matures or due to changes in mission, technology, threat, etc.

For **Requirements Analysis**, the SoS engineering team:

- Develops an overlay onto constituent systems architectures in a structured way, leveraging DoW Architecture Framework and the UAF for the systems to work together; and defines how they will share information and resources to support SoS capability objectives. The team addresses key questions about the SoS, including:
  - Which systems provide what functionality to the SoS?
  - What are the end-to-end mission threads for the SoS?
  - What behavior is expected of the constituent systems?
  - What data needs to be exchanged to implement the threads?
- Allocates functions to constituent systems as the SoS architecture develops and evolves and documents them in the functional baseline through the requirements management process.
- Uses the SoS CONOPS to analyze the constituent systems requirements and the effects of interdependencies on the SoS architecture and uses the results to develop and evolve specific SoS architecture design and functionality.
- Monitors and analyzes changes in constituent systems requirements that could affect the SoS requirements, and ensures these changes are reflected appropriately in the SoS architecture.

For **Architecture Design**, the SoS engineering team:

- Develops the de-facto architecture design framework for the SoS to become more useful and mature over time and evaluates how well it persists in the face of both priority requirements changes and external changes that impact the SoS.
  - To the maximum extent possible, ensures the architecture is modular, open, and flexible so it can evolve to incorporate changes that need to be managed at SoS level.
- Identifies and defines critical attributes of the physical system interfaces and elements and documents the reference architectures and standards that support the SoS architecture design framework to ensure it supports critical SoS mission objectives.

- Manages the SoS architecture design framework, reviews updates, and negotiates agreements with SoS and constituent system managers to ensure critical SoS mission objectives are achieved.

For **Technical Planning**, the SoS engineering team:

- Establishes an architecture development process and carefully coordinates it with constituent systems technical plans to implement improvements to the SoS architecture while the architecture is evolving to meet its mission objectives.
- Plans technical reviews and audits, coordinated with SoS and constituent system managers and other stakeholders and SMEs as required. These reviews and audits include changes implemented in the SoS architecture design framework and any resulting decisions made.

For **Decision Analysis**, the SoS engineering team:

- Develops decision criteria and uses tools to conduct decision analysis on the SoS architecture design framework to generate alternatives, assess alternatives, synthesize results, identify uncertainties, assess the impact of uncertainties, communicate trade-offs, present recommendations to decision makers, and implement the action plan. The criteria need to be carefully considered to balance:
  - Functionality and performance objectives for the SoS.
  - Extensibility and flexibility of the design to accommodate change.
  - The time frame and funding available to the SoS to support changes in constituent systems.
  - Adaptability of constituent systems and SoS to changes.
- Performs sensitivity analysis on the SoS architecture design framework to synthesize alternative results, prepares a decision briefing with action/implementation plan(s), and provides appropriate recommendations to decision makers regarding the impact of alternative results to achieving critical SoS mission objectives.

For **Requirements Management,** the SoS engineering team:

- Captures and manages the SoS requirements and generates requirements for the constituent systems as the SoS architecture develops and evolves.
- As required, reviews identified requirements, requirements analysis results, requirements changes and updates, and requirements traceability with stakeholders.

For **Risk Management**, the SoS engineering team:

- Identifies and analyzes the risks to achieving SoS objectives by considering cross-cutting issues of the SoS architecture design framework, use, functions, implementation, and dependencies. The following are typical risk considerations:
  - Architecture that precludes addressing key functionality or performance requirements.
  - Difficulty harmonizing the data across the SoS.
  - Architecture is too inflexible and needs to be changed with new SoS or constituent system requirements.
  - Systems are unable to adapt to the architecture (due to technical concerns, workload, funding, or unwillingness to change/take on risk).
- Mitigates the identified risks by developing strategies (accepting, avoiding, transferring, controlling, or burn-down) to reduce the probability of their occurrence or minimize their impact on the SoS architecture design framework.
- Documents the results of the risk assessment for the SoS architecture design framework overlay, which includes implemented changes on the constituent systems architecture directed at improving the operational performance of the SoS. The results are reviewed and agreed upon by SoS stakeholders.

For **Configuration Management,** the SoS engineering team:

- Captures the impact of architecture design changes at any time as part of the CM process. The SoS architecture design changes should also be accessible to the SE teams for the constituent systems.
- Identifies the configuration of each constituent system's architecture by maintaining an updated SoS baseline and uses CM tools to help map the interrelationships between constituent systems architecture and the overlaid SoS architecture.
- Tracks and maintains that changes to one system's reference architecture baseline configuration are captured in the SoS reference architecture baseline configuration. This provides traceability of changes and supports rollback or recovery when needed.
- Verifies and audits controls for developing and evolving an SoS architecture with an established CCB; assesses whether the controls put in place by SoS configuration management process are having the desired effect to ensure the proper operation of each constituent system in its operational environment and in context of the SoS operational environment.

For **Data Management**, the SoS engineering team:

- Continually collects data about the SoS architecture through iterations of changes (which may be asynchronous or concurrent). Types of data collected include, but are not limited to:
  - Architecture drivers and trade-offs
  - Architecture description including CONOPS
  - Systems, including functionality and relationships
  - SoS mission threads
  - End-to-end behavior of SoS to meet objectives, including flow and control of information
  - Architecture risks
  - Technical plans for architecture development/implementation
- Ensures that the established data management process enables comprehensive architecture data (both technical and operational data) captured from all constituent systems, and ensures its accuracy, integrity, integration, and availability across the SoS. When possible, the system architecture model can be used and should include the environmental context, including its representation of how the system interacts with the environment.
- Assesses the constituent systems use of data standards and data exchange protocols to determine compatibility with SoS data architecture.

For **Interface Management**, the SoS engineering team:

- Collaborates with the constituent SE teams to define the common or shared data syntax and semantics including system behaviors and actions (information exchange and trigger events) that serve to moderate the collective behavior of the systems in the SoS.
- Facilitates cooperation and agreement among all stakeholders on individual system interface changes to integrate into the SoS architecture and performs updates to interface specifications to capture the changes.
- Reviews technical updates regularly with stakeholders through an established SoS-level interface working group, which serves as the communication link among those responsible for interfacing systems, end products, enabling products, and subsystems within the SoS architecture.

- Updates SoS architecture models to capture internal and external interface changes and dependencies as part of the enterprise architecture and continuously monitors new changes to ensure the changes are properly implemented in the architecture.

3.2.5 Monitoring and Assessing Changes

A core activity of SoS engineering is to anticipate changes outside the control of the SoS that could affect the functionality or performance of an SoS capability. This includes changes to the technologies used to support the SoS, changes to the missions of the individual systems, and external changes that affect the SoS. To be successful, the SoS systems engineer requires a solid understanding of the SoS baseline and a broad awareness and understanding of trends in enabling technologies, technology insertion, and mission evolution. Further, the SoS engineering team needs to be aware of development and modernization activities and schedules both for the SoS and for its constituent systems. By understanding the impact of potential changes, the SoS engineering team can either intervene to prevent problems or develop strategies to mitigate the impact on the SoS.

An SoS is composed of multiple interdependent systems, which evolve independent of the SoS and of each other in ways that could affect the SoS. In turn, the systems could be affected by changes in the SoS. Unless the development activities of the systems are monitored and assessed, the performance of the SoS may decline due to the effect of a new system's configuration on SoS operations. For systems that are particularly critical to an SoS, the SoS engineering team may want to participate in the CCB of the system to register concerns regarding the impact of system changes on the SoS, and to encourage the system manager and SE team to accommodate SoS considerations in the system development.

It is critical that the SoS engineering team engage with the systems engineers of the constituent systems to understand the nature of their changes and to assess the potential impacts on the SoS. The SoS engineering team may identify alternatives for implementing the changes that would not affect the SoS and work to influence the systems to adopt alternatives.

The SoS engineering team must be particularly attuned to changes that have an impact on the cyber posture, such as new software versions in a constituent system that introduce vulnerabilities, or updated threat intelligence. Such changes should trigger a re-assessment of cyber risk, potentially leading to a new Mission Based Cyber Risk Assessment (MBCRA) or a targeted regression test event.

A major challenge is to sensitize the constituent systems' SE teams to the types of changes in their systems that are relevant to the SoS, and to create an environment of trust, where systems engineers are willing to share their plans early without fear that the SoS response may hamper their ability to support their own system user needs. To address this challenge, some SoS have established early configuration boards where systems engineers of constituent systems are asked to share all anticipated changes with the SoS engineering team early in the planning processes.

The process is tailored to make it easy to share plans early, and only when the plans affect the SoS are technical details needed. If issues are identified at the earliest stages, actions can be taken to minimize the disruption to SoS engineering plans. Another approach is to have members of the SoS engineering teams selectively participate in the configuration and technical reviews of key systems. In any case, the SoS engineering team needs to be mindful that the time of systems engineers for the constituent systems is already fully committed, even without the SoS.

As a result, in an SoS environment, the SoS engineering team needs to:

- Continually monitor proposed system changes and assess their impacts on the SoS.
- Identify opportunities for enhanced functionality and performance.
- Prevent or mitigate problems for the SoS and individual systems.
- Negotiate with systems engineers for constituent systems regarding how system changes are made, to prevent undesirable effects on the SoS and vice versa.
- Update the SoS product baseline as individual system updates or changes are deployed.

Figure 3-7 depicts the *Monitoring and Assessing Changes* core element input/output exchanges with the other core elements. The SoS engineering team identifies, tracks, and anticipates changes that will affect the SoS. They should engage constituent system engineers early and regularly to understand their combined planned changes and the likely impacts to both the SoS and the constituent systems. To help accomplish this, this core element receives inputs from:

- *Translating Capability Objectives* core element – expectations of the SoS and associated high-level requirements.
- *Understanding Systems and Relationships* core element – understanding the constituent systems, their relationships, and plans for known changes.
- *Assessing Performance Against Capability Objectives* core element – current performance versus desired performance.

External Influence inputs:

- Changes (in mission, technology, functionality, performance, threat, modernization efforts) to the individual systems, systems external to the SoS with which the SoS may interact, and associated schedules.
- Changes in demands on the SoS (new CONOPS, unplanned use of or demand for SoS capabilities).
- Changes in demands on the individual systems (new CONOPS, unplanned use of or demand for constituent system capabilities).

Output: The *Monitoring and Assessing Changes* core element provides an understanding of how changes affect the SoS and impact requirements and solution options to the *Addressing Requirements and Solution Options* core element. This core element also provides individual system changes to support SoS objectives, which impact systems and their relationships to the *Understanding Systems and Relationships* core element. By understanding the impact of proposed or potential changes, the SoS systems engineer can either intervene to preclude problems or develop strategies to mitigate the impact on the SoS.

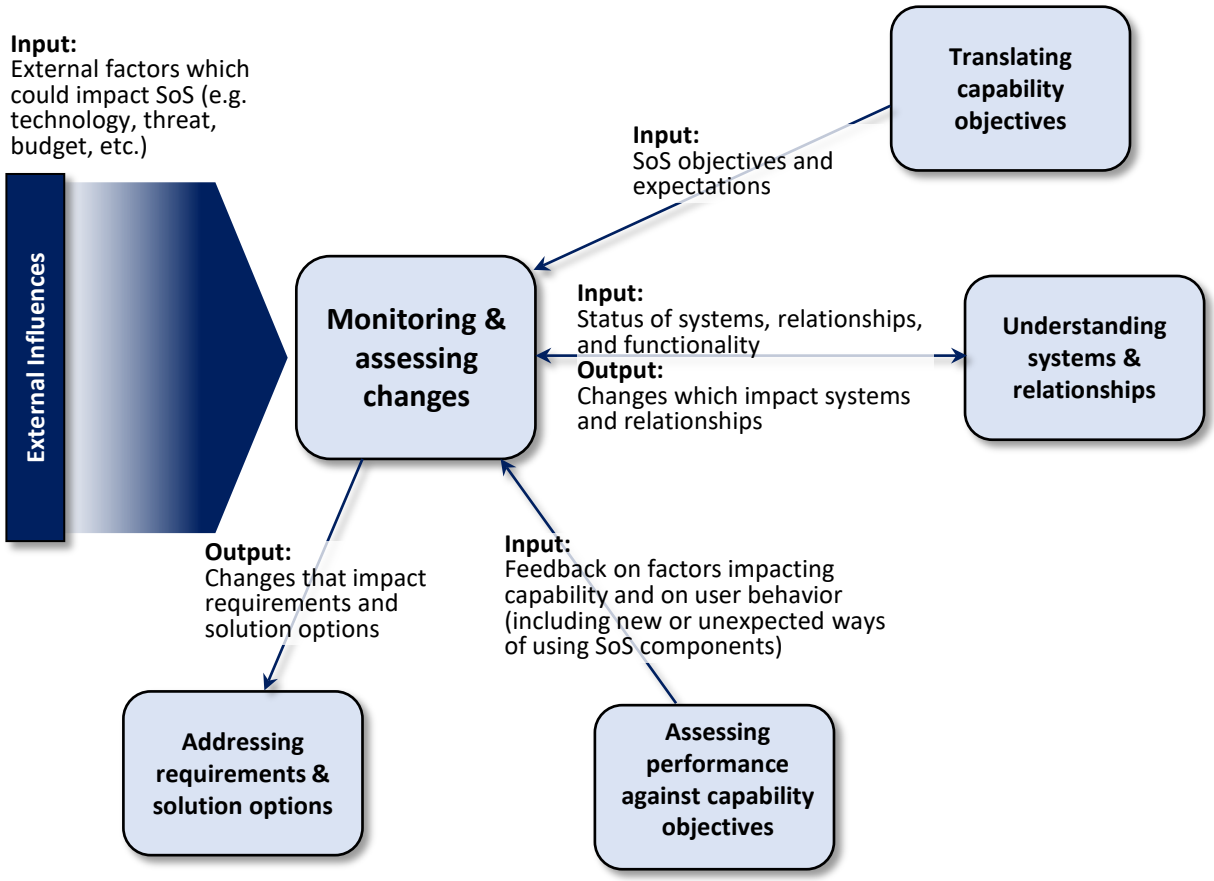


Figure 3-7. Relationship of Monitoring and Assessing Changes to Other SoS Engineering Core Elements

SE Processes in Monitoring and Assessing Changes

The following are recommended SoS engineering team activities for each of the SE processes as they pertain to the *Monitoring and Assessing Changes* core element. These are not all inclusive, and depending on the specific SoS and constituent systems, the SoS team should tailor their activities to their SoS needs.

For **Stakeholder Requirements Definition**, the SoS engineering team:

- Regularly collaborates with stakeholders to capture and characterize any updates in the capability objectives and SoS requirements.

For **Requirements Analysis**, the SoS engineering team:

- Identifies the impact of changes at the SoS level (e.g., capability objectives, threats, etc.) on the requirements of constituent systems and their relationships.
- Identifies the impact of changes at the constituent system level (e.g., technology, functionality, etc.) that may lead to conflicting requirements and cause SoS conflicts.

For **Architecture Design**, the SoS engineering team:

- Assesses the potential impact to SoS architecture and update design, if necessary, based on changes to constituent system components, interfaces, or SoS objectives.

For **Validation**, the SoS engineering team:

- Assesses if changes to SoS or constituent systems' requirements impact the validation of the SoS capability objectives.

For **Decision Analysis**, the SoS engineering team:

- Focuses the decision analysis on identifying and evaluating the impact of changes that might affect the SoS. This includes changes in enabling technologies, technology insertion, and mission evolution made by the constituent systems. It also includes consideration of potential changes in demands on the SoS (e.g., new CONOPS, unplanned use of or demand for SoS capabilities).
- Conducts sensitivity analysis and uses decision analysis tools to evaluate the impact of changes that might affect the SoS architecture, generate alternatives, assess alternatives, synthesize results, identify uncertainties, assess the impact of uncertainties, communicate trade-offs, present recommendations to decision makers, and implement the action plan.

For **Technical Assessment**, the SoS engineering team:

- Updates and employs user-oriented metrics to assess progress toward meeting SoS capability objectives and collect data to assess this progress.
- Assesses the technical progress in defining, planning, implementing, integrating, and testing the changes/upgrades in SoS as well as individual constituent systems.

For **Requirements Management**, the SoS engineering team:

- Documents newly added requirements due to imposed changes (either in SoS or constituent systems) and manages their traceability to the configuration of the deployed SoS.
- Reviews new and existing requirements to ensure they remain aligned with the SoS capability objectives for the purpose of validation traceability.

For **Risk Management**, the SoS engineering team:

- Identifies risks introduced by changes such as:
    - Technology maturity, especially version stability (this is a critical factor in SoS success).

  - Constituent system development and evolution.
  - Inclusion of legacy systems – while this may appear to lessen SoS risk, it may in fact complicate the SoS with several unknowns and hence increase risk (e.g., diminishing manufacturing or parts).
  - Evolving threats.
- Ensures risks introduced by changes are analyzed, mitigated, monitored, and documented to ensure no impedance to achieving SoS performance and capability objectives. The results are reviewed and agreed upon by SoS managers stakeholders.

For **Configuration Management**, the SoS engineering team:

- Maintains the "as is" configuration of the SoS as the constituent systems implement and deploy their own upgrades. These upgrades typically contain new functions needed to support SoS capabilities as well as new functions needed by the constituent system users and stakeholders.
- Establishes a CCB consisting of SoS stakeholders to review and assess how planned changes to constituent systems may affect the SoS.
- Performs configuration change management, which includes verification and audits, of each constituent system in an SoS by maintaining the baseline of each system's configuration. This helps monitor proper operation of each constituent system in its operational environment and in the context of the SoS mission objectives.
- Monitors the configuration management version control processes to ensure that changes to one system's configuration do not disrupt the overall SoS. This provides traceability of changes and supports rollback or recovery when needed.

For **Data Management**, the SoS engineering team:

- Focuses on data associated with changes that have been identified and evaluated, the results of the evaluation, and any action taken to mitigate the adverse effects of problematic changes.
  - The SoS engineering team documents this history to help address similar issues in the future.
  - The SoS engineering team ensures the history is communicated with the constituent systems and documentation is accessible.
- Implements a data management process to monitor and assess constituent system changes to support SoS mission objectives including those related to mission, technology, functionality, performance, and modernization efforts.

- Ensures data integrity and interoperability between systems by establishing common data standards and exchange protocols while monitoring and assessing constituent system changes that affect the SoS.

For **Interface Management**, the SoS engineering team:

- Uses the interface management process to track individual system interface changes and monitor progress of integrating the individual systems into the desired SoS architecture.
- Collaborates with constituent system PMs, SE teams, other stakeholders, and SMEs to establish an SoS-level Interface Working Group that serves as the communication link between those responsible for interfacing systems, end products, enabling products, and subsystems within the SoS architecture.
- Facilitates cooperation with and obtains agreement from all stakeholders on system interface changes to integrate into the SoS architecture.
- Updates the interface specifications to capture SoS internal and external interface changes and dependencies and continuously monitor new changes to ensure SoS mission objectives are achieved.

3.2.6 Addressing Requirements and Solution Options

An SoS has requirements both at the SoS level formed by the interoperating constituent systems and at the level of the individual constituent systems themselves. Depending on the circumstances, the SoS engineering team may have a role at one or both levels. At the SoS level, as with systems, a process is needed to collect, assess, and prioritize user needs, and then to evaluate solution options for addressing these needs.

Solution options for addressing capabilities/requirements may include:

- Adding new systems
- Adding existing (but new to SoS) systems
- Updating or extending functionality of existing systems
- Convincing owners of constituent systems to defer their changes in support of the SoS

These actions are outside the control of the SoS manager and systems engineers and will need to be negotiated with the constituent systems' owners. The SoS engineering team needs to understand the technical and organizational contexts and constraints of the individual systems when identifying options to address SoS needs, and to consider the impact of those options at the systems level.

The SoS engineering team should work with requirements managers for the individual systems to identify the specific requirements to be addressed by each of the systems (that is to collaboratively derive, decompose, and allocate requirements to specific systems). This activity is compounded at an SoS level due to the multiple acquisition stakeholders engaged in an SoS. The objective is to identify options that balance needs of the systems and the SoS, since in many cases there may be no clear decision authority across the SoS.

The SoS engineering team works with the SoS manager and stakeholders to review, prioritize, and recommend which SoS requirements to implement in each iteration. The selected requirements for each iteration are moved into the SoS functional baseline and become the new functional baseline for the iteration. This analysis includes controlling top-level SoS requirements changes to maintain stability and coherence. The SoS engineering team, again working with systems, is then responsible for leading the development and evaluation of technical approaches to address requirements as well as the selection of approaches to meet the requirements. The product of these activities is a technical plan for evolving the SoS, typically through incremental changes on the part of the systems and sometimes with added components specifically for the SoS.

Designs for implementing changes to the systems are done by the systems engineers of the constituent systems. The architecture of an SoS, if done well, will provide the persistent framework for identifying and assessing design alternatives and will provide stability as different requirements emerge. A well-engineered modular and open architecture will also withstand the impact of changes in one area on other parts of the SoS. High-level requirements are developed and updated and are then ready for further analysis and assignment to a functional baseline.

As the SoS engineering team addresses this core element, along with the last element, *Orchestrating Upgrades to SoS*, they are essentially implementing a two-level version of the process used with the SE of an individual system. At the top level, the SoS engineering team recommends requirements to be addressed and works with the constituent systems to identify and assess alternative options. In parallel, the SE teams of the constituent systems are working at the system level, assessing options for changes in their systems to address the needs. The result is selection of an option and a plan for implementing that option.

SoS systems engineers need to adopt a holistic approach when addressing requirements. In a single system development, in the best case the systems engineer has a set of prioritized requirements documented as a formal user capability need and validated. In an SoS, on the other hand, requirements evolution is often driven by a variety of sources:

- SoS environment changes
- Emergent behaviors

- Constituent system changes
- SoS upgrade problems
- User insights and needs
- Technology opportunities

This means that the SoS systems engineer needs to look more broadly at the set of longer-term needs and opportunistically address requirements in ways that practically leverage ongoing system activities and remain flexible to adapt to changes in user needs and priorities.

These analyses for the SoS are based on trades considering cost and risk, much as when other development decisions are made for the systems. While the changes are being made to support SoS needs, after implementation they become part of the constituent system product baselines that the system owners are responsible for maintaining over their life cycle. Consequently, life cycle costs of changes are a decision factor. Across the SoS, different sets of solution options are considered. It may be necessary to identify a wider range of options in an SoS because of the larger numbers of constraints presented by the SoS environment.

Decisions about where in the SoS to address SoS requirements (allocated baseline) are based on analysis done by both the SoS engineering team and the SE teams of the constituent systems. Together they identify and assess approaches for updating systems to meet SoS needs much as they do when system changes are examined to meet user needs. Systems engineers for the SoS and constituent systems assess the full range of issues, including life cycle cost, technical, and integration risk. Upgrades to the systems consider science and technology opportunities, lessons learned from technology explorations, commercial-off-the-shelf (COTS) solutions, and opportunities to leverage earlier solutions. Approaches to SoS data requirements can be found in the Data Services Environment with a registered DoW service. These are factored into decisions about which system changes should be made in an increment of SoS development. The SoS engineering team should understand the trade space and performance budgets allocated to individual systems to allow ongoing measurement and evaluation of an individual system's performance to meet SoS objectives.

In making recommendations about changes, the SoS engineering team needs to balance needs between the SoS and the constituent systems, leveraging the capabilities and plans of the systems that benefit the SoS. In the worst case, where the needs of the systems user's conflict with the objectives of the SoS, the SoS engineering team needs to identify these conflicts and assess ways to mitigate the risks inherent in them.

The development plans of the systems are also an important input to the SoS technical planning process because, in most cases, the SoS will add changes to the system development plans. The result is likely to be an asynchronous development and delivery of parts of SoS iterations, and in a large SoS, multiple iterations may be under way concurrently. This means the SoS engineering team should reflect the technical plans in the IMS and identify critical review events, risk assessment plans, and synchronization points. For a large SoS, this is not trivial. This highlights the need for continuous coordination in SoS fielding, which requires frequent planning, integration, and capturing of capabilities and limitations as the SoS evolves.

At the SoS level, typically only the SoS requirements are managed and considered by the SoS engineering team. Constituent system requirements are typically the responsibility of the systems. In most cases, the upgrades planned for the individual system will not address the needs of the SoS. The SoS engineering team needs to be aware of the requirements of the systems as well as plans for funding and scheduling changes so they may anticipate impacts of system changes on the SoS. In addition, knowledge about system requirements and technical plans is critical for the SoS engineering team because this knowledge helps the systems engineer identify options for addressing SoS requirements that may include leveraging efforts of the individual systems. The experience of SoS shows that the needs of the SoS can differ considerably from the aggregate needs of the systems.

Consequently, the SoS engineering team needs to identify situations where meeting needs at the SoS level may lead to potential sub-optimization of individual systems. The subsequent resolution is often negotiated between the SoS and system managers with support of the systems engineers. The trade-offs can be addressed through a value-driven design process to weigh the alternatives in terms of their comparative values to various users. Systems that are disadvantaged by these decisions may be resistant to support future SoS development, so it is important for the SoS manager and systems engineers to ensure systems understand the drivers and rationale for SoS decisions.

The SoS engineering team sometimes needs to consider non-optimal requirements allocation options to meet cost and schedule targets. For example, an optimal individual system may not be able to incorporate needed functions in the current increment, but other non-optimal systems might be able to achieve this goal. In an SoS it may be difficult to manage redundant capabilities in individual systems if the systems need the redundant capability to meet their own needs when operating separately or the needs of other SoS in which they participate.

Depending on the specific SoS, an SoS organization typically does not have the authority to execute these activities, so arranging to implement solutions is based on collaboration between managers of the SoS organization and the systems. New systems/components may be developed by one of the owners of the existing systems or by the SoS office itself. The SoS office developing a component of the SoS should be viewed as a dual hat or additional role separate from the role of the SoS engineering team.

Finally, this core element perhaps more than others, may involve a great deal of negotiation on the part of the SoS engineering team. Even when there is analysis that a change to one system will enable the SoS to meet a requirement and funding is available to implement it, this does not guarantee the system's PM, sponsor, and systems engineer will be willing to implement the added functionality. There may be issues when a system is part of multiple SoS especially if they have competing demands for system support. Changes that satisfy the one SoS may not be acceptable to the second SoS.

Often, the SoS engineering team and manager must convince the systems engineer and PM for a constituent system that it is in the constituent system's interest to change its implementation to meet the SoS needs. To the degree that the SoS engineering can identify ways that the changes needed for the SoS can be done without affecting a constituent system's development schedule or that the changes support a system's longer-term development objectives, a constituent system will be more receptive to taking on added functionality to support the SoS objectives.

As noted above, the product of this element is a technical plan for implementing the changes for the next iteration of the SoS. In developing this plan, the SoS engineering team follows the principles for technical planning for systems, paying attention both to defining critical event-driven reviews and to risks throughout the process. Particular attention is paid to area of importance in an SoS including the development of an IMS, which focuses on key integration points and links to the detailed development schedules maintained by the systems. The plan also addresses roles and responsibilities supported by MOAs, which detail specific commitments of the participant in the increment development.

Figure 3-8, below, depicts the *Addressing Requirements and Solution Options* core element input/output exchanges with other core elements. This core element receives inputs from:

- *Monitoring and Assessing Changes* core element – expected impacts of changes on SoS, including planned individual system changes, SoS objectives, mission engineering, and organizational changes.
- *Understanding Systems and Relationships* core element – windows of opportunity for changes and associated options.

- *Developing and Evolving SoS Architecture* core element – current SoS architecture and associated constraints.
- *Orchestrating Upgrades to SoS* core element – problems/issues associated with implementation of previous planned SoS updates.

Output: The *Addressing Requirements and Solution Options* core element provides technical plans for SoS upgrades to the *Orchestrating Upgrades to SoS* core element, and status of systems, relationships, and functionality to the *Developing and Evolving SoS Architecture* core element. The SoS engineering team can recommend requirements and solution options, and work with constituent systems to identify and assess alternatives. Constituent system teams assess options for their own systems in parallel, and through collaboration with the SoS engineering team, evaluate impacts on the SoS. Through this collaboration, conflicts between SoS and constituent systems can be identified, risk mitigation plans developed, and alternatives evaluated.

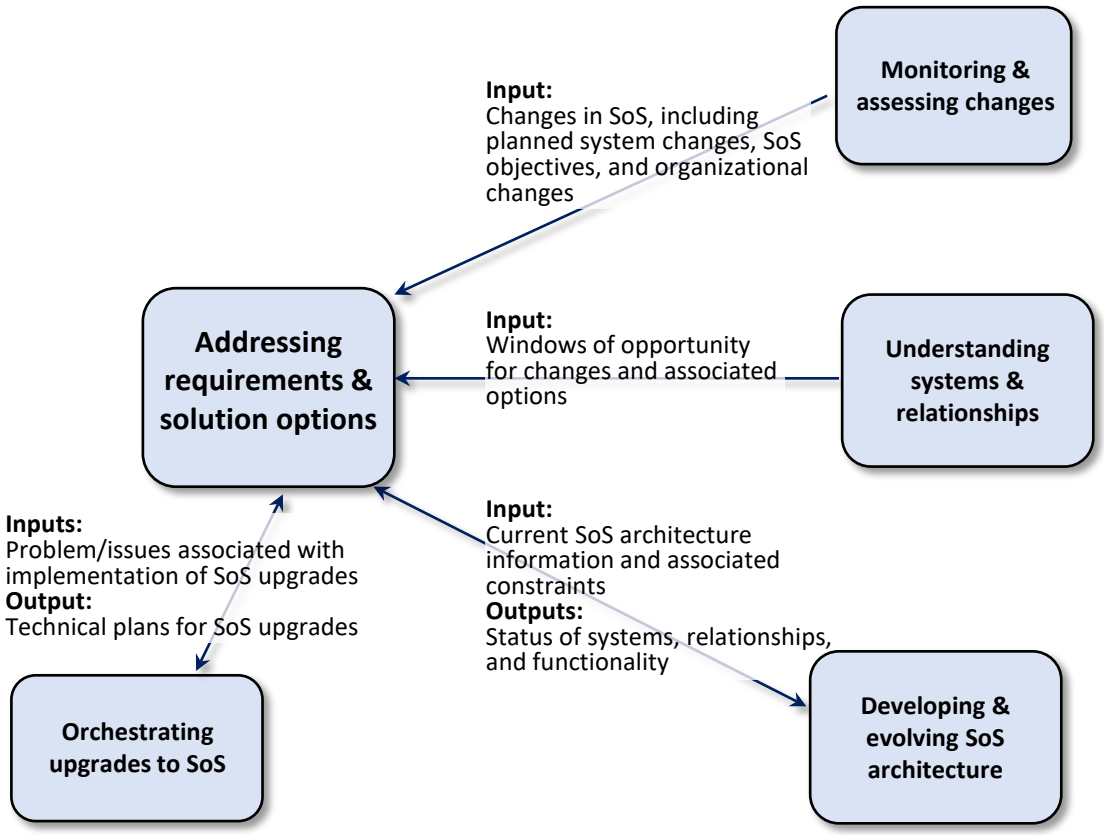


Figure 3-8. Relationship between Addressing Requirements and Solution Options and Other SoS Engineering Core Elements

SE Processes in Addressing SoS Requirements and Solution Options

The following are recommended SoS engineering team activities for each of the SE processes as they pertain to the *Addressing SoS Requirements and Solution Options* core element. These are not all inclusive, and depending on the specific SoS and constituent systems, the SoS team should tailor their activities to their SoS needs.

For **Stakeholder Requirements Definition**, the SoS engineering team:

- Conducts trade-off analysis on requirements and solution options with SoS and constituent system stakeholders to ensure the SoS capability objectives will be met.
- Reviews SoS requirements with SoS and constituent system stakeholders and updates the solution options regularly to ensure they remain aligned with changes in constituent systems.
- Ensures high-level requirements developed and updated are ready for further analysis and assignment to a functional baseline.
- Establishes the allocated baseline as solutions are developed and maps requirements to individual systems for implementation.

For **Architecture Design**, the SoS engineering team:

- Identifies viable solution options to satisfy SoS requirements and defines an approach to implement the selected option(s) in the SoS architecture design framework.
- Identifies internal and external requirements and evaluates the solution option (s) impact on the SoS architecture design framework.
- Manages the SoS architecture design framework updates due to changes in both requirements and solution option(s).
- Reviews the results and negotiates agreements at a set frequency with SoS stakeholders or established board to ensure SoS mission objectives are achieved.

For **Technical Planning**, the SoS engineering team:

- Considers solution options for meeting SoS requirements with respect to constituent systems' available resources, schedule, life cycle milestones, and cost and develops a technical plan to implement the preferred option.
    - In an SoS, this technical plan reflects negotiations with the systems engineers for constituent systems, since in most cases the SoS systems engineer has no control over the constituent systems.

    - The final product is a technical plan for the iteration of SoS evolution.
- Conducts technical reviews and audits on a set frequency with SoS stakeholders and decision makers to negotiate and implement the preferred solution to requirements with respect to available resources, schedule, life cycle milestones, and cost to support the SoS mission critical objectives.

For **Decision Analysis**, the SoS engineering team:

- Focuses on addressing two questions:
    - Which of the requirements solutions can be reasonably implemented in the next iteration?
    - What are the options for implementing them?
- Performs decision analysis using a much broader trade space to accommodate a more uncertain and dynamic environment, to determine which solutions to which requirements can be implemented at a given iteration.
- Uses decision analysis to identify windows of opportunity for implementing solutions for SoS requirements in individual systems within their systems constraints.
- Uses decision analysis tools to continuously monitor and evaluate the impact of changes in requirements and their solution options that can affect the SoS architecture.
- Prepares a decision briefing with action/implementation plan(s) for different requirements solution options and provides appropriate recommendations to decision makers regarding the resulting impacts on the SoS architecture.

For **Requirements Management**, the SoS engineering team:

- Works with the SE teams from the constituent systems to identify requirements and solution options that support the SoS mission critical objectives and align with the constituent systems requirements flow down.
- Conducts reviews on a set frequency with an established CTC that consists of SoS stakeholders to implement requirements changes and solution options including their traceability to constituent systems to achieve desired SoS mission critical objectives.

For **Risk Management**, the SoS engineering team:

- Develops the risk management process to address the following questions:
    - What are the risks associated with implementing each solution option?

  - What are the risks of not addressing potential impacts of changing individual systems?
  - What are the resources necessary to mitigate the identified risks for each solution option?
- Develops risk mitigation plan to include risk acceptance, risk avoidance, risk transfer, risk control, or risk burn-down to reduce the likelihood or impact of the risks on the SoS and constituent systems.
- Reviews and documents progress of the risk burn-down history with an established risk management board that consists of SoS stakeholders.
- Continuously monitors and assesses risks introduced by requirements changes and their solution option(s) in the SoS and constituent systems to ensure no negative impact to achieving desired SoS critical mission objectives.

For **Configuration Management**, the SoS engineering team:

- Manages requirements changes and the implementation of solution option(s) on the functional and allocated baselines of the next SoS iteration and on the affected constituent systems baselines.
- Uses the configuration management version control processes to ensure that requirements changes and implemented solution option(s) on individual system's reference architecture baseline configuration is captured in the SoS reference architecture baseline configuration as necessary.
- Reviews, with the CCB, the effectiveness of configuration controls as solution options are implemented.

For **Data Management**, the SoS engineering team:

- Identifies and documents data requirements that address SoS requirements changes and their solution option(s) and use it to explain to SoS stakeholders what was considered, what was decided, and why.
  - The record can serve as a starting point for assessing additional requirements changes and solution options over time.
- Ensures that the data management process enables the capture of comprehensive data (both technical and operational) from all constituent systems that addresses requirements changes and solution option(s) and ensures data accuracy, integrity, integration, and availability across the SoS to achieve mission-critical objectives.

For **Interface Management**, the SoS engineering team:

- Develops a strategy to employ new standard/common interfaces (operational data and semantic data), including data communications, by replacing the legacy interfaces in constituent systems in the SoS to meet the current needs and to address requirements changes and their solution options.
- Facilitates cooperation and agreements through Interface Working Group with all stakeholders on system interface changes and integration in constituent systems and the SoS.
- Updates documentation to capture SoS internal and external interface changes and continuously monitors for new changes to achieve the desired SoS mission-critical objectives.

3.2.7 Orchestrating Upgrades to SoS

Once a solution option for addressing a requirement has been selected, it is the SoS engineering team's role to work with the SoS sponsor, the SoS manager, the constituent systems' sponsors, managers, systems engineers, and contractors to fund, plan, contractually enable, facilitate, integrate, and test upgrades to the SoS. *Orchestrating Upgrades to SoS* is triggered when the various organizations involved in a solution implementation accept a technical plan for addressing SoS requirements. Most important, the plan includes the managers and systems engineers of the constituent systems, who will be implementing the changes that they agreed to during the plan development. This plan is then executed under this element.

In *Orchestrating Upgrades to SoS*, the SoS engineering team oversees the implementation of the technical plan associated with an SoS upgrade, then integrating, testing, and evaluating the constituent systems in the SoS environment. The team is not, however, controlling the individual baselines of the constituent systems. The actual changes are made by the constituent systems' owners, but the SoS engineering team orchestrates the process, taking a lead role in the coordination, integration, and test across the SoS and provides oversight to ensure that the changes agreed to by the systems are implemented in a way that improves SoS performance.

When executing the SoS plans, the SoS engineering team applies SE processes, but at a higher level, to coordinate actions of organizations that may be quite independent. At both the SoS and constituent system levels, the systems engineers will need to work with the PMs to determine the best coordination of some of the iterations to meet scheduled upgrade rhythms. Tools like critical path and critical chain (resource alignment to critical path) analyses can support SoS implementation planning. In a large SoS, multiple iterations may be under way concurrently.

External factors may affect the execution of this technical plan and may interrupt the ability to implement the changes. External factors may be technical issues—such as characteristics of the host system—that systems engineers might not have fully understood during the planning process. These technical issues might drive up the cost of the SoS solution, take more time to implement the solution, or even make the solution technically infeasible. There might also be programmatic issues, budget cuts, or new higher-priority development needs directed by the user of the system. In any case, these external factors may require the SoS engineering team to revisit the technical plans or adjust expectations.

As they execute the plan and complete the SoS upgrades, the SoS engineering team assesses the performance of the modified SoS. Changes made in constituent systems to support the SoS are tested and evaluated, as are the sets of systems with changes that contribute to SoS performance improvements. The T&E results provide feedback to the developers during implementation and later inform the users and sponsors about deployment decisions. As a result, the SoS engineering team receives feedback on problems/issues with new SoS solutions and on changes to the constituent systems and their functional relationships resulting from the SoS upgrade.

In *Orchestrating Upgrades to SoS*, negotiating and pacing the upgrades are key aspects of the systems engineer's role. SoS orchestration can include both deliberate, plan-based increments and capability-driven builds. In either case, the evolving SoS needs to accommodate the asynchronous system development processes for multiple constituent systems. In most cases, it is nearly impossible to align the development cycles of all constituent systems. Consequently:

- Who does what and when will be driven by practicalities as much as technical considerations.
- Systems engineers need to develop an incremental approach that leverages the activities already planned for in the constituent systems.
- Designs must be flexible with respect to building and fielding parts of a solution, since SoS design releases are often driven by system schedules.
- Systems engineers and the T&E community need to be creative about T&E, leveraging a variety of data, M&S, test results, and venues.

Effective SoS engineering assumes that the systems engineers for the constituent systems are implementing SE for the individual systems and that the SoS engineering team can therefore focus on the areas that are critical across the SoS. Needed changes are implemented in the constituent systems under their own SE process. The SoS engineering team coordinates across these processes, which may or may not be compatible. Coordinating across these processes involves substantial negotiation and may result in a reassessment of requirement solution options and approaches if the logistics or technical feasibility breaks down.

Throughout orchestration, the systems are implementing changes according to the negotiated SoS technical plans, and they are following their own SE and T&E processes. The SoS engineering team works with the SE teams for the constituent systems to enable SoS insight into progress of these system developments. The SoS engineering team members are responsible for integration, verification, and validation of the changes across the suite of system updates under an SoS increment, including T&E tailored to the specific needs of the increments. Their efforts may result in both performance assessments and a statement of capabilities and limitations of the SoS increment from the perspectives of SoS users and users of the individual systems.

These assessments may be conducted in a variety of venues, including distributed simulation environments, system integration laboratories, and field environments. The assessments can take a variety of forms, including analysis, demonstration, and inspection. Often, the SoS engineering team leverages planned activities of the constituent systems to address SoS issues.

SoS engineering approaches based on multiple small increments offer a more effective way to structure SoS evolution. Large implementations typically will not work in this environment since it is not feasible with multiple asynchronous independent programs. Specifically, several SoS initiatives have adopted what could be termed a “bus stop,” spin, or block-with-wave type of development approach. In this type of approach, there are regular time-based SoS “delivery” points, and systems target their changes for these points.

Programs conduct integration, test, and evaluation for each delivery point. If systems miss a delivery point because of technical or programmatic issues, they know they have another opportunity at the next delivery point (there will be another bus coming to pick up passengers in 3 months, for instance). The impact on the SoS of the constituent systems missing the scheduled bus should be evaluated and addressed. By providing this type of SoS battle rhythm, discipline can be inserted into the inherently asynchronous SoS environment. In a complex SoS environment, multiple iterations of incremental development may be under way concurrently.

Approaches such as this may have a negative impact on certification testing, especially if the item is software, related to interoperability, and/or safety issues (such as Airworthiness Release). When synchronization is critical, considerations such as this may require large sections of the SoS, or the entire SoS, to be tested together before any of the pieces are fielded.

Figure 3-9 depicts the *Orchestrating Upgrades to SoS* core element input/output exchanges with the other core elements. This core element receives inputs from:

- *Addressing Requirements and Solution Options* core element – technical plan to facilitate, monitor, and coordinate the changes being implemented in the constituent systems to achieve SoS performance improvements and added capability.

- *External Influences* – external factors that impact the ability to execute and implement the technical plan.

Output: The *Orchestrating Upgrades to SoS* core element provides information about problems or issues encountered with implementing upgrades to the *Addressing Requirements and Solution Options* core element, updates to systems and relationships to the *Understanding Systems and Relationships* core element, and a technical plan to initiate a performance assessment of the modified SoS to the *Assessing Performance against Capability Objectives* core element.

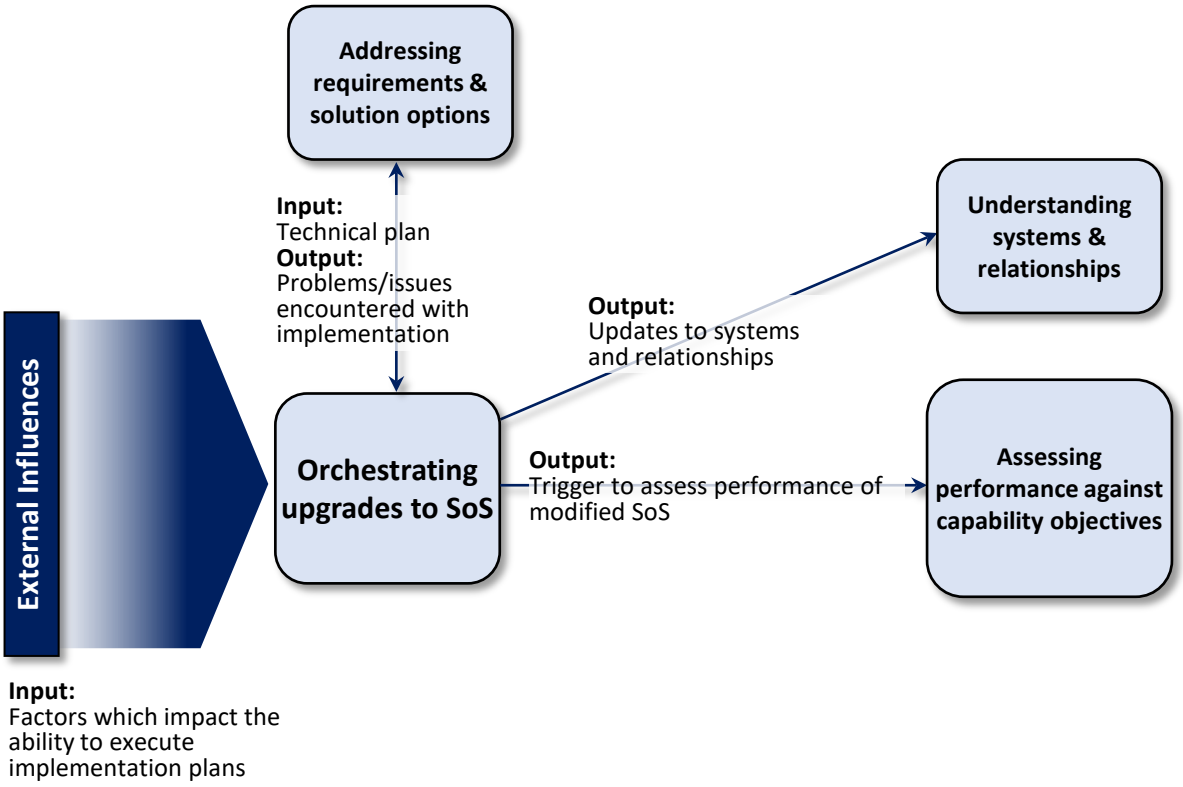


Figure 3-9. Relationship between Orchestrating Upgrades to SoS and Other SoS Engineering Core Elements

3.2.8 SE Processes in Orchestrating Upgrades to SoS

The following are recommended SoS engineering team activities for each of the SE processes as they pertain to the *Orchestrating Upgrades to SoS* core element. These are not all inclusive, and depending on the specific SoS and constituent systems, the SoS team should tailor their activities to their SoS needs.

For **Implementation**, the SoS engineering team:

- Provides guidance on upgrades made for the SoS and negotiates with SE teams for constituent systems to accommodate asynchronous implementation of changes.
- Oversees and coordinates with SE teams for constituent systems as they conduct decision analysis and trade studies to determine the best way to implement the SoS requirement solution within their system.
- Identifies and analyzes the constraints that new technology insertion or changes to constituent systems impose on the SoS, while maintaining normal operations to ensure that SoS mission-critical objectives are achievable.
- Monitors constituent system implementation of upgrades.

For **Integration**, the SoS engineering team:

- Integrates the constituent system changes, focused on end-to-end functionality and performance across the SoS, while the SE teams of the constituent systems are responsible for implementation and integration of changes within their systems. In an SoS, asynchronous system development may necessitate asynchronous upgrades integration.
- Conducts simulations and integration testing to assess how well systems perform in an operationally realistic environment before deployment of SoS upgrades.
- Performs regression testing to ensure that constituent systems are not adversely affected by SoS upgrades or the SoS is not adversely affected by individual system changes.
    - For integration and testing of SoS upgrades, the constituent systems are often treated as a "black box" (only in terms of their inputs and outputs), unless the SoS behavior is particularly sensitive to the behavior of a specific system.

For **Verification**, the SoS engineering team:

- Performs end-to-end verification testing in a realistic environment on SoS upgrades while the SE teams of the constituent systems are responsible for performing verification testing to ensure that the systems have implemented upgrades according to plans.
- Documents the verification results of the SoS upgrades and analyzes them against the verification success criteria to ensure the SoS and constituent systems have been upgraded according to plan.
- Reviews the verification results of the SoS upgrades with SoS stakeholders.

For **Validation**, the SoS engineering team:

- Uses M&S techniques to conduct end-to-end validation testing on SoS upgrades to ensure that SoS mission objectives are achieved, and ensures the simulations support physical testing results.
- Performs limited end-to-end validation testing on SoS upgrades *in an operational environment* for mission-critical functions, while the SE teams for the constituent systems are responsible performing validation testing to ensure that the changes systems have implemented have the desired effects.
- Documents the validation results of the SoS upgrades and analyzes them against the validation success criteria to ensure the SoS and constituent systems are performing to expectations.
- Reviews the validation results of the SoS upgrades with SoS stakeholders.

For **Transition**, the SoS engineering team:

- Focuses on supportability and sustainment for the SoS that relies upon transition activities that are conducted and managed at the constituent system level.
- Verifies that the upgrades transitioned by the constituent systems maintain bi-directional traceability to support and sustain SoS functionalities in operational environment.
- Documents the upgrade transition results and reviews with SoS stakeholders.

For **Decision Analysis**, the SoS engineering team:

- Focuses on the most critical decisions that relate to:
  - What can be done when upgrades do not go as planned?
  - When a system cannot implement changes as planned, what should be done to ensure that the SoS benefits from the other changes?
  - What adjustments can be made to compensate for any impacts on the SoS?
- Prioritizes an SoS architecture that minimizes interdependencies so it can buffer the SoS and constituent systems from the impacts of problems encountered in upgrade implementation.
- Uses decision analysis tools to evaluate the impact of upgrades on the SoS cost, schedule, and performance to achieve the desired SoS mission critical objectives and prepares a decision briefing with action/implementation plan(s) and appropriate recommendations for decision makers.

For **Technical Assessment**, the SoS engineering team:

- Is responsible for monitoring progress of the constituent systems as they implement changes and assessing technical risks as part of the SoS technical plan.
- Collaborates with SE teams for the constituent systems to perform technical assessments on critical upgrades to the SoS or constituent systems and to address entry/exit criteria as they apply to the technical plan of each.
- Conducts technical reviews and audits with SoS stakeholders as upgrades are implemented in the SoS.

For **Requirements Management**, the SoS engineering team:

- Updates the SoS RTM to ensure traceability between the constituent systems and the SoS to ensure no critical stakeholder's needs are missed and the desired SoS mission objectives are achieved.
- Reviews requirements, with CTC, based on upgrades in the SoS and continent systems and documents the decisions made.

For **Configuration Management**, the SoS engineering team:

- Updates the SoS technical baseline and documents all changes made to the SoS configuration until its transition through the version control process.

For **Risk Management**, the SoS engineering team:

- Collaborates with SE teams for the constituent systems to identify and manage potential risks that relate to the SoS upgrades as individual systems implement changes.
- Coordinates, through an established RMB, risk assessment results and mitigation plan progress for risks associated with SoS upgrades and documents the agreement on the path forward.
- Continuously monitors the implementation of mitigation plans for risks associated with upgrades to reduce the probability of their occurrence or minimize their impact on the performance of the SoS and individual systems that support achievement of SoS mission objectives.

For **Data Management**, the SoS engineering team:

- Focuses on capturing data about the changes to constituent systems implemented as part of the upgrade to the SoS and ensures the data compatibility across the SoS and constituent systems through the data management processes.

- Ensures that the data management process enables the capture of comprehensive data (both technical and operational data) from all constituent systems that addresses system changes for SoS upgrades, and ensures its accuracy, integrity, integration, and availability across the SoS to achieve mission objectives.

For **Interface management**, the SoS engineering team:

- Facilitates cooperation and agreement on constituent system interfaces due to upgrades to the SoS through an established Interface Working Group with all stakeholders.
- Tracks the implementation of the agreed-upon interface and constituent system changes to support SoS upgrades including conflicts/problems and resolutions identified during implementation through the interface management processes.
- Updates documentation to capture SoS internal and external interface changes and continuously monitors new changes due to upgrades to support interoperability between the SoS and constituent systems.

Appendix A: Additional Engineering Considerations for SoS

Innovation and continuous improvement are fundamental to sustaining excellence in military SE. This section explores strategies for integrating cutting-edge technologies, adapting to evolving threats, and fostering a culture of innovation within military SoS engineering practices. It emphasizes the need for ongoing education, collaboration across disciplines, and proactive leadership to navigate the complexities of modern warfighting environments.

Anticipating future trends and challenges is crucial for military systems engineering to keep pace with evolving technologies and global dynamics. DoW engineers must ensure that military systems are not only robust and flexible but also capable of adapting to emerging threats and new operational demands.

A.1 Digital Engineering in an SoS Environment

Digital engineering (DE), as codified by DoDI 5000.97, has emerged as a critical enabler in the design, development, and sustainment of complex systems, particularly within a SoS environment. The application of DE practices provides significant improvements in the integration, testing, and evolution of constituent systems by leveraging advanced modeling, simulation, and data management capabilities. This appendix explores how DE methodologies can be effectively applied within the SoS context to enhance efficiency, decision making, and life cycle management.

A.1.1. Key Components of Digital Engineering in SoS

DE in an SoS context incorporates several key practices and technologies to address the complexities of interconnected systems:

- Model-Based Systems Engineering (MBSE): MBSE includes the use of models as authoritative sources of truth, helping to integrate essential systems within an SoS. Using models facilitates a better understanding of relationships, dependencies, and interfaces across the systems, providing clarity in managing interactions within the SoS. MBSE tools are a major part of the current effort for digital engineering.
- Digital Twin: A digital twin is a virtual representation of a physical entity and includes detailed information about that entity. It involves a two-way data connection between the virtual and physical instances, allowing for real-time or near-real-time data synchronization throughout the entity's life cycle stages. This synchronization is achieved through data connections, which can be single-directional or bidirectional, ensuring credibility and appropriate synchronization rates. A digital twin system encompasses the digital twin and the interconnected entities, data connections, models, and interfaces.

- Digital Thread: The digital thread is a collection of linked authoritative digital information about a process, product, or system whose consistency is actively managed throughout the life cycle. This enables accessibility, traceability, currency, applicability, and credibility of information, thus facilitating the capture, communication, and use and reuse of knowledge to inform decisions that realize value efficiently. A digital thread provides an integrated view of data across the system life cycle, helping track decisions, changes, and outcomes in real time.
- Data-Centricity: DE facilitates a shift from traditional document-centric approaches to data-centric ones. In an SoS environment, information from disparate systems is accessible, reusable, and securely shared across all stakeholders. Data standards and metadata governance are crucial for integrating constituent systems and achieving interoperability.
- Federated and Integrated Digital Ecosystems: A federated and integrated digital ecosystem is a dynamic network of interconnected, yet independently managed digital technologies and platforms designed to optimize the DoW's engineering, acquisition, operations, and sustainment efforts. This approach enhances mission outcomes and efficiency by balancing local autonomy with centralized coordination. Federated ecosystems allow individual units or departments to maintain control over their specific technologies and data. At the same time, integration ensures that these disparate systems can seamlessly collaborate and share information across the broader DoW infrastructure.

A.1.2. Benefits of Digital Engineering in SoS

- Enhanced Decision Making: By integrating models, simulations, and real-time data from all constituent systems, DE enables faster and more accurate decision making. This is particularly important in mission-critical environments where timely responses are essential.
- Continuous Integration, Verification, and Validation: DE supports continuous integration, verification, and validation by allowing system engineers to test SoS configurations in virtual environments before implementation. This reduces the risk of unexpected behaviors and enhances overall system reliability. Continuous verification and validation helps to identify and reduce risk early by identifying defects causing unexpected behavior. This helps to ensure that the reliability of the system will meet the user's needs and requirements.
- Managing Complexity: Managing complexity is a key benefit of DE in an SoS environment. SoS typically consists of multiple independent systems each with its own life cycle, design, and operational requirements. The inherent complexity of coordinating these systems, ensuring interoperability, and addressing emergent behaviors can

overwhelm traditional engineering methods. DE mitigates these challenges by providing a model-based, data-centric approach that integrates diverse systems into a unified digital environment.

A.1.3 Challenges of Implementing Digital Engineering in SoS

- Interoperability: Given the independent nature of systems within an SoS, ensuring each system can communicate and exchange data effectively is crucial. Through standardized data formats and interfaces, DE practices can help mitigate system integration challenges.
- Life Cycle Synchronization: Systems in an SoS often follow different life cycles. DE can address this by offering real-time insights into individual systems' development, upgrade, or retirement, highlighting changes in individual systems that may affect the SoS capabilities.
- Data Management: Handling large volumes of data generated by interconnected systems is challenging. DE practices introduce automation in data management, which can improve the consistency and quality of the data used to evaluate and optimize system performance.

The future of DE in an SoS environment includes the integration of AI and ML to predict system behaviors and optimize SoS performance. As the DoW continues to shift toward a data-centric paradigm, DE will play a pivotal role in ensuring systems remain adaptable, resilient, and capable of meeting evolving operational needs. DE represents a transformative approach to managing the complexities of SoS by offering advanced tools for integration, decision making, and life cycle management. Adopting DE practices will be instrumental in driving the efficiency and effectiveness of future SoS operations.

A.2 Modeling and Simulation in an SoS Environment

M&S provides a technical toolset that is regularly used to support systems acquisition and engineering. Programs apply M&S throughout the system development life cycle supporting early concept analysis, through design, DT&E, integration, and OT&E.

Because of the characteristics of SoS, M&S can be a particularly valuable tool to the SoS engineering team and can support SoS engineering in several areas. Models, when implemented in an integrated analytical framework, can be an effective means of understanding the complex and emergent behavior of systems that interact with each other. They can provide an environment to help the SoS engineering team to create a new capability from existing systems and consider integration issues that can have a direct effect on the operational user.

M&S can support analysis of architecture approaches and alternatives and can support analysis of requirements and solution options. M&S also supports decision making in an SoS environment. To simulate system-level decision making in a SoS environment, the SoS engineering team should consider using agent-based modeling and discrete event simulation.

Because it can be difficult or infeasible to completely test and evaluate capabilities of the SoS, M&S can be effectively applied to support T&E at different stages throughout the SoS engineering process. The SoS engineering team should consider using M&S to understand the end-to-end performance of the overall SoS before implementation.

M&S in an SoS environment can be challenging because of the likely separately developed models with differing developmental environments, but if early models of the systems forming the SoS can be constructed and validated, potential problems may be identified and understood at early stages of the life cycle. Consequently, it is important to include planning for M&S early in the SE planning, including the resources needed to identify, develop, or validate M&S to support SE and T&E. Currently, with the DoW move to digital environments and data-centric systems development, it is possible to construct integrated SoS models that can interact and share data from the constituent systems more easily using application programming interfaces (APIs) (see A.5.1 and Appendix C). Additional information on M&S can be found at https://cto.mil/sea/dems. Refer to DoDI 5000.61 for policy on the verification and validation of models and simulations.

A.3 Modular Open Systems Approach in an SoS environment

MOSA implements an open architecture, government reference architecture, reference model, or system model with severable modules, enabling components to be openly competed among a diverse range of suppliers, which fosters innovation and drives down costs. This loose coupling of components uses open interfaces with open standards that results in a modular architecture. By carefully defining interfaces, organizations can create a modular architecture that is not only cohesive and functional but also scalable and adaptable to changing technological and operational needs.

SoS engineers should develop and share reference architectures that document SoS modular functionality, module boundaries, and module interfaces. This can improve communication and collaboration across the SoS, provide standardization and consistency, accelerate development and deployment, and facilitate interoperability and standards compliance.

Given the tension between the needs of systems themselves and the demands of the SoS, there is a real advantage to an SoS architecture based on a MOSA and loose coupling. MOSA provides systems maximum flexibility to address changing needs of original users and permits engineers

to apply technology best suited to those needs without an impact on the SoS. SoS architecture trades may place a greater emphasis on approaches which are extensible, flexible, and persistent over time and which allow the addition or deletion of systems and changes in systems without affecting other systems or the SoS as a whole.

While it is possible that the systems supporting an SoS objective will comply with such an architecture at the outset, the development of an open architecture and migration of systems over time is a typical and desirable approach for an SoS. Service-oriented architecture (SOA) is an example of this type of architecture approach.

For more information, see the SE&A MOSA web page (https://www.cto.mil/sea/mosa), current DoW policy (USW(A&S) Memorandum September 2026).

A.4 Mission Engineering in an SoS Environment

Mission engineering is an interdisciplinary process encompassing the entire technical effort to analyze, design, and integrate current and emerging operational needs and capabilities to achieve desired mission outcomes (Mission Engineering Guide 2023).

The mission engineering process decomposes missions into constituent parts to explore and assess relationships and impacts in executing the end-to-end mission. Mission engineering is used to identify and quantify gaps, issues, or opportunities across missions and seeks to address these by assessing the efficacy of potential capability solutions (materiel or non-materiel) that enhance mission outcomes.

From a practical implementation standpoint, the Mission Engineering process should be executed using formal, standards-based modeling languages to ensure rigor and interoperability. The “end-to-end mission approaches” are best modeled as SysML Activity Diagrams, where the sequence of mission tasks is clearly captured. To align with architectural standards, these activities can be organized within the UAF Operational Viewpoints, specifically providing the detail needed in behavioral views like “Operational Processes (Op-Pr)”, which are subsequently mapped to high-level capabilities using the “Operational Traceability (Op-Tr)” view. This approach creates a direct, analyzable trace from high-level mission objectives down to the specific functions that constituent systems must perform, providing the quantitative, mission-based inputs needed for effective systems engineering.

Mission engineering results inform decisions on military requirements, acquisition, research, and development as well as enable an early shift from qualitative to quantitative analysis. The methodology evaluates end-to-end mission approaches that include measurable elements amid warfighter-defined, threat-informed operational contexts. Mission engineering can assess a range of potential solutions (materiel and non-materiel) within a mission context to inform systems or SoS design and integration considerations, operational concepts, and trade-offs in Doctrine, Organization, Training, Materiel, Leadership and Education, Personnel, Facilities, and Policy, based on impacts to the mission.

Mission engineering has direct application to systems engineering processes by providing a better understanding of characteristics, performance parameters, functions, and features of systems and SoS that have an impact on mission outcomes. 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 results of mission engineering are used for a variety of purposes. For instance, findings can inform technology investments, suggest alternative ways to use current systems, identify mission gaps and preferred approaches to addressing these gaps, and trigger the initiation of a new acquisition to meet capability gaps. Mission engineering results may satisfy the requirements for a Capabilities Based Analysis or provide the starting point for prototyping. For more in-depth information on mission engineering, see the OUSW(R&E) Mission Engineering web page (https://ac.cto.mil/mission-engineering/).

A.5 Software Engineering in an SoS Environment

Software plays a fundamental role for development of a system and for an SoS. The purpose of software engineering (SWE) is to provide highly disciplined and rigorous processes for software and system architecture, design, and development to ensure quality software for the warfighter. In the DoW, SWE practices within an SoS ensure the development of high-quality, reliable software components that seamlessly integrate with other systems. This allows the overall SoS to function effectively and meet mission requirements by mitigating risks associated with complex software interactions and dependencies across multiple systems, while prioritizing security and adaptability to evolving threats.

Properly used SWE processes can mitigate cost and schedule risks by allowing programs to identify and remove software-related technical debt early in development, while delivering warfighter-needed capabilities at the speed of need. Early removal and continuous evaluation of software-related technical debt can increase acquisition efficiency and lead to higher success rates during operational testing and the continuing development process. Modern SWE processes include incorporation of Agile methodologies and the DoW's Development, Security, and Operations (DevSecOps) framework to enable continuous integration and more rapid delivery of capabilities to the warfighter in an SoS.

While software is the glue that ties all the systems together to provide the warfighter a needed capability, there are key differences in software engineering practices for a single system as compared with an SoS. For example:

- Different acquisition pathways may be used by each independent system, if it is not a "directed" type of SoS.
- Each system within an SoS has different users, and they may cross multiple domains, e.g., air, ground, sea, and space, and may have differing needs or concerns. Gaining coherent user feedback will require close alignment with the end-to-end mission threads and needs to involve all user domains.
- Data in different systems may have used different ontologies, as well as differing semantics, e.g., billet versus job position versus assignment.
- Each system within the SoS may have different approaches as to how data is available or accessible, and may not be fully aligned with DoW data directives, e.g., Visible, Accessible, Understandable, Linked, Trustworthy, Interoperable, and Secure (VAULTIS).
- There may not be one software architect or software lead, and thus development may be more consensus-based.
- Different levels of maturity and unaligned architectures, e.g., monolithic legacy code or modern microservices.
- Systems within the SoS may be operating on different levels of security classification; any constraints imposed by the different classifications should be examined early in any effort to develop an SoS capability.
- Systems within an SoS may be exposed to different types of cyber threats.
- An SoS may be more vulnerable than each individual system, and the weakest link could expose the SoS.

A mature software development pipeline contains software assurance capabilities. The pipeline incorporates guardrails such as automated architectural conformance checks (e.g., SonarQube, CAST) as well as automated testing and regression testing. Feedback from monitoring in production, collecting telemetry, and measuring performance are also part of the assurance processes. Thus, an SoS environment requires close collaboration among development, security, and operations.

SWE efforts may follow one of several acquisition pathways. Many modern software efforts use the Software Acquisition Pathway as described in DoDI 5000.02 "Operation of the Adaptive Acquisition Framework." This pathway is designed for software-intensive systems or for software-intensive components or subsystems, such as those found in independently useful systems within a larger SoS construct. The pathway facilitates rapid and continuous delivery of software capability to the warfighter, integrating modern iterative software development methodologies, such as Agile, Lean, human-centered design, and DevSecOps, to deliver secure and resilient software capability rapidly and iteratively to the end user in the operational environment.

A.5.1 Software Challenges in SoS

Software Technologies

Software technologies, and their applicability to the DoW's ever-evolving capability needs, are changing in typically five aspects: (1) application lifetime (from months sometimes years, to weeks sometimes days), (2) development process (from waterfall to Agile and DevSecOps), (3) application architecture (from Monolithic and N-tier to Microservices), (4) deployment and packaging (from Physical Servers and Virtual Servers to Containers), and (5) application infrastructure (from hosted and data centers to cloud, edge, and hybrid). Incorporation of AI technologies may introduce additional changes in all five aspects.

The DoW's expanded use of several software engineering technologies include, but are not limited to: API, Container, Container Orchestration, Container Sidecar, the Cloud-Native Computer Foundation, Continuous Integration/Continuous Delivery Pipeline, Microkernels, Microservices, Service Mesh, Virtual Machine, and Zero Trust (ZT). Taken together, these technologies make possible faster development, improved security, and increased flexibility in DoW SoS engineering. These technologies permit military powers to compose existing capabilities to form qualitatively new capabilities that are more resilient, react faster, and have greater lethality.

For example, the move from the more traditional monolithic software architecture to a more modular and microservices-based architecture leverages technologies such as APIs. An API is a set of protocols, routines, and tools for integrating software applications. They define how different software components should interact with each other, within a system and across the SoS, allowing developers to create applications that can leverage the functionality of other software systems or internal services. Benefits of APIs for the DoW include improved innovation, improved interoperability, increased efficiency, enhanced security, and better data management. In simpler terms, an API is a way for different software applications to talk to each other and share data. In this regard, APIs offer a means to exploit high-quality data across the DoW, employing a data-centric approach to SoS development (see https://cto.mil/sea/swe).

Artificial Intelligence

AI and ML will continue to influence and transform SWE practices in SoS engineering. AI may be embedded within one or more independent systems that make up the SoS. It can be an integration point to bring multiple systems together, can help in the design of a system, and can assist engineers in exploring alternate approaches. Automation such as AI-based code generation and AI-enhanced automated testing are among the more common uses for AI/ML in development of systems and SoS. In addition, AI may be useful in an SoS environment through its use in code reviews, where it may suggest code improvements. It can also identify possible security issues across software development and integration efforts from the individual independent systems.

Development Processes

Development processes are evolving from traditional document-driven Waterfall processes to the more continuous Agile and DevSecOps processes. DoW Waterfall processes have been structured around “projects” with a defined start and end (typically several years). The processes deliver a product with a useful life of 5 to 10 years (although sometimes they have been left in place much longer), but replacing or rebuilding the legacy product requires another major project entailing significant effort, cost, and risk. Many SoS will employ legacy systems that were developed using legacy processes and will integrate capabilities across multiple legacy and non-legacy systems. Regardless, each system must continue to evolve to remain relevant to the warfighter. In the future, changes will be expected to be more rapid to stay relevant to the current threat environment. SoS engineering needs to be aware of what each system uses for its development processes.

In newer efforts, the emphasis has shifted from "projects" with a defined start and end date to "products" that are considered "never done" and thus continuously evolve to meet the needs of end users. Product and technology life cycles are shorter, and architectures that once were large and monolithic give way to modern, microservice-based architectures. These are oriented around many loosely coupled services developed independently by small, agile teams following Agile and DevSecOps methodologies. This approach can deliver small batches of capability to the end user in short delivery cycles measured in days and weeks (Software Engineering for Continuous Delivery of Warfighting Capability (SWE Guide) 2025).

This short cycle approach enables a feedback loop replacing excess effort devoted to prediction with a more efficient approach relying on experimentation and iteration. Rather than avoid all risk, the experimental approach allows one to fail fast once and pivot to better solutions with a savings in time and effort. Such an approach harmonizes naturally with an SoS context in that integration with other systems can proceed incrementally and opportunistically in response to evolving mission needs.

Configuration Management

Software within an SoS's constituent systems may be continuously updated. Software changes in one system may affect other systems through interdependence between them, and consequently, may affect the overall SoS. Thus, baseline changes within each system may need to be coordinated across the SoS to ensure the overall SoS capability does not "break" when an update is released for one constituent system.

Increasing Use of Automation

Rather than relying on manual testing mechanisms at the end of development, modern software testing is automated and occurs on an ongoing basis, at the speed of the development team. Rather than manually installing software on physical servers, developers may bundle applications into pre-integrated containers, which can be dropped onto standard, highly virtualized infrastructure that can adapt to meet the needs and demand of users. Processes for code and infrastructure builds, integration, testing, release, and deployment all can be automated. These innovations enable software engineers to develop and field applications in months or days rather than years (SWE Guide).

User Engagement and Feedback Loop

The consistent integrated involvement of the end user or warfighter is key to ensuring that any frequent re-prioritization of requirements or capabilities based on value assessments during the software engineering process remain grounded on the need to deliver user-defined capabilities. Within the Agile or DevSecOps methodology, these feedback opportunities with the end user offer the technical and technical management members of the independent systems to coordinate and collaborate with the overall SoS engineering effort. Leveraging the retrospectives in Agile sprints or the continuous monitoring phases of the DevSecOps approach can contribute to delivering capabilities of value to the warfighter. SoS engineering feedback may be constrained, and special efforts may be needed to coordinate feedback across the SoS.

A.5.2 Other SWE Considerations in SoS

With composability of systems, the criticality of functions may shift within each of the systems in the SoS. As a result, cybersecurity analysis may need to be revisited to account for these changes. Cybersecurity in a SoS should extend beyond software considerations to include hardware, communication paths, people, processes, and technology, all of which contribute to the overall risk profile. To ensure mission assurance, cyber resilience performance requirements should be established for the SoS. These requirements must be measurable, testable, and aligned with mission performance needs, ensuring that the SoS can maintain critical functions under cyber threats and adversarial conditions. In current operations, the ability to apply real-time patches to software in the SoS allows quick and effective mitigation of adversary cyberspace attacks that could disrupt the SoS capabilities.

As further reference, SoS systems engineers or PMs may find the SWE Guide useful. Specifically, the section on “Overcoming Challenges with Best Practices” describes challenges and best practices for stages of the software development life cycle (SDLC) such as Requirements, Software Architecture, Design, Coding, Testing, System Integration and Test, and Operations.

- Integration and Interoperability: Since SoS involve systems that are often independently developed, integrating them into a cohesive whole can be challenging. There may be differences in technology, communication protocols, and data formats. Each system must be able to count on the other systems of the SoS to produce data according to the established data exchange expectations or APIs.
- Complexity: The sheer scale and diversity of the systems involved make SoS design and management highly complex. Modeling, testing, and verification become non-trivial tasks.

- Emergent Properties: SoS exhibit behaviors or properties that cannot be predicted from the individual components alone. New opportunities for innovation may exist and new approaches for discovery learning may be needed to leverage the opportunities. Emergent properties may introduce new vulnerabilities. Agile development environments can best take advantage of these opportunities.
- Scalability: As SoS grow (e.g., adding more systems, increasing complexity), ensuring that the overall system continues to meet performance and operational requirements is a challenge.
- Governance and Coordination: With decentralized ownership and control, coordinating changes, updates, and decisions across multiple stakeholders is difficult.
- Security and Privacy: Securing an SoS is more complicated than securing a single system, especially when sensitive data must be shared across independent systems. Different systems may have different security policies and requirements.
- Software Compatibility: In an SoS environment, individual systems build their own software systems independently. Integrating them to meet the higher mission objectives can be challenging mainly because of incompatibility issues in the event systems are not designed to work well with each other or with the SoS operating system. In addition, new software often requires the latest operating system version. If the operating system is outdated, it might not support the new software's features or security enhancements.

A.5.3 Best Practices for SWE in SoS

- Early and Continuous Integration: Integrating components early in the development cycle may help uncover compatibility issues and can reduce integration problems later.
- Clear Interfaces and Protocols: Standardizing communication interfaces and protocols between systems is essential for ensuring smooth interoperability. An API-first strategy should be considered in achieving SoS objectives.
- Modular and Scalable Architecture: Designing systems in a modular way allows for more efficient addition, removal, or replacement of components as the SoS evolves. Software should consider use of microservices, containers, Kubernetes and other approaches to modularity. A loosely coupled and cohesive SoS architecture should be emphasized as the SoS evolves.
- Continuous Monitoring and Feedback: Consistently monitoring the behavior of the system and collecting feedback allows for adaptive adjustments, which is critical for maintaining performance and stability. There are many concerns, such as, but not limited to: operational data, performance data, data security, latent defects, network traffic loading, etc.

- Stakeholder Collaboration: Since SoS typically involve many different stakeholders, regular communication and collaboration between them are necessary to ensure that each component aligns with the SoS's goal. Each individual system has its own set of requirements that may not take into consideration the SoS.
- Cybersecurity: Early integration of security for SoS development includes ZT as a foundational process. The basic principle here is to protect data at its point-of-presence. Do not use networks to secure data. ZT presumes that all networks are breached. To ensure effective implementation of ZT across the SoS, it is critical to address how ZT principles are integrated into legacy and existing systems that will become part of new acquisitions. This requires the development of new requirements and the execution of engineering processes across the various systems to achieve comprehensive ZT implementation.
- Data Centric: High-quality data is a strategic asset and needs to be treated as such. In an SoS, data is of primary importance for the interoperability of combined systems. Appendix C provides more detail for what it means to be data centric; this is foundational for building any system.

A.6 Value Engineering in an SoS environment

Value engineering (VE) is defined in the SE Guidebook as "a systematic approach that analyzes the functions of systems, equipment, facilities, services, and supplies to ensure they achieve their essential functions at the lowest life cycle cost consistent with required performance, reliability, quality, and safety. VE is performed in accordance with DoDI 4245.14 to eliminate or modify any element that significantly contributes to the overall cost without adding commensurate value to the overall function." It is one of many tools for "increasing value to the warfighter" that can be used to "improve military worth or reduce acquisition and ownership costs wherever it is advantageous." The functions across systems or within systems of an SoS can likewise be analyzed to achieve greater war fighting value and reduce resources for the same value, or both.

Incorporating VE adds value to the development and sustainment of SoSs. The strength of the VE methodology is the ability to analyze system information to develop functions that describe that current system. Formatting requirements, processes, or product attributes into functions promotes enhanced generation of alternatives that a team can effectively evaluate to produce value-added, cost-effective solutions. Therefore, VE can assist in analyzing requirements and addressing issues at both the SoS and constituent levels throughout the life cycle. It can specifically enhance certain SE processes that are applied to SoSs and that are defined in the SE Guidebook:

Completed VE initiatives are submitted for implementation using either a Value Engineering Proposal (VEP) or a Value Engineering Change Proposal (VECP). VEPs are developed through the application of the Value Methodology and submitted by individual civilians or contractors providing support within the government. VEPs may or may not save dollars but must improve the value of the item under study. VECPs are submitted by contractors to the contracting officer under the VE clause of a contract (FAR 52.248). FAR (Subpart 48.102, para (a)) requires the contracting activity to include VE provisions in appropriate supply, service, architect-engineer, and construction contracts. A VECP must generate net acquisition savings and requires a change to the instant contract to be implemented. As a result, the government shares a portion of net acquisition savings with the contractor according to FAR guidance. For more guidance on VE please refer to SD-24.

A.6.1 Technical Management Processes

- Technical Planning: The SoS engineer, together with the systems engineers, can incorporate the methodology to effectively scope the effort required to develop the SoS and to develop plans that optimize available resources of constituent systems.
- Requirements Management: VE can assist analyzing SoS requirements to ensure cohesion with constituent systems' requirements throughout the life cycle (i.e., recognize redundant requirements to mitigate conflicts).
- Risk Management: VE can be used with risk mitigation strategies to develop an effective risk mitigation plan that addresses cost, schedule, and performance objectives.

A.6.2 Technical Processes

- Stakeholder Requirements Definition: VE can ensure that only the necessary technical requirements are derived from the stakeholder capability needs.
- Requirements Analysis: VE can assist in a similar manner to the stakeholder requirements definition process, but with emphasis on the end-user needs. It can also ensure that the correct allocation and derivation of SoS requirements flow down to the system levels.
- Architecture Design: VE can be used to develop alternative design solutions and to evaluate these solutions to select the most effective final design. The application of MOSA during architecture design presents an opportunity to apply the VE methodology through the analysis of alternative modular partitions in the design to reduce life cycle cost.

A.7 Cyber Considerations

Cyber threats pose significant risks to the integrity, availability, and confidentiality of SoS architectures, especially given their complexity and interdependencies. SoS and their constituent systems must be engineered to ensure resilience in contested, degraded, and adversarial environments, enabling mission-critical functions to persist despite disruptions or operational challenges. Cyber considerations for SoS are recommended for inclusion as an integral part of the SoS development process. In SoS development, cyber and system security play a particularly important role in:

- Engineering and architecture design trade-offs
- Interface, integration, and interoperability considerations
- Data exchanges
- Emergent SoS capabilities not present in constituent systems
- Adaptability
- Flexibility
- Survivability
- Resilient performance capabilities

The following are vulnerability areas accentuated in SoS architectures that cyber threats can exploit:

- Interfaces
- Data exchanges
- Interoperability APIs
- Boundaries between systems
- Gaps in design coordination (intentional or otherwise)
- Architecture constraints/inefficiencies, or simple ineffective performance due to misaligned requirements
- Incorrect implementation
- Missing needed capabilities (often caused by funding constraints or misalignments)

Addressing these vulnerabilities during the engineering process increases the likelihood that SoS architectures are designed to be secure, resilient, and capable of performing critical functions under adversarial threats. SoS engineering should implement continuous, integrated T&E to manage and validate the cybersecurity and resilience of the composite SoS in a contested environment.

The primary mechanism for identifying and prioritizing cyber risks at the SoS level is the MBCRA. The SoS engineering team, supported by a cyber working group, should conduct MBCRAs that consider the aggregated risk from constituent systems and the emergent risks created by their interactions.

A.8 Resource Management Considerations

For Resource Management, the SoS engineering team:

- Identifies, defines and documents resources shared across the SoS. Resources are those inputs required to support effective operation of the SoS and its constituent systems, including but not limited to energy, cooling, manpower, and infrastructure.
- Defines resource specifications that fully describe the required support and limitations of the SoS and constituent systems to ensure adequate support and operation for the SoS.
- Facilitates cooperation and agreement among all stakeholders to establish resource boundaries for constituent systems and the SoS.
- Continuously monitors changes on internal and external interfaces and dependencies between constituent systems and the SoS and reviews changes with SoS managers, stakeholders, and constituent system owners through an established resource management process.

Appendix B: SE Processes in an SoS Environment

B.1 SE Technical Management Processes in SoS Environment

This section discusses the eight SE technical management processes defined in the SE Guidebook as applied in an SoS environment. The processes provide a framework for technical planning, decision analysis, technical assessment, requirements management, risk management, configuration management, data management, to manage system or SoS development throughout the acquisition life cycle.

Table B-1. Technical Management Processes as Described in the SE Guidebook

Technical Management Process	Description
Technical Planning Process	The process provides a framework to define the scope of the technical effort required to develop, deploy and sustain the system, and provides critical quantitative inputs to program planning and life cycle cost estimates.
Decision Analysis Process	The process transforms a broadly stated decision opportunity into a traceable, defendable and actionable plan. It encompasses one or more discrete analyses at one or more lower levels and aggregates them into a higher-level view relevant to the decision maker and other stakeholders.
Technical Assessment Process	The process provides a fact-based understanding of the current level of product knowledge, technical maturity, program status and technical risk by comparing assessment results against defined criteria.
Requirements Management Process	The process maintains a current and approved set of requirements over the entire acquisition life cycle. This helps ensure delivery of a capability that meets the intended mission performance, as stipulated by the operational user and it provides requirements traceability back to user-defined capabilities.
Risk Management Process	Ensures program cost, schedule, and performance objectives are achieved at every stage in the life cycle and to communicate to all stakeholders the process for uncovering, determining the scope of, and managing program uncertainties.
Configuration Management Process	The process establishes and maintains the consistency of a system's functional, performance and physical attributes with its requirements, design and operational information and allows technical insight into all levels of the system design throughout the system's life cycle.
Technical Data Management Process	The process provides a framework to acquire, manage, maintain and ensure access to the technical data and computer software required to manage and support a system throughout the acquisition life cycle.
Interface Management Process	The process provides a framework to identify, define, manage and ensure compliance with internal and external system interfaces. The Interface Management process helps ensure that developers capture all internal and external interface requirements and requirements changes in accordance with the program's Configuration Management Plan.

Source: SE Guidebook

B.1.1 Technical Planning Process in SoS Environment

Technical planning activities ensure that systems engineering processes are applied properly throughout a system or SoS life cycle. Technical planning, as opposed to program planning, addresses the scope of the technical effort required to develop the system or SoS. A mandated tool for system technical planning is the SEP. While policies do not explicitly discuss SoS, SoS managers should apply key systems engineering tenets: assign a lead SoS (systems) engineer, develop a technical plan (EEMP as discussed in section 2.9), and conduct technical reviews that involve independent SMEs.

Technical planning can be more difficult for SoS than for single systems because the SoS engineering team is required to plan the evolution of the SoS in the context of the independent technical plans for the constituent systems. The highly asynchronous, parallel nature of system-level engineering activities makes good planning and coordination, and management of SE processes and products, more critical at the SoS level. Systems engineers from constituent systems are already performing technical planning for their own systems, and SoS technical planning should consider and possibly augment the plans of those constituent systems. SoS technical planning must be adequately resourced because of the inherent competition with the individual programs for systems engineers' attention. To appropriately mitigate risk to the SoS effort, SoS systems engineers must actively engage system-level systems engineers in SoS technical planning. Some SoS programs form an SE council or body to address cross-cutting SoS planning.

Note: Either an EEMP or SEP is acceptable if it addresses the technical planning necessary to meet SoS objectives.

B.1.2 Decision Analysis Process in SoS Environment

In an SoS environment, the SoS systems engineer addresses issues concerning alternative ways to meet SoS capability needs through available systems or new systems. Throughout SoS evolution, the SoS systems engineer decides how to adapt, extend, and augment the current ensemble of systems to meet user capability needs. Factored into these decisions are the approaches and costs for transition and sustainment. In this context, the systems engineer supports decision making with quantitative and qualitative data analytic methods.

In larger SoS involving multiple legacy systems, it is important to understand how coupling multiple systems affects the behavior of the systems and the SoS, particularly unanticipated emergent behavior and indirect effects. Feasibility evidence derived from M&S, collaborative efforts of SMEs, and focused experiments can be used to address these and other SoS issues.

Because SoS decisions may have implications for constituent systems, SoS decision analysis needs to explicitly consider the perspective of affected systems, stakeholders, etc.; however, time and resources are often at a premium for systems engineers of the constituent system. This may limit the level of involvement by the constituent system SE teams. Consequently, the SoS systems engineer may need to anticipate the issues that will affect the constituent systems and include an assessment of them as part of the SoS decision analysis.

Finally, the SoS systems engineer is challenged to develop approaches to evolve the ensemble of systems to meet new needs while accommodating the independently owned and funded constituent systems, which themselves are often evolving to meet their own system users' needs. To attain this delicate balance and support decisions that are typically outside of the SE purview, the SoS systems engineer must understand systems and their relationships from multiple perspectives, including technical and organizational relationships.

These decisions include analysis of options and trades for SoS design given current characteristics and development plans of systems; assessments to determine which requirements can be addressed in what time frame given system objectives, funding, and development schedules; and analysis of how internal and external changes may affect the SoS. Several activities, including the Software Engineering Institute's SoS Navigator initiative, are examining these needs and approaches (Brownsword et al., 2006).

B.1.3 Technical Assessment Process in SoS Environment

In SoS, technical assessment addresses both technical progress at the SoS and system level. In SoS, technical assessment of progress addresses two areas: (1) progress toward meeting SoS capabilities and (2) progress toward implementing changes and upgrades to the SoS, including changes in systems and inserting new systems into the SoS.

In the first area, because the SoS engineering team typically addresses user capability needs by adapting multiple systems and technology insertion over time, it is important to develop user-oriented metrics that can be applied across venues to assess progress toward meeting these objectives and to collect data to assess this progress. Although in most cases at least some of the systems in the SoS already exist at the time the SoS is recognized, the metrics should be independent of the specific systems because specific constituent systems may change over time.

In the second area, as plans for SoS upgrades are developed and implemented, the SoS systems engineer needs to assess progress in defining, planning, implementing, integrating, and testing the changes made to affect the upgrade. This assessment includes technical assessment of the changes in the individual systems that will be planned and implemented under the auspices of the program management and systems engineers of the constituent systems. In defining upgrades, the maturity of technologies to be incorporated is particularly critical in an SoS environment. Indicators of maturity include metrics such as version stability.

The SoS systems engineer needs insight into the system-level work, but ideally system-level work is planned, implemented, and assessed as part of the constituent system's SE process. Whether a member of the SoS engineering team participates in the system reviews or the systems engineer for the constituent systems provides updates to the SoS systems engineer, technical assessment is based on the resources available and the criticality of the changes to the SoS. Good SE practice requires the SoS systems engineer to follow a disciplined technical review process for the SoS as defined in its EEMP.

The SoS systems engineer is specifically interested in system implementation progress that affects the SoS functionality, performance, or schedule (this is akin to the importance of critical IMS synchronization points to SoS engineering) because these issues could be a source of risks for the SoS. Assessment encompasses functionality in the systems and the interfaces between each system and the other systems in the SoS to implement the SoS thread, including data communications and data use.

The SoS technical assessment includes assessing technical progress of integrating, testing, and evaluating the composite SoS. The SoS technical plans identify key decision points where technical reviews should be conducted, including the criteria for those reviews. Technical reviews should address plans for integration and T&E, including when and where they will occur and the risks associated with them. The SoS systems engineer is responsible for technical reviews with active participation of the constituent systems engineers to the degree that these reviews can leverage integration and T&E events planned and implemented by the constituent systems. In general, incorporating SoS assessment into system-level events is a preferred approach for SoS efforts.

The challenge in this area is planning and implementing in the context of the asynchronous development schedules of the systems. This means that if systems *a*, *b,* and *c* all make changes for an SoS improvement, changes in these three systems will be implemented and deployed under the development schedules of the systems. Problems arise when system *a* develops and fields before *b* and *c* are ready for integration and T&E. An approach is needed to assess changes in system *a* without availability of changes in *b* and *c*, and to manage the risks in this asynchronous approach. This may affect SoS design, which needs to be tolerant of new functionality without full implementation of the functional thread. This may also increase the burden of accommodating risk mitigations in the later systems. M&S may be useful in addressing situations such as this, where a simulated version of changes in *b* and *c* could serve as a surrogate for system *a* integration.

B.1.4 Requirements Management Process in SoS Environment

As discussed in section B.2.1, Stakeholder Requirements Definition, the SoS systems engineer is an active participant in the development of requirements based on SoS capability objectives and must consider not only requirements at the SoS level but also requirements of users of the constituent systems. Requirements management begins with the developed SoS requirements and traces the SoS requirements throughout the process and over time. Requirements for the constituent systems typically will be managed separately for each system by its systems engineer using their own processes. At a minimum, the SoS systems engineer needs to be informed about these processes. In addition, he or she should ensure that new requirements on systems to meet the SoS needs are reflected in the systems' requirements management processes in a way that is linked to SoS requirements management.

The SoS systems engineer needs to recognize when there are redundant or conflicting requirements across constituent systems. In an SoS context, redundancy across individual systems may be acceptable, desirable, and even necessary when considering the roles that individual systems play apart from the SoS. In some cases, duplicative requirements or functionality across the constituent systems may cause SoS conflicts. For example, when multiple systems in an SoS each have different methods of computing track correlation, the combined results provide poor estimates of enemy targets. It may be important to manage and resolve any conflicts, but it may be too costly or disruptive to attempt to back out contentious, redundant requirements or functions.

Requirements management in the classical sense is just as critical to the success of the SoS. SoS systems engineers need to define requirements in terms of measures of outcome and mission measures of effectiveness to derive SoS measures of performance that can then be allocated to individual systems as part of the SoS process across the relevant SoS engineering elements; however, there are some unique challenges for requirements management in an SoS.

In an environment of evolving threats and an evolving CONOPS, a critical aspect of the requirements management activity is the identification and management of new requirements over time, and the correlation and traceability between the desired capabilities and the configuration of the deployed SoS. The requirements management function must support these activities in a flexible and agile manner. Furthermore, although requirements management may focus on specific functional requirements of the SoS and individual systems, it is also important to address and manage the communications and data exchange requirements in the context of the SoS.

B.1.5 Risk Management Process in SoS Environment

Risks identified and managed by the SoS engineering team are those related to the SoS itself and its mission objectives. SoS risks are often tied to the strength of feasibility evidence developed during the decision analysis and design solution activities. These risks might be related to SoS scalability, quality of service, technology maturity, coordination of SoS risk management activities across the individual systems, ability of constituent systems to provide needed SoS functions on time, and other individual system risks that might affect the overall SoS success or other individual systems. Table B-2 summarizes potential problems identified in dealing with the risk management process at the SoS level. The table is adapted from "Risk Management for System of Systems" (Conrow 2005). The table has been slightly modified to describe possible risk management problems associated with a notional SoS.

Table B-2. Potential Problems in Managing Risk at SoS Level

Potential Risk Management Problem	Problem Description
Dealing with Multiple Stakeholders	An SoS consists of multiple stakeholders and the need of each stakeholder can vary. In addition, the associated cost, performance, and schedule factors can be different for different stakeholders. These differences often lead to contention and potentially sub-optimal design solutions, funding allocation, schedule priority, and increased risk.
Multiple System-level Risk Management Processes	An SoS deals with multiple system-level risk management processes. Ideally, these processes should be highly compatible but often they are not. This can contribute to weak or an ineffective risk management process at the enterprise SoS level. Risk management differences between systems and organizations can lead to omission of some risks that could impact the performance of enterprise SoS.
Long Enterprise SoS Life Cycles	An SoS can be expected to have long life cycles, ranging from many years to decades. The individual programs may have different levels of maturity varying from early development to operations. The resulting non-uniform acquisition maturity potentially complicates risk management at the SoS level.
Multiple Technical Risk Management	Risk outside of programmatic (cost and schedule) risk can be classified as technical risk. As a general practice, program level technical risks can be grouped into risk categories. Technical risk categories can exist from low Work Breakdown Structure (WBS) levels to the program level (WBS level 1) or enterprise SoS level. Several of these risk categories are not explicitly evaluated during program risk management processes and can potentially complicate technical risk management at the SoS level.
Integration Risk Management	Integration risk is present on many Military Service programs and can be felt throughout the SoS. System-level hardware/hardware, hardware/software, and software/software are common forms of integration risk. In addition, new forms of integration risk such as data-centric integration issues not commonly seen at the individual program level may occur at the SoS level. The potential level of integration risk can be substantial because of a tendency to underestimate integration difficulty as well as overestimate the maturity of items that require integration. This is even more problematic when integration risks manifest late in the program life cycle.
Multiple Functional Performance Risk Management	A system functional performance may include the ability to demonstrate that desired functions or requirements can be met at specified performance level. System functional performance risk is rarely estimated during development phase (this is different from system design risk, which generally assumes that a requirement can be met by the nature of the design). Initial modeling, simulation, and emulation followed by appropriate incremental demonstrations, prototyping, and testing can be helpful throughout the development and integration cycle to potentially reduce functional performance risk to an acceptable level. If such important SE process is not taken early in system development phase, functional performance short falls can translate to risk late in a system life cycle. Furthermore, if insufficient progress is made in demonstrating the performance level of key functions that can be achieved by individual system, this can impact achieving mission-level capabilities in the SoS architecture.
Multiple Interface Complexity Risk Management	Complex hardware and software interfaces often exist within individual Military Service programs. While there may be a desire to explicitly treat complexity in a risk analysis, it is generally difficult to accurately relate complexity to risk. Furthermore, efforts to estimate the risk of interface complexity directly may lead to uncertain, subjective, and/or erroneous results. Interface complexity is typically related to the probability of occurrence term of risk and unrelated to consequence of occurrence. This can be difficult to analyze using known risk assessment methodology.

Source: Adapted from Conrow 2005.

In addition, multiple system-level operational and cybersecurity risk management issues can contribute complex problems at the SoS level.

Risk management for an SoS begins with the identification of SoS objectives and the identification of the risks that threaten the achievement of those objectives. While it is true that minor individual program risks could be major risks to the SoS, it is also true that significant system risks may have little or no impact on the SoS functionality. Furthermore, there may be risks to a set of SoS objectives that are not risks to the constituent systems (e.g., unwanted emergent behavior, infrastructure, integration risks, cost risk).

Major risks associated with SoS may relate to the limited influence the SoS systems engineer may have on the development of critical individual systems in addition to technical risks associated with those individual systems and platforms. Independent evolution of the individual systems can lead to unforeseen deviations from SoS program objectives (life cycle cost, performance, schedule). Each of the core SE management processes can identify and support risk assessment. As addressed in section B.1.3, Technical Assessment, the SoS manager and systems engineers must understand each individual system's planned evolution. In some cases, mitigation strategies for SoS can include preplanned substitutions of individual systems, especially if some of the systems are reaching their service life and may be retired, undergoing Service Life Extension Programs, remanufacture, and so on; however, in many cases, it may not be an option to replace high-risk or problematic individual systems, and risks associated with these systems need to be addressed in other ways.

Special care should be taken in evaluating the incorporation of legacy systems in an SoS, particularly those with incomplete technical documentation. Although subsystem risks may not have a significant impact on the parent individual system, they could constitute a major impact on the SoS and may require different approaches to calculate or buy down risks accumulated across multiple systems. Programs should avoid using only the risk criteria already employed at the system level but instead should develop impact criteria and ratings at the SoS level. The criteria should be updated over time to reflect risk tolerance at the SoS level.

Among other measures, an integrated RMB should be established with members from individual systems encouraged to participate; however, it may be difficult to persuade members from individual systems to participate in the SoS-level risk board since it is not their primary focus. The board can look across the SoS and its objectives as the basis for identifying and assessing risk to the SoS. A senior person from the SoS organization should lead the effort to ensure the necessary chain of command.

Since the initial articulation of SoS objectives may not support detailed requirements development, early experimentation focused on military utility and worth can be an important risk-reduction activity. For more detailed information on risk management, see the DoD Risk, Issue, and Opportunity (RIO) Management Guide for Defense Acquisition Programs (2023).

B.1.6 Configuration Management Process in SoS Environment

CM should be applied to the establishment and management of the SoS technical baselines (functional, allocated, and product). It is important to manage these baselines at the SoS level so systems engineering can help structure and control the SoS evolution over time. These baselines also can be used by the constituent systems as they consider changes to their own configurations. A CTC and CCB should be established to manage these baselines.

SoS CM requires an understanding of the systems that support the SoS objectives and their relationships. For the SoS to be successful, the SoS systems engineer needs to have a good understanding of the constituent systems, their characteristics that are salient to the SoS, and the way they currently work together to address the end-to-end SoS needs. While systems engineers of the constituent systems are responsible for the detailed CM of those systems, those characteristics that affect the SoS are mirrored in the SoS CM. The systems engineers should be able to track requirements between the SoS CM and salient elements of the CM of constituent systems.

B.1.7 Data Management Process in SoS Environment

The SoS data include information on the development plans of the systems and their management and funding profiles as well as other information relevant to SoS progress. A key challenge for data management in an SoS context is gaining access to data from constituent systems in a form that facilitates analysis of cross-cutting issues. This process can be complicated because different systems create and retain different data, and common data may not be readily available across systems. Systems may be reluctant to share data outside of the system context, and some needed data may be considered proprietary and held by developers, classified at multiple levels, or compartmented. These challenges pose issues for cross-cutting SoS decision analysis.

An MOA may be one solution to the SoS data problem. In the MOA, systems engineers might define an approach for SoS data management that includes data access, data use and sharing, and creation of an SoS shared repository for common data, all managed in a way that reassures stakeholders that access to their data will be controlled.

Throughout the SoS engineering process, critical data needs to be captured and understood in the context of SoS activities. As a source of data critical to the SoS may evolve or become unavailable, an understanding of critical data needs may allow the discovery of another data source for the SoS activities. This is particularly important for an SoS because there are more diverse participants in an SoS evolution. Available data on SoS activities will be a key to ensuring transparency in SoS processes across participants at both the system and SoS levels.

B.1.8 Interface Management Process in SoS Environment

In most cases, the SoS provides an end-to-end capability consisting of actions coordinated through the sharing of information across the systems; therefore, interface management is a key activity. Information sharing and interface management are one component of the end-to-end operation of an SoS.

As the DoW moves toward data centricity, the classical interface control discipline is increasingly being replaced by network and web standards. Standards are available in ASSIST (https://assist.dla.mil), and IT standards that have been mandated for use in DoW systems are listed in the Defense Information Technology Standards Registry.

Data and metadata harmonization are becoming the central interface issues, with the result that the focus of interface management will be on data exposure and semantics. In many cases, the SoS engineering must pay more attention to data, data interoperability, and semantics than to interface issues. In most cases, the SoS does not "control" individual system interfaces; rather, the interfaces are "managed" through agreements and negotiation. An individual system may be part of more than one SoS, and consequently interfaces and interface changes may affect more than one SoS. Resources in this area include the DoD Metadata Registry (DMR) and the DoD Net-Centric Enterprise Services Service Registry. Approaches to a particular SoS data requirement can be found in the DMR with a registered DoW service.

B.2 SE Technical Processes in SoS Environment

This section discusses each of the eight technical processes defined in the SE Guidebook as they are applied in the SoS environment. The eight technical processes discussed below provide a framework for ensuring and maintaining traceability between stakeholder requirements, design, and the eventual delivered capability whether it be at the system or SoS level.

Table B-3. Technical Processes as Described in the SE Guidebook

Technical Process	Description
Stakeholder Requirements Definition Process	The process translates stakeholder capability needs into a set of technical requirements.
Requirements Analysis Process	Requirement Analysis results in the decomposition of end-user needs into clear, achievable, and verifiable requirements. The activities support allocation and derivation of requirements down to the system elements representing the lowest level of the design.
Architecture Design Process	Translates the outputs of the Requirements Definition and Requirement Analysis processes into alternative design solutions and selects a final design solution.
Implementation Process	The process provides a system that satisfies specified design and stakeholder performance requirements. The outputs of the process include the detailed design, down to the lowest level system elements in the system architecture, and the fabrication/production procedures of forming, joining, and finishing, or coding for software. Depending on technology maturity, the Implementation process may develop, buy or reuse system elements to render the system.
Integration Process	The process of incorporating the lower-level system elements into a higher-level system element in the physical architecture.
Verification Process	The process provides the evidence that the system or system element performs its intended functions and meets all performance requirements listed in the system performance specification and functional and allocated baselines. The process confirms the system element meets the design-to or build-to specifications. Answers the question, “Did you build the system correctly?”
Validation Process	The process provides the objective evidence that the system capability complies with stakeholder performance requirements, achieving its use in its intended operational environment. Validation answers the question, “Is it the right solution to the problem?”
Transition Process	The Transition process moves any system element to the next level in the physical architecture. For the end-item system, it is the process to install and deploy the system to the user in the operational environment.

Source: SE Guidebook

B.2.1 Stakeholder Requirements Definition Process in an SoS Environment

In an SoS, requirements are developed at two levels. The SoS engineering team addresses requirements across the SoS, and the systems engineers for each system address the system requirements from their users. The SoS engineering team is primarily concerned with the translation of SoS capabilities or needs into SoS requirements that provide the basis for SoS design solutions that also encompass the constituent systems.

In an SoS environment, requirements development requires an understanding of constituent system capabilities, high-level SoS requirements, and the interactions between the two. Because these requirements may be satisfied by an existing system, the requirements should be described in terms of needed functionality and not implementation details, so that alternative ways to meet those requirements can be evaluated. The requirements should evolve so that early experimentation and military utility assessments can be used to enhance the operational community's understanding of the integrated SoS capability to be developed.

Because an SoS typically evolves over time, requirements may change based on both internal and external factors. As a result, requirements development may be an ongoing SoS activity, and the SoS requirements will evolve as well. Stakeholders' requirements definition in an SoS often continues through SoS architecture design and implementation since the architecture generates requirements for systems in the SoS.

During each iteration of SoS development, the SoS systems engineer reviews requirements and seeks to address them with available solutions, factoring in the requirements and development plans of the systems in the SoS. As solutions are implemented, detailed designs are developed for each system that is making changes. The development approach, timing of increments, and the tempo for revisiting the requirements are determined by the SoS manager and SE team based on the characteristics of the SoS.

For new acquisitions, requirements are often developed and validated through a formal requirements process. In some cases, this process applies to SoS, and the SoS is handled under the auspices of an acquisition program. In other cases, SoS capabilities are identified based on feedback from operations, strategic direction, or other drivers with the focus on leveraging existing or developing systems and not new acquisitions. In these cases, individual acquisition programs may contribute components to the SoS, but the SoS itself is not handled as an acquisition.

The major challenge for SoS stakeholders in defining requirements is the complexity of developing requirements for a broad capability within the context of systems that have their own requirements and stakeholders. The stakeholders for an SoS include users and proponents for the SoS, as well as the stakeholders for the constituent systems who may not share the perspective of the SoS. Building a common understanding of SoS needs and approaches with the SoS and constituent system stakeholders is key to SoS success, but building a stakeholder community takes time. In many cases the SoS systems engineer is responsible only for the SoS-level requirements, but constituent system requirements may continue to evolve or change, and these may have an impact on the SoS. At a minimum, the SoS systems engineer needs to remain cognizant of the changing requirements of the constituent systems.

In most cases, the technical requirements for a system or a system increment have been defined and are provided to the systems engineer as a starting point. In an SoS, because requirements may be at a higher level or cast in terms of capabilities rather than requirements, the systems engineer plays an important role—working with stakeholders and the SoS manager to articulate the high-level technical SoS requirements that provide a basis for the SE for the SoS. Similarly, identifying the systems affecting SoS objectives and understanding their technical and organizational relationships goes beyond what a systems engineer typically does to address the interfaces for a new system.

The SoS systems engineer looks across the SoS to recommend requirements to be addressed in an increment, working with the SE teams of the systems to identify opportunities and approaches for addressing the requirements and developing a plan for the increment through analysis and trades. The SE teams for constituent systems examine options for implementing new functionality in their own systems using the same processes they would apply for any type of a requirement. Once a plan is developed, the systems implement the plan while the SoS systems engineer provides oversight and takes responsibility for integration and test across the systems for the SoS objectives.

B.2.2 Requirements Analysis Process in SoS Environment

In an SoS environment, requirements analysis changes from a one-time, up-front process to a more-or-less continuous process. Sources of change, both internal and external to the SoS, are more pronounced and persistent. The result is that the emphasis of requirements analysis in an SoS engineering environment is to accommodate unforeseen change.

For a new system, the systems engineer can begin requirements analysis with a clean sheet to allocate functionality, whereas for an acknowledged SoS, the requirements analysis must consider the functional allocation reflected in the constituent systems of the SoS. SoS requirements analysis focuses on composition of existing functionality to meet SoS needs. In the SoS, the systems engineer focuses the requirements analysis on identifying which systems can support the capabilities that are needed, not on iterating the synthesis and analysis of results until a desirable solution is achieved.

To do this, the SoS systems engineer must understand and assess available systems together with their future development plans (bottom-up analysis). In addition, the SoS systems engineer must understand the needed SoS functionality and how that functionality might be partitioned across legacy constituent systems, systems under development, and systems still in planning (top-down analysis). The SoS systems engineer needs to factor in the degree of difficulty in integrating constituent systems through structured assessments and reviews with users, focusing particularly on legacy systems openness. Less flexible legacy systems may constrain the SoS design and final SoS capability.

B.2.3 Architecture Design Process in SoS Environment

The architecture design process in an SoS environment is more complex than in a single system environment because of the challenges of multiple stakeholders, integrations, and test timelines, and the degree of interface developments. The SoS architecture design process occurs at two levels: the SoS framework-level and the constituent-system level. The SoS systems engineer develops an architecture for the SoS and overlays it on the constituent systems to provide a persistent framework for evolution of the SoS as part of the overarching organizational enterprise architecture. The architecture design in an SoS incorporates approaches to meet specific requirements that typically encompass changes in the constituent systems to enable the SoS-level capabilities. This design process is normally the responsibility of the systems engineers of the affected systems. The results of these processes are reflected at the top level in the allocated baseline for the SoS and in updates to the technical baselines for the affected systems.

An important step in engineering the SoS is to sort out how to allocate the SoS-driven requirements to individual systems. In an SoS, functional allocation is typically based on identifying where needed functions are supported by systems and assessing how to leverage this functionality for the SoS. Functional "mapping" might be a better way to describe this process, which involves considerations beyond the technical (including operational, fiscal, schedule, and other programmatic considerations). It requires coordination with component programs and probably multiple process iterations.

SoS implementation may require the implementation of SoS-specific components needed to meet the SoS objectives. These components frequently take the form of "middleware" or "glue-ware" needed to mediate between legacy systems that were not originally designed to work together. The options for meeting these SoS objectives are the same as those available to meet other requirements: using COTS solutions, leveraging a capability in one of the individual systems, or developing a new component. If a new component is to be developed, it may be managed either directly by the SoS or by another organization (e.g., individual system program management). In either case, this new development is treated like another constituent system of the SoS by the SoS engineering process.

When the SoS architecture is first acknowledged, the SoS design specifications are placed under configuration control as the SoS allocated baseline. The baseline captures the design information as well as the traceability to the constituent systems. The individual systems are responsible for incorporating the allocated SoS design requirements and maintaining their own allocated baseline at the system level. While it is important for the SoS engineering team to have insight into the constituent system baselines and the associated plans for implementing those baselines, the constituent system baselines remain under the control of the individual systems.

During the SoS architecture design process, the SoS systems engineer works with the SE teams for the systems to conduct trade studies at the SoS level. The trade studies assess potential changes in current and planned systems to address SoS requirements. Iterations of the Stakeholder Requirements Definition and Requirement Analysis processes also may be required to achieve a feasible design solution. The best overall SoS architecture may result in constituent system changes that require adjudication and additional iterations of the SoS design.

Stakeholder Requirements Definition, Requirements Analysis, and Architecture Design are highly interdependent activities for an SoS—even more so given the larger number of stakeholders, a (frequently) distributed management structure, an evolving CONOPS, and systems at different levels of maturity. Trade studies, ideally supported by models and simulation, are performed to explore alternative solutions; the studies must consider performance, schedule, and total life cycle cost.

The discussion here and in the preceding section focuses on the development of the allocated baseline (architecture of the SoS and design changes to systems) to support the SoS objectives as reflected in the functional baseline for the SoS. For evolutionary SoS development, the architecture is a key element crossing the SoS increments. If well designed, the architecture, particularly the key convergence points, is persistent across multiple increments, and user functionality can be enhanced by adding or upgrading constituent systems without disrupting the changes made in previous increments.

In order to add or upgrade constituent systems without disrupting the changes made in previous increments, the SoS engineering team can implement a MOSA as an acquisition and design strategy. This strategy would use open standards, architectures, and interfaces to support a modular, loosely coupled, and highly cohesive system structure. Applying MOSA while developing an SoS architecture limits the impact on the individual constituent systems, allowing for changes in functionality and technology in some systems without impact on others or on the SoS objectives. These changes should be reflected in the SoS technical baselines. Architecture management over time and across increments is an important part of the broader SoS engineering process as our understanding of SoS grows. More information on MOSA can be found in Appendix A of this guide.

B.2.4 Implementation Process in SoS Environment

Implementation in an SoS typically takes the form of constituent system changes, which together create new or enhanced SoS capability. The systems engineers and developers of the constituent systems take the lead in the implementation process, and the SoS systems engineer acts as facilitator, negotiator, technical reviewer, and, ultimately, integrator.

The SoS implementation activity is planned by the SoS systems engineer in coordination with the manager of the SoS and the managers and systems engineers of the constituent systems. The constituent systems carry out SoS implementation in concert with their own development, and to the degree possible their system-level processes and supporting activities are leveraged. Because the systems each have their own processes and development schedules and it is typically impossible to synchronize across multiple programs with different contexts, creating a workable approach across systems is a major SoS challenge.

SoS implementations typically involve some type of incremental approach that allows systems to deliver improvements in stages, with the SoS-level improvement contingent on delivery of all the enhancements by the different systems. One way to achieve an incremental approach is a development method characterized as an SoS Wave Model (a.k.a. “bus stop approach”) where incremental changes are delivered at specified intervals (e.g., every 3 months). For more information on the SoS Wave Model, see Figure 3-1 “SoS Engineering Implementers’ View” (page 27) and “An Implementers’ View of Systems Engineering for Systems of Systems” (Dahmann et al. 2011).

If a problem arises and a system misses a delivery, the system developer defers the delivery to the next delivery point (i.e., the next time the bus stops). In this way, the SoS enforces a regular rhythm for the development process, which accommodates the asynchronous nature of the system processes. The asynchronous nature of the constituent system processes poses challenges for integration and T&E as well as the design since pieces of the overall solution may be delivered and even deployed without the full end-to-end capability being in place.

B.2.5 Integration Process in SoS Environment

Integration across the SoS is a core role for the SoS systems engineer. While the systems engineers of the constituent systems are responsible for implementation and integration of changes within their systems, the SoS systems engineer is responsible for integration of the end-to-end functionality and performance across the SoS. Because implementation in an SoS may be asynchronous, integration may be asynchronous as well. A primary use of M&S in SoS is the creation of "stand-in" emulations of SoS components to support integration and test. Integration facilities are a common tool for SoS integration and test, and networked facilities are becoming more common. These facilities provide a venue both for integration testing as the development of different parts of an SoS are delivered and for system-level regression testing after SoS capabilities have been added, to ensure they continue to support their system-level applications.

B.2.6 Verification Process in SoS Environment

As is discussed in section B.2.4 (Implementation), changes to the SoS are typically implemented by the constituent systems. Changes to the systems are documented at the top level in the SoS allocated baseline and reflected in detail in the technical baselines for the systems. Verification is done against these baselines. Verification involves demonstrating that the design meets the design specification and the SoS meets technical performance requirements. Verification at the SoS level must include activities to confirm that cybersecurity requirements, including those for interfaces and data exchange between systems, are correctly implemented. The SoS systems engineer should be cognizant of detailed test plans developed and implemented by the systems and should oversee the results of the testing as it applies to the SoS.

Verification activities might include demonstration, inspection, similarity considerations, and test at the system level. The SoS systems engineer, responsible for SoS verification, oversees the verification process to ensure that the changes meet the needs of the SoS capability and to assess risks to the SoS associated with the constituent system development. The objective is to leverage the constituent system SE processes as much as possible; accordingly, the system-level engineers typically verify that changes made in their systems reflect the changes requested. This verification is normally done as part of the system-level development and as part of SE at the system level.

B.2.7 Validation Process in SoS Environment

Validation of SoS capabilities assesses whether the changes made in the SoS have the desired end-to-end effects. Validation involves demonstrating that the design meets the capability objectives (derived requirements) of the user. The SoS team should ensure that changes key to the SoS are included in constituent systems' T&E plans and should leverage the results of

the T&E. Validation must include end-to-end operational testing in a contested cyber-environment to ensure the SoS can accomplish its mission resiliently.

To the degree possible, T&E is conducted as part of the SoS development process in an environment in which the SoS is tested end-to-end. The goal is to ensure that the changes in constituent systems have the desired effect on the SoS results. This testing may be done in an integration and test laboratory environment or as part of an exercise or a live test. The challenge for the SoS is that the number of systems can sometimes be large, and full live testing can be prohibitively expensive or difficult to schedule in a reasonable time.

To the degree possible, it is advantageous to conduct end-to-end testing in conjunction with testing of the constituent systems, leveraging their investments in time and resources. In some cases, all the constituent systems may not be available; consequently, the SoS systems engineers may need to use simulations or emulations of unavailable ones. These simulations/emulations can be development efforts. Such an approach would need to be planned well before it is needed. SoS systems engineers assess risks to determine how best to conduct validation so that live testing is focused on those areas with the highest risk.

In addition to T&E of changes in constituent systems, there is often an effort to collect SoS performance data from the operational environment. This data can be used to validate the expected performance resulting from changes in the SoS, and they also can identify factors that affect SoS performance. These factors are important. They add a degree of fidelity to the broader use-case environment for the SoS, which may affect, suggest, or illuminate options for future investments.

B.2.8 Transition Process in SoS Environment

SoS upgrades are transitioned to the field by the system owners based on their own processes. Because SoS upgrades are implemented in the constituent systems, it is the owners of those systems who have the responsibility to field and maintain the system with the upgrades introduced to support the SoS. Planning for the life cycle support of the enhanced systems needs to be considered at the time that solutions are being evaluated with the total cost of options including life cycle support, and hence need to be addressed as part of a decision analysis (discussed in section B.1.2).

In some cases, supporting transition can go beyond considerations of the components of the individual systems and may include requirements, like adding overall bandwidth, that are the result of the SoS capability as a whole and need to be considered by the SoS systems engineer. Requirements like these must be identified early, considered in the selection of options, and coordinated by the SoS systems engineer with the relevant organizations.

Appendix C: Data Centricity, Artificial Intelligence and Zero Trust Cybersecurity for SoS

The MBSE System Model is the authoritative source of truth for defining these APIs and the data they exchange. The System Model should be used to:

- Formally define the information and data elements to be exchanged.
- Specify the interfaces (e.g., as SysML ports) and the operations they support, which provide a machine-readable definition for the APIs.

This approach ensures that as the SoS architecture evolves in the System Model, the data and API specifications evolve with it, maintaining consistency and providing a stable foundation for a data-centric environment.

These three topics are grouped here because they are synergistically related to each other. AI operation is very dependent on having ample trusted data. The data-centric environment provides this and ZT provides the necessary data security. When combined with the evolving DoW digital environment to make all information and services machine readable, the user will have unprecedented access and data assurance. The SoS engineering team benefits from this symbiotic relationship as described in the numbered sections below for each topic. These first sections establish a foundation for data-centricity, which the other topics build upon.

C.1 SoS Development for Real-Time Campaign Support

The purpose of this section is to introduce the idea of using real-time concepts to support the warfighter. Essential for real-time SoS are a data-centric approach supported by a ZT Enterprise security construct. Systems to be examined should have digital twin models with APIs to support the integration of systems into full blown SoS models. The real systems need to have APIs or their equivalent for real-time integration for mission engagement. With these conditions met, AI can then be used to formulate solutions for commanders in the field. These concepts are currently being used in the Combined Joint All Domain Command and Control (CJADC2) program, where real-time sensor-to-shooter solutions are being accomplished. In its current configuration, any domain sensor can supply targeting information to any domain weapons platform through a common command and control system in real time.

C.2 Transition from Net-Centricity to Data-Centricity Paradigm

Net-centricity is the ability to provide a framework for full human and technical interoperability that (1) allows all DoW users and mission partners to share needed information in a timely manner and in an understandable and actionable form, and (2) protects information from those who should not have access to it. Net-centricity is primarily hardware-focused, and it was the choice for application before the DoW's recent data-centric implementations for data sharing within systems.

C.2.1 Data as a Strategic Asset

In recent years, DoW has focused on advancing data-centric practices to maintain its competitive advantage. Data is critical to improving performance and creating decision-advantage in all echelons—from the battlespace to the board room. To support this effort, DoW leaders must ensure that all DoW data is visible, accessible, understandable, linked, trustworthy, interoperable, and secure. Data is a strategic asset, and part of being a "data-centric" organization means treating it as such. To support this effort, DoW leaders must ensure that all DoW data is visible, accessible, understandable, linked, trustworthy, interoperable, and secure (VAULTIS).

- Visible: Consumers can locate the needed data.
- Accessible: Consumers can retrieve the data.
- Understandable: Consumers can find descriptions of data to recognize the content, context, and applicability.
- Linked: Consumers can exploit complementary data elements through innate relationships.
- Trustworthy: Consumers can be confident in all aspects of their use of the data for decision making.
- Interoperable: Consumers and producers have a common representation and comprehension of data.
- Secure: Consumers understand data safeguarding responsibilities, follow classification management procedures, and know that data is protected from unauthorized use and manipulation.

The transition to a data-centric framework also allows the warfighter to access information from anywhere, at any time, at any speed, thereby eliminating the need for the user to be connected to the hardware on which the data is stored or available for use. In short, the transition from a network-centric to a data-centric paradigm means that the network must accommodate the data, rather than the data needing to accommodate the network.

In 2023, DoD released its Data, Analytics, and Artificial Intelligence (AI) Adoption Strategy (DoD DAAIAS 2023), which outlines how DoW will strengthen the organizational environment in which DoW deploys data, analytics, and AI capabilities for enduring decision advantage. As part of this, the document describes the transition to a data-centric framework for SoS development.

C.2.2. Zero Trust

The ZT Strategy, a new form of DoW-wide guidance, adopts a mind-set of "zero trust" regarding adversaries' access to data. A ZT mind-set assumes that all systems are breached, and that data must protect itself. ZT is more than an IT solution, and it extends throughout DoW's capabilities, technologies, solutions, and processes to include staffing, training, and professional development processes as well.

To support the shift to ZT, the Federal Government is moving to a ZT architecture, which attempts to secure data not through physical systems but through a least-privileged-dynamic, software-driven, point-of-presence rule-based development. Physically based net-centric systems use hardware and software–operated systems to secure data, whereas front-end process try to prevent access to the system. Data-centric principles are applied in new ways to support data self-protection.

Underpinning data-centricity is the move to a digital environment. This concept expects most network content to be machine readable, which enables AI, data, and services sharing in a ZT cyber environment. As an example: information system architectures now use DoW Architecture Framework viewpoints to depict information about the architecture instead of providing a machine-readable model that can generate the same information. The viewpoints are text descriptions of the data, whereas the model, with instruction on how to use it, can supply the information in a machine-readable format along with the ability to extract additional information as needed. This model-based engineering approach is gaining preference in DoW systems.

The environment created through a digital environment based on ZT, data-centric, and AI principles will aid the SoS engineering team to develop both existing and real-time solutions for the warfighter. The SoS systems engineer should use these principles in all phases of development and sustainment.

Potential SoS needs to be developed in a machine-readable environment where engineering models of the various systems can be combined simply using APIs. Since SoS are amorphous structures with usually poorly defined boundaries, modeling individual system behavior becomes more important and needs to be machine readable.

With a digital environment, applying data centricity, ZT, and AI concepts becomes more viable. Data-centric architectures enable data integration regardless of the network and ZT guarantees its integrity. The use of AI is dependent on large amounts of data and services that are machine readable. The SoS systems engineer now has a more robust toolset to configure SoS existing and in real-time to support commanders in the field.

The conversion from a net-centric to a data-centric approach establishes the use of communities of interest to solve high-priority data, information, and services issues facing the DoW. At the same time, SE is trending away from engineering point-to-point interfaces and toward exposing data to the enterprise in a common vocabulary, built around key principles:

AI requires large amounts of data to be useful. Access, security, and quality of data are paramount to have a useful man-machine interface that can be trusted.

Furthermore, strategies shall establish the goal of accomplishing this information exchange by exposing services to the enterprise. A fundamental tenet of the services approach is to expose information through well-defined APIs that are independent of the implementation of the service. This tenet results in a looser coupling of the systems in an SoS and enables relatively autonomous evolution of the constituent systems.

The DoW approach to net-centricity and data-centricity is relevant to SoS of all types. The process of networking multiple systems to support the user capability is a common element of almost all SoS. In addition, standards identified for use in DoW systems make the assumption that data-centric policies and practices will be applied as appropriate throughout the SE process for SoS in the future.

C.2.3. Data-Centric Concepts

Figure C-1, Layered Architecture, illustrates a typical data-centric application structure in which the data is managed in the data layer. The figure is an example of a data-centric application and is not the only way to implement this concept as most man-machine interfaces are manifested through some application that is fit-for-purpose.

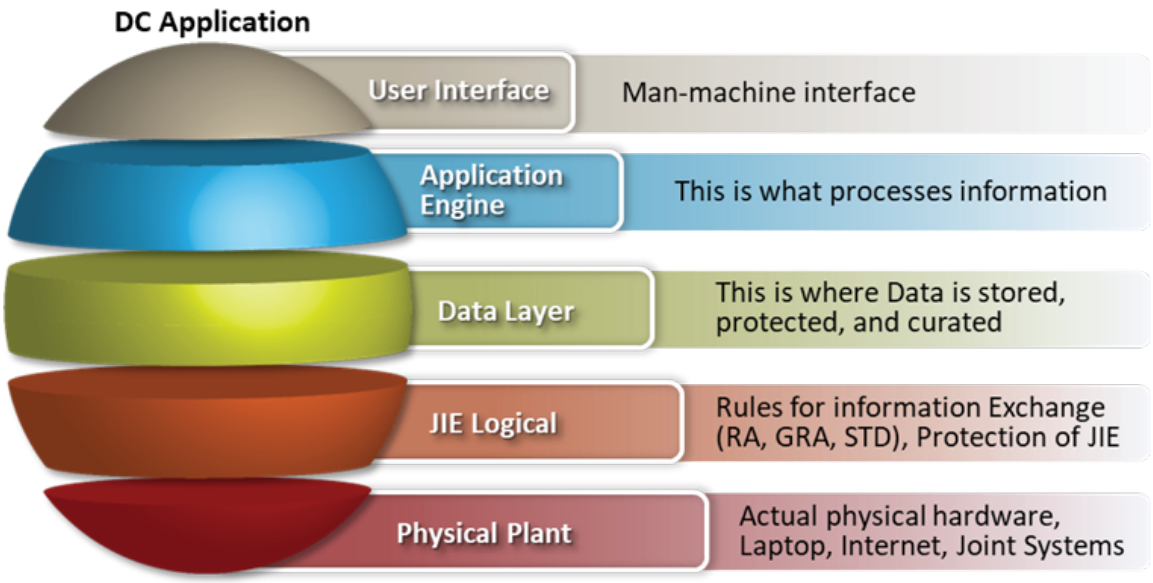


Figure C-1. Layered Architecture

Figure C-2 illustrates how multiple departments share a common database to increase efficiency for the enterprise. This approach works well when the organization can mutually agree on a common database structure.

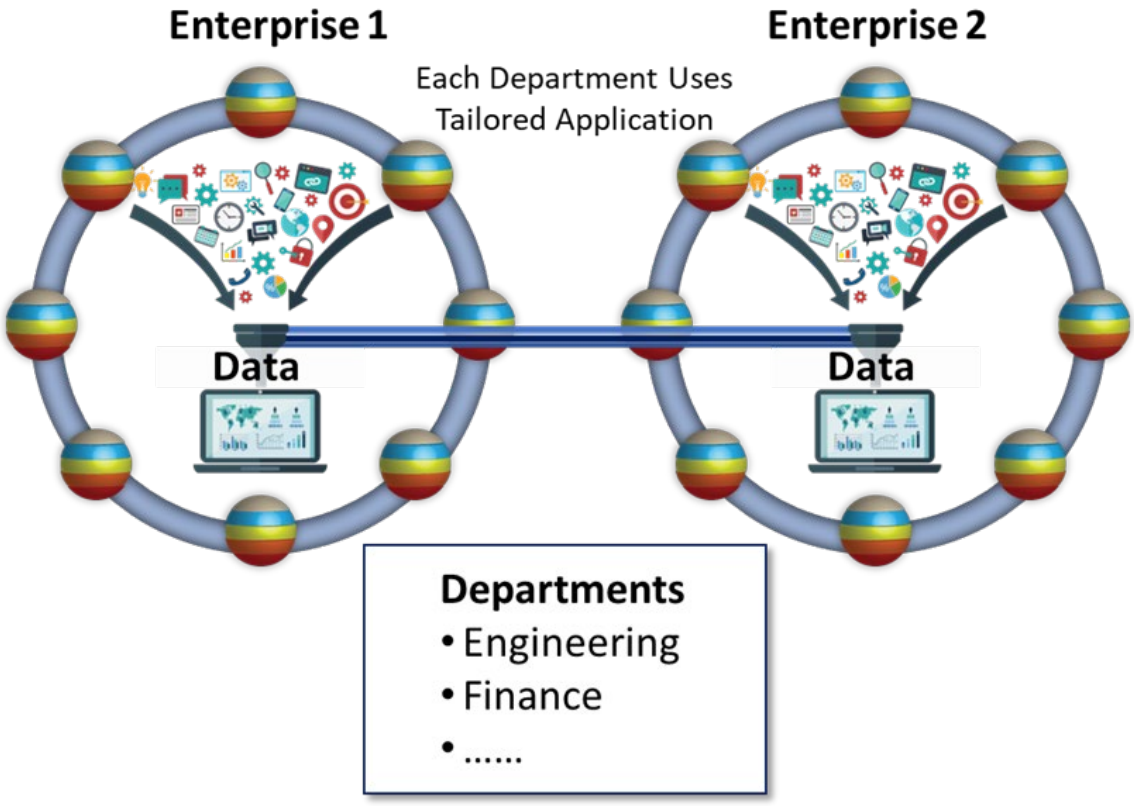


Figure C-2. Data-Centric End State

Figure C-3 illustrates the current solution for sharing databases across different enterprises. The API is a connection between computers or between computer programs. It is a type of software interface, offering a service to other pieces of software. A document or standard that describes how to build such a connection or interface is called an API specification. The DoD API Technical Guidance (April 2025) can supply a much more in-depth description. This is the glue that allows one system to share data between multiple systems without the need for a dedicated interface between each system, as is necessary in a net-centric system. This avoids the problem of defining an interface for each connection necessary to make complex systems work. The API basically provides instruction on how to read information from the database in a machine-readable form.

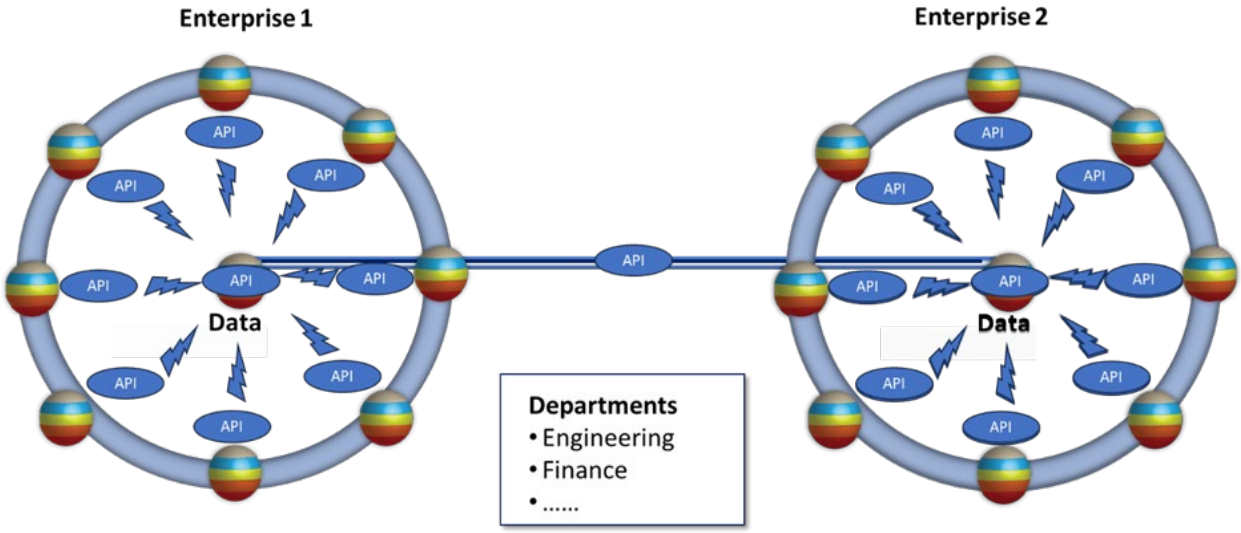


Figure C-3. Transition to End State

Figure C-4 illustrates an SoS with high connectivity. Multiple paths to necessary data are provided using APIs, which are the glue that makes complex SoS work. Multiple-path access for information sharing is a fundamental characteristic of a data-centric system. The CJADC2 program makes use of this concept to rapidly develop complex networks to deliver sensor-to-shooter solutions for real-time operations. z

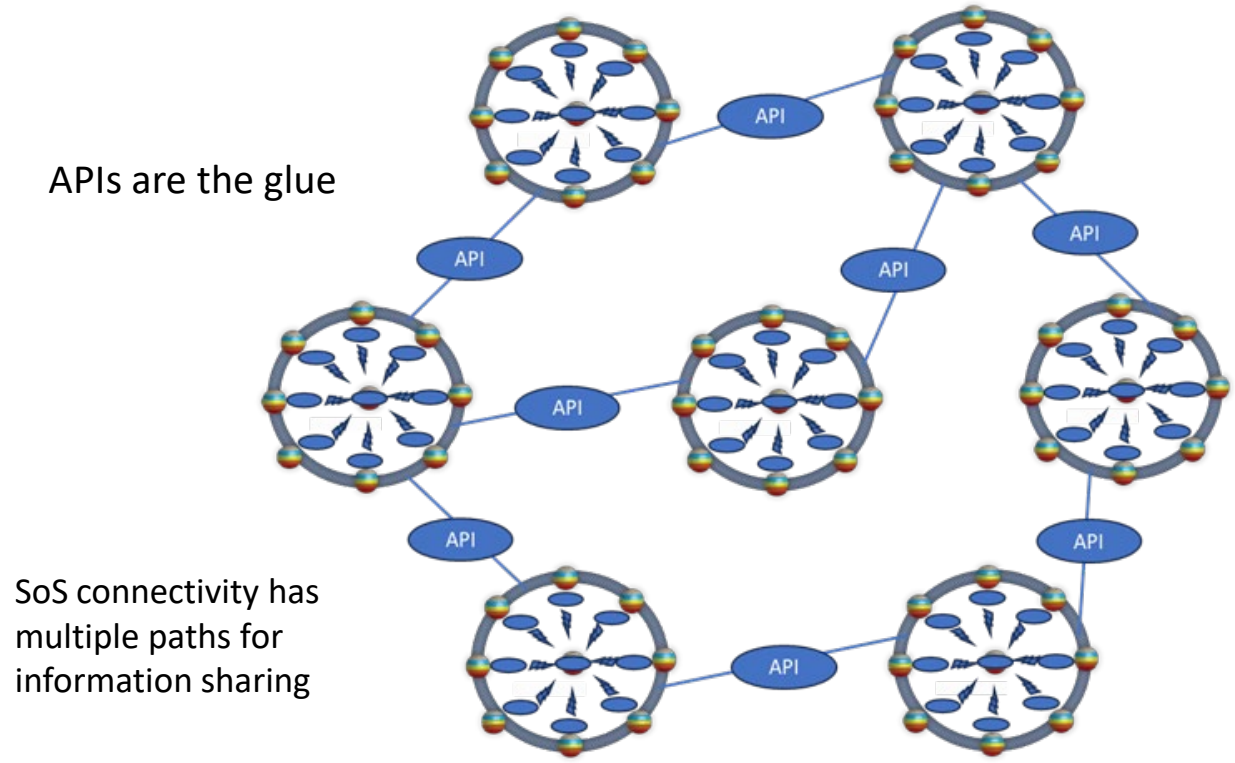


Figure C-4. Transition to Data-Centric SoS End State

C.3 Data Centricity Attributes

Following are 10 attributes relevant to data centricity:

1. Enhanced Data Accessibility and Availability

   A data-centric architecture ensures that data is easily accessible and available to authorized users across the organization. This accessibility fosters better decision making and enables real-time insights. Basic data-centric principles are at play here: You do not need the application that created the data to access or use the data. Metadata tagging is used to increase the visibility and understandability of data so that it can be found and also to grant access to data. Architectures have a data layer where point-of-presence security is performed for trust and security. While implementing data centricity, the use of APIs allows different applications to exchange interoperable data and information. APIs can be developed between organizations as well or at any level needed to link data.

2. Improved Data Quality and Governance

By focusing on data products and quality data as a strategic goal, organizations are motivated to maintain high data-quality standards. Implementing robust data governance practices becomes integral to ensure accuracy, security, reliability, and compliance with regulations.

3. Agility and Scalability

Data-centric architectures provide the flexibility to adapt to changing business needs. They allow for seamless scalability, enabling organizations to handle increasing volumes of data and accommodate evolving technologies.

4. Advanced Analytics and Business Intelligence

Leveraging a data-centric approach enhances an organization's ability to extract meaningful insights through advanced analytics and business intelligence tools. This leads to informed decision making and a competitive edge for the warfighter.

5. Cybersecurity and Data Protection

The concept of "vaulting" data, a term often associated with secure storage, is crucial in data centricity. A data-centric architecture emphasizes robust security measures, encryption, and access controls, safeguarding sensitive information from unauthorized access and cyber threats. This is done at the data point of presence and does not depend on system boundary protection methods. The DoW began implementation of the ZT method at three levels of implementation in 2025 with expected completion in 2027.

6. Cost Efficiency

Organizations can achieve cost savings by streamlining data processes and keeping data management efficient. Techniques include optimizing storage resources, reducing data redundancy, and minimizing the complexities associated with traditional, siloed IT architectures. Metadata tagging in particular makes data very efficient to find and provide access for use.

7. Customer-Centric Solutions

In the digital era, understanding customer behavior and preferences is vital. A data-centric architecture helps organizations create personalized and customer-centric solutions, thereby improving user satisfaction and loyalty. Customer applications can access interoperable data products without going through the application(s) that created the data. Data products evolve over time based on user feedback from consumers and changing requirements. Iterative updates made by the data product development teams ensure that data remains relevant and valuable to the needs of the consumer.

8. Integration of Emerging Technologies

The flexibility of a data-centric architecture facilitates the integration of emerging technologies such as AI, ML, and the Internet of Things. This integration drives innovation and keeps organizations at the forefront of technological advancements.

9. Collaboration and Knowledge Sharing

Using common services for data product orchestration encourages collaboration among different departments throughout the organization. This way, teams can share insights, collaborate on projects, and collectively contribute to the organization's goals.

10. Future-Proofing the Organization

In a rapidly evolving technological landscape, a data-centric IT architecture serves as a foundation for growth and innovation. It positions organizations to adapt to emerging trends and technologies to foster long-term relevance and competitiveness.

In summary, embracing a data-centric IT architecture and vaulting for secure data management offers a strategic advantage in the data century. The benefits range from improved decision making and analytics to heightened security and adaptability. Ultimately, it contributes to the overall success and sustainability of modern organizations.

AI depends on the availability of data to learn how to interact with operators to develop useful man-machine interfaces. A data-centric enterprise design is central to this ability to collect data from all sources with minimal dependance on human action. A well-developed data-centric SoS will reduce barriers to data collection while maintaining data security.

The introduction of APIs in conjunction with a data-centric environment is the perfect condition for AI operations. Machine-readable access to multiple needed databases provides the data-rich environment needed for AI enhanced operations. The AI system then can be introduced in the SoS to better enable operation of SoS. The AI can quickly provide commanders with options to engage the enemy with rapidly deployable SoS. Multiple SoS options and expected outcomes are important in rapidly evolving theaters of conflict.

C.4 AI Benefits and Best Practices in SoS Environment

Integrating AI into the SoS engineering process requires careful planning, design, and execution given the complexity and interdependencies of the systems involved. Following are 12 best practices for leveraging AI within the SoS engineering process to promote efficiency, scalability, and robustness:

1. Clear Understanding of SoS Engineering Objectives
    - AI can be used to meet the larger objectives of the entire SoS, not just the individual systems. AI should align with the overarching mission, ensuring that it contributes to the SoS desired behaviors across various individual systems.
    - AI can be used to define performance metrics in the SoS environment to monitor system performance, system optimization, and adaptive decision making.
2. Modeling and Simulation
    - AI algorithms, particularly ML and deep learning (DL) can be used to predict system behavior under varying conditions. In SoS engineering, the ability to simulate and predict system interactions is crucial, and AI models can help simulate complex behaviors that may otherwise be difficult to model using traditional methods.
    - AI can also be used to explore and simulate rare or extreme scenarios that might not be covered in standard system analysis, allowing for more robust system design.
3. Data Integration and Management
    - AI can process and analyze large data sets that emerge from multiple systems in the SoS engineering environment. AI tools can be used to aggregate data from sensors, system outputs, and operational feedback for improved decision making.
    - AI data fusion modeling tools can be used to integrate heterogeneous data from multiple subsystems, providing more accurate and holistic insight into the overall individual system and SoS health, performance, and status.

4. Autonomy and Decision Support
    - AI-based decision support systems can help human operators make informed choices in a complex SoS engineering environment. AI can also enable autonomous systems to make real-time decisions, increasing efficiency and reducing human workload.
    - Incorporating ML algorithms can allow the SoS to learn from experience and adapt autonomously to changes in the operating environment.
5. Interoperability and AI Standards
    - As SoS engineering often involves multiple independent systems, AI solutions need to be interoperable across different platforms and technologies. Using open standards and frameworks can facilitate seamless integration of AI components in SoS engineering.
    - Acknowledge emerging AI standards, guidelines, and best practices, especially for critical DoW systems. Recognize the ethical implications of AI decision making and ensure AI-driven decisions are transparent, auditable, and explainable.
6. AI Explainability and Transparency
    - In an SoS context, program personnel need to be able to explain the reasoning behind AI decisions. Implementing Explainable AI (XAI) methods and processes can help systems engineers and operators understand and trust AI outputs and interpret and adjust the AI system as needed.
    - Implementing a human-in-the-loop AI system is essential to allow human operators to oversee and interact with the decision-making process, mission objectives, and domain expertise.
7. Scalability and Flexibility
    - As SoS engineering environments often expand and evolve, SoS engineering should keep AI systems scalable and flexible enough to handle the increasing complexity and scale of the integrated systems.
    - An incremental and iterative approach for integrating AI into SoS engineering can begin with pilot projects and gradually scale AI capabilities to accommodate more complex scenarios and larger system integrations.
8. Safety and Security Considerations
    - Implementing AI techniques to detect and respond to cyber threats in SoS engineering can include anomaly detection, behavior analysis, and automated response mechanisms.

- AI systems should be fully tested for reliability, fail-safe operation, and safety under all operating conditions. This is especially critical when AI impacts critical infrastructure or safety-sensitive applications.

9. Continuous Learning and Evolution
    - In the SoS engineering process, implementing continuous learning and feedback loops, in which AI systems can be updated with new data, system performance metrics, and operator inputs, allows for keeping AI models up-to-date over time.
    - AI systems should facilitate knowledge transfer between subsystems, enabling lesson learned from one part of the system to inform others. This knowledge transfer can help improve individual system and SoS performance and reduce redundancies.
10. Collaboration Between Stakeholders and AI Engineers
    - Close collaboration among stakeholders (e.g., engineers, operators, system designers) and AI specialists is recommended so AI models will be designed with practical knowledge of the individual system and SoS needs, limitations, and objectives.
    - AI systems should be designed with the end users in mind. Considering human factors and usability aspects helps ensure the AI systems are effective in real-world scenarios.
11. Validation, Verification, and Testing
    - AI models need to undergo rigorous testing, validation, and verification processes. This includes testing under various operational conditions, edge cases, and failure modes. The Chief Digital and Artificial Intelligence Office (CDAO) has developed a set of T&E frameworks aimed at building trust in AI-enabled systems within the DoW. These frameworks cover various dimensions of AI evaluation, including model T&E, human systems integration T&E, systems integration T&E, and operational T&E. Each framework equips testers with important insights into AI system assessment. For detail information on T&E frameworks in AI-enabled systems is available from the blog "CDAO Test and Evaluation Strategy Frameworks."
    - Since SoS engineering environments are highly dynamic, AI models should be tested for robustness and adaptability to unforeseen circumstances, including rare failure events and system degradations.
    - See the DT&E of Artificial Intelligence-Enabled Systems Guidebook (2025) and the Cyber DT&E of Artificial Intelligence Enabled Systems online appendix for more information on testing of AI enabled systems.

12. AI Integration with SoS Life Cycle
    - Programs should consider AI integration throughout the entire individual system and SoS life cycle-from initial concept, through design and deployment, to operation and decommissioning. This practice ensures that AI elements are adaptable and can evolve as the individual system and SoS matures.
    - Maintaining detailed documentation of AI models, decisions, and system changes is critical for maintaining traceability and accountability throughout the individual system and SoS life cycle.

By implementing these best practices, programs can effectively integrate AI into the SoS engineering process, enhancing system performance, adaptability, and decision making while managing the inherent complexity of interconnected systems.

C.5 Zero Trust

The DoW ZT-based cybersecurity program is a broad-based effort to secure data at its point of presence. The effort includes a two-phased approach, Target and Advanced, as described in the DoD Zero Trust Strategy (DoD CIO 2022). The effort is evolving and contributes to ensuring the data used for AI is secure.

Implementing ZT security model in an SoS environment requires careful consideration because of the complexity, distributed nature, and interdependencies of the systems involved. ZT assumes that no system, user, or device (inside or outside the network) should be trusted by default. It emphasizes continuous verification, least privilege access, and micro-segmentation. Following are 10 best practices for applying ZT in an SoS environment:

1. Understand the SoS Architecture
    - A thorough understanding of the SoS architecture is important for the successful implementation of ZT. This includes understanding the relationships between systems, the routes of data transmission, and the communication dynamics among different components.
    - It is important to define the required data and information flows while recognizing the trusted boundaries that exist between systems, subsystems, and external entities.
    - Establishing these parameters facilitates a more secure and efficient integration of ZT principles within the SoS framework.

2. User and Device Identity Management
    - Implementing multi-factor authentication (MFA) for all users and devices is a best practice. Authentication should be robust, leveraging technologies including biometrics, hardware tokens, or certificates.
    - Devices connecting to the systems should be authenticated and their security posture verified (e.g., through endpoint detection and response solutions or mobile devices management).
    - Establishing strict access controls based on roles helps ensure users and devices access only the resources necessary for their tasks.
3. Least Privilege Access
    - Implementing least-privilege policies helps ensure users and systems are granted the minimum access necessary to perform their functions. This approach should be dynamic and based on real-time assessment.
    - Applying micro-segmentation principles by dividing the network into small, isolated segments or zones to limit lateral movement and reduce the blast radius in case of breach. Each segment should have its own access control policies.
    - Implementing just-in-time access controls, granting temporary privileges for a specific time when needed to further reduce exposure.
4. Continuous Monitoring and Analytics
    - Monitoring user and system behavior continuously, using ML advance analytics to detect anomalies. This is essential for identifying insider threats or compromised credentials.
    - Implementing telemetry to collect detail logs and metrics about all activities across the SoS. The data will provide insights into potential security risk and essential for incident report.
    - Integrating threat intelligence feeds to dynamically adjust security policies based on emerging threats.
5. Secure Communication Channels
    - Encrypting data both at rest and in transit by implementing strong encryption protocols to prevent unauthorized interception or tampering with sensitive data, especially in communication between systems and subsystems.
    - Apply secure APIs for system integration, ensuring authentication and authorization are enforced for all external calls, especially in complex SoS environments.

6. Automated Response and Remediation
    - Develop a robust incident response plan with clear steps for mitigating potential breaches. Use an automated response mechanism to quickly isolate affected systems and contains threats.
    - Leverage tools like Security Orchestration, Automation, and Response (SOAR) platforms to automate the detection and response processes, ensuring a fast reaction to security incidents.
7. Third-Party and Supply Chain Security
    - If third-party systems are part of the SoS construct, such systems should also be subject to the ZT principles. This includes strong identity management and strict monitoring of their access.
    - Secure the supply chain by verifying the integrity and security of software and hardware used across the systems and SoS, especially in a distributed environment with multiple providers.
8. Granular Policy Enforcement
    - Access decisions should be based on real-time risk assessments, considering the identity of the user/device, context (location, time), the resource being accessed, and the security posture of the requesting entity.
    - Implement context-aware authentication that can trigger additional layers of verification based on situational factors (e.g., accessing sensitive data outside of working hours or from new location).
9. Collaboration and Communication Security
    - Develop secure communication and collaboration platforms for internal and external communication. Apply encryption, identity verification, and data protection policies on these platforms.
    - Consider implementing Data Loss Prevention (DLP) mechanisms across all systems within the SoS to prevent accident or malicious data leakage.
10. Testing and Validation
    - Regularly conduct red team exercises to simulate attacks on the SoS to identify vulnerabilities in the ZT architecture and to improve detection and response capabilities.
    - Constantly reassess ZT policies and controls to ensure they are up to date with evolving threats and the changing needs of the SoS environment.

- See DoDI 5000.89, “Test and Evaluation,” DoW Manual 5000.103, “Cyber Developmental Test and Evaluation,” and the DoD Cyber DT&E Guidebook (2025) for more information on cyber test and evaluation.

By applying these best practices, organizations can build a resilient ZT architecture that ensures security and integrity across the complex and interconnected systems within an SoS environment.

DC, AI, and ZT combine to provide an SoS environment for rapid development and combined arms use for the warfighter. DC and ZT provide AI with the ability to operate effectively and in a way that can be trusted in modern warfare.

Glossary

Agile Approach: End user(s) team with developers in order to make instant decisions on user functionality. High-level requirements are initially prioritized and developed quickly by small teams in order to provide a working product to the customer. Multiple rapidly executed increments are developed, and capabilities are released to the customer as soon as possible. Prototypes may be used as a starting place and use a modular, open-systems approach. Agile methods are typically used for small, low risk projects. (Warfighting Acquisition University (WarU) Glossary)

Architecture: The structure of components, their relationships, and the principles and guidelines governing their design evolution over time. (WarU Glossary). The architecture of an SoS is a persistent technical framework for governing the evolution of an SoS over time.

Capability: The ability to complete a task or execute a course of action under specified conditions and level of performance. (WarU Glossary)

Constituent System: Independent system that forms part of a system of systems (SoS). Constituent systems can be part of one or more SoS. Each constituent system is a useful system by itself, having its own development, management, use, goals, and resources, but interacts within the SoS to provide the unique capability of the SoS. (ISO/IEC/IEEE 21841, 2019)

Data Centricity: An environment where data is the primary and permanent asset separated from systems/applications making data available to a broad range of tools and analytics within and across security domains for enrichment and discovery. (DoD Data Strategy, 2020)

Data Loss Prevention: A security strategy focused on detecting and preventing data leakage or loss through unauthorized use, exfiltration, or destruction. DLP tools deployed only at a system boundary are inadequate to address corruption of data throughout the system. Therefore, DLP tools are placed at identified enforcement points throughout the architecture to detect and mitigate data breaches and exfiltration. (NSA CIS, March 2024)

Design: The characteristics of an entity (system, component, SoS, etc.) in sufficient detail to enable the entity's implementation. Typical characteristics include components, control logic, data structures, input/output formats, interface descriptions, and algorithms. (IEEE Std 610.12) The design of an SoS consists of the architecture of the SoS together with changes to the designs of the constituent systems that enable them to work together according to the architecture.

Development, Security, Operations (DevSecOps): A combination of software engineering methodologies, practices, and tools that unifies software development (Dev), security (Sec), and operations (Ops). It emphasizes collaboration across these disciplines, along with automation and

continuous monitoring to support the delivery of secure, high-quality software. DevSecOps integrates security tools and practices into the development pipeline, emphasizes the automation of processes, and fosters a culture of shared responsibility for performance, security, and operational integrity throughout the entire software life cycle, from development to deployment and beyond. (DoD DevSecOps Fundamentals 2.5, 2024)

Element: An "element" in systems engineering is defined as a component of a larger structure. A component of a large structure can be hardware, software, people, facilities, data, processes, and services. These individual elements, when combined, create a system that delivers a result not achievable by individual element alone. In the context of SoS engineering, "core element" is defined as engineering activities that are essential for managing the integration and life cycle of an SoS.

Emerging: Principle that entities exhibit properties which are meaningful only when attributed to the whole, not to its parts. These properties cannot be reduced or decomposed back down to the those of any individual constituent system. (Checkland, 1999)

Family of Systems (FoS): A set of systems that provide similar capabilities through different approaches to achieve similar or complementary effects (WarU Glossary). For instance, the war fighter may need the capability to track moving targets. The FoS that provides this capability could include unmanned or manned aerial vehicles with appropriate sensors, a space-based sensor platform, or a special operations capability. Each can provide the ability to track moving targets but with differing characteristics of persistence, accuracy, timeliness, etc." This definition is included for completeness. FoS are fundamentally different from SoS because a family of systems lacks the synergy of a system of systems. The family of systems does not acquire qualitatively new properties as a result of the grouping. In fact, the member systems may not be connected into a whole. This guidebook specifically addresses SoS, but some of its contents may apply to FoS.

Governance: Process of establishing and enforcing strategic goals and objectives, organizational policies, and performance parameters. (ISO/IEC/IEEE 21840, 2019)

Management: System of controls and processes required to achieve the strategic objectives set by the organization's governing body. (ISO/IEC/IEEE 21840, 2019)

Micro-segmentation: Network security technique that involves dividing a data center or cloud environment into very small, isolated security segments, allowing administrators to define granular security policies for each individual workload or application, effectively limiting lateral movement of threats and minimizing the impact of a breach by isolating compromised segments. (NSA CIS, March 2024)

Modular Open Systems Approach (MOSA): An acquisition and design strategy consisting of a technical architecture that uses open system interfaces. The strategy supports a modular, loosely coupled and highly cohesive system structure that allows severable system components at the appropriate level to be incrementally added, removed, or replaced throughout the life cycle of a system platform to afford opportunities for enhanced competition and innovation.

Net Centricity: The ability to provide a framework for full human and technical interoperability that (1) allows all DoW users and mission partners to share needed information in a timely manner and in an understandable and actionable form, and (2) protects information from those who should not have access to it. (DoD Data Strategy, 2020; DoD Net-Centric Data Strategy, 2003)

System: Combination of interacting elements organized to achieve one or more stated purposes. (ISO/IEC/IEEE 21840, 2019)

System of Systems (SoS): A set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities (WarU). Both individual systems and SoS conform to the accepted definition of a system in that each consists of parts, relationships, and a whole that is greater than the sum of the parts; however, although an SoS is a system, not all systems are SoS.

SoS Engineering: Deals with planning, analyzing, organizing, and integrating the capabilities of a mix of existing and new systems into an SoS capability greater than the sum of the capabilities of the constituent parts. Consistent with the DoW vision and enabling net-centric operations, SoS may deliver capabilities by combining multiple collaboratives and autonomous-yet-interacting systems. The mix of systems may include existing, partially developed, and yet-to-be- designed independent systems. (WarU)

System of Interest (SoI): A system whose life cycle is under consideration.

Technical Debt: An element of design or implementation that is expedient in the short term, but that would result in a technical context that can make a future change costlier or impossible. Technical debt may result from having code issues related to architecture, structure, duplication, test coverage, comments and documentation, potential bugs, complexity, coding practices, and style which may accrue at the level of overall system design or system architecture, even in systems with great code quality. (DoD SWE DT&E in DevSecOps Guidebook, 2025**)**

Waterfall Approach: Development activities are performed in order, with possibly minor overlap, but with little or no iteration between activities. User needs are determined, requirements are defined, and the full system is designed, built, and tested for ultimate delivery at one point in time. A document-driven approach best suited for highly precedented systems with stable requirements. (WarU Glossary)

Zero Trust (ZT): A more robust cybersecurity model that eliminates the idea of trusted or untrusted networks, devices, personas, or processes, and shifts to multi-attribute-based confidence levels that enable authentication and authorization policies based on the concept of least privileged access. (DoD ZT Strategy, 2022)

Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
CCB	Configuration Control Board
CIS	Cybersecurity Information Sheet
CJADC2	Combined Joint All Domain Command and Control
CM	Configuration Management
CONOPS	Concept of Operations
COTS	Commercial Off-the-Shelf
CS	Constituent System
CTC	Capability Trade Council
DC	Data Centric
DE	Digital Engineering
DevSecOps	Development, Security, and Operations
DLP	Data Loss Prevention
DMR	DoD Metadata Registry
DoD	Department of Defense
DoW	Department of War
DSP	Defense Standardization Program
DT&E	Developmental Test and Evaluation
EEMP	Enterprise Engineering Management Plan
FoS	Family of Systems
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
IMS	Integrated Master Schedule
ISO	International Standardization Organization
IT	Information Technology

IWG	Interface Working Group
KPP	Key Performance Parameter
MBCRA	Mission-Based Cyber Risk Assessment
MBSE	Model Based Systems Engineering
MET	Mission Engineering Thread
ML	Machine Learning
MOA	Memorandum of Agreement
MOSA	Modular Open Systems Approach
MOU	Memorandum of Understanding
M&S	Modeling and Simulation
MT	Mission Thread
NSA	National Security Agency
OMG	Object Management Group
OOSEM	Object-Oriented Systems Engineering Method
OT&E	Operational Test and Evaluation
OUSW(R&E)	Office of the Under Secretary of War for Research and Engineering
PM	Program Manager
RIO	Risk, Issue, and Opportunity
RMB	Risk Management Board
RTM	Requirements Traceability Matrix
SDLC	Software Development Life Cycle
SE	Systems Engineering
SEP	Systems Engineering Plan
SLA	Service Level Agreement
SOA	Service-Oriented Architecture
SME	Subject Matter Expert
SoI	System of Interest
SoS	System (or Systems) of Systems
SoS Engineering	System of Systems Engineering

SWE	Software Engineering
SysML	Systems Modeling Language
T&E	Test and Evaluation
UAF	Unified Architecture Framework
VAULTIS	visible, accessible, understandable, linked, trustworthy, interoperable, and secure
VECP	Value Engineering Change Proposal
VEP	Value Engineering Proposal
WarU	Warfighting Acquisition University
WAS	Warfighting Acquisition System
WWW	World Wide Web
ZT	Zero Trust

References

ASSIST. Defense Logistics Agency (DLA). https://assist.dla.mil

Boardman, John, and Brian Sauser. *Systems Thinking: Coping with 21st Century Problems.* Boca Raton, Florida: CRC Press, 2008.

Boxer, P., et al. "Systems-of Systems Engineering and the Pragmatics of Demand." IEEE Systems Conference, Montreal, Canada, April 7-10, 2008.

Brownsword, L., et al. "System-of-Systems Navigator: An Approach for Managing System-of-Systems Interoperability." Software Engineering Institute, Technical Note-019, April 2006.

Checkland P. "Systems Thinking, Systems Practice." New York: John Wiley & Sons, 1999.

Chief Digital and Artificial Intelligence Office (CDAO). "Testing & Evaluation Framework." CDAO Test and Evaluation Strategy Frameworks (blog). DoW CDAO, May 20, 2024. https://www.ai.mil/Latest/Blog/Article-Display/Article/3940283/cdao-test-and-evaluation-strategy-frameworks/

Conrow, E.H. "Risk Management for System of Systems." Risk-Services.Com, February 2005.

Dahmann, Judith, et al. "An Implementers' View of Systems Engineering for Systems of Systems." Conference paper. 2011 IEEE International Systems Conference, Montreal, QC, Canada, 4-7 April 2011 https://doi.org/10.1109/SYSCON.2011.5929039

DoD Application Programming Interface (API) Technical Guidance, Minimum Viable Capability Release (MVCR) 3, March 2026. https://www.cto.mil/wp-content/uploads/2025/05/API-Tech-Guidance-MVCR-2-2025_0516-Cleared.pdf

DoD Cyber Developmental Test and Evaluation Guidebook. Office of the Undersecretary of Defense for Research and Engineering. Version 3.0, June 2025. https://www.cto.mil/wp-content/uploads/2025/08/Cyber-DTE-Guidebook-V3-June2025_Final_87e72e.pdf

DoD Cyber DT&E of Artificial Intelligence Enabled Systems. Office of the Undersecretary of Defense for Research and Engineering. [Accessed January 2026] https://www.dau.edu/guidebook/cyber-dte-ai-enabled-systems

DoD Data, Analytics, and Artificial Intelligence (AI) Adoption Strategy (DoD DAAIAS)." Deputy Secretary of Defense, June 27, 2023. https://media.defense.gov/2023/Nov/02/2003333300/-1/-1/1/DOD_DATA_ANALYTICS_AI_ADOPTION_STRATEGY.PDF

DoD Data Strategy, September 30, 2020. https://media.defense.gov/2020/Oct/08/2002514180/-1/-1/0/DOD-DATA-STRATEGY.PDF

DoD DevSecOps Fundamentals 2.5, October 2024. https://dodcio.defense.gov/Portals/0/Documents/Library/DoD%20Enterprise%20DevSecOps%20Fundamentals%20v2.5.pdf

DoD Directive 7045.20, "Capability Portfolio Management." Office of the Under Secretary of Defense for Acquisition and Sustainment, September 25, 2023. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodd/704520p.pdf?ver=Y_7TIls2RHyUIYVGHOH6iA%3D%3D

DoD Instruction 4245.14, "DoD Value Engineering Program." Under Secretary of Defense for Research and Engineering, October 1, 2024.

DoD Instruction 5000.02, "Operation of the Adaptive Acquisition Framework," Change 1. Office of the Under Secretary of Defense for Acquisition and Sustainment, June 8, 2022. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500002p.PDF

DoD Instruction 5000.61, "DoD Modeling and Simulation Verification, Validation, and Accreditation." Office of the Undersecretary of Defense for Research and Engineering, September 17, 2024.

DoD Instruction 5000.88, "Engineering of Defense Systems." Office of the Undersecretary of Defense for Research and Engineering," November 18, 2020. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500088p.PDF;

DoD Instruction 5000.89, "Test and Evaluation." Office of the Undersecretary of Defense for Research and Engineering, November 19, 2020.

DoD Instruction 5000.97, "Digital Engineering." Office of the Undersecretary of Defense for Research and Engineering, December 21, 2023.

DoD Instruction 5000.98, "Operational Test and Evaluation and Live Fire Test and Evaluation." Director, Operational Test and Evaluation, December 9, 2024.

DoD Instruction 8410.02, "Support to DoD Information Network Operations." Office of the DoD Chief Information Officer, December 8, 2021. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/841002p.pdf

DoD Manual 4120.24, "Defense Standardization Program (DSP) Procedures." Under Secretary of Defense for Research and Engineering, September 24, 2014, Incorporating Change 2, Effective October 15, 2018.

DoW Manual 5000.103, "Cyber Developmental Test and Evaluation." Office of the Under Secretary of Defense for Research and Engineering, February 25, 2026.

DoD Manual 5000.101, "Operational Test and Evaluation and Live Fire Test and Evaluation of Artificial Intelligence-Enabled and Autonomous Systems." Director, Operational Test and Evaluation, December 9, 2024.

DoD Mission Engineering Guide, Version 2.0, October 1, 2023. https://ac.cto.mil/wp-content/uploads/2023/11/MEG_2_Oct2023.pdf

DoD Net-Centric (NC) Data Strategy. DoD Chief Information Officer, May 9, 2003. https://dodcio.defense.gov/Portals/0/documents/Net-Centric-Data-Strategy-2003-05-092.pdf

DoD Risk, Issue, and Opportunity (RIO) Management Guide for Defense Acquisition Programs. Office of the Under Secretary of Defense for Research and Engineering, December 2023. https://www.cto.mil/sea/pg/

DoD Systems Engineering Plan (SEP) Outline, Version 4.1. Office of the Under Secretary of Defense for Research and Engineering, May 2023. https://ac.cto.mil/wp-content/uploads/2023/05/SEP-Outline-4.1.pdf

DoD Zero Trust (ZT) Strategy. DoD Chief Information Officer, October 21, 2022. https://dodcio.defense.gov/Portals/0/Documents/Library/DoD-ZTStrategy.pdf

DT&E of Artificial Intelligence-Enabled Systems Guidebook. Office of the Under Secretary of Defense for Research and Engineering, February 2025. https://www.cto.mil/wp-content/uploads/2025/08/TE_of_AIES_Guidebook-Feb-2025-1.pdf

Gunning, D., et al. "XAI—Explainable Artificial Intelligence." *Science Robotics* 4(37), eaay7120, 2019. https://doi.org/10.1126/scirobotics.aay7120

Integrated Master Plan and Integrated Master Schedule Preparation and User Guide. Office of the Under Secretary of Defense for Research and Engineering, May 2023. https://www.cto.mil/wp-content/uploads/2024/10/IMP-IMS-Guide-May2023-C1-Oct2024.pdf

Internet Engineering Task Force. https://www.ietf.org/

ISO/IEC/IEEE 21840, "Guidelines for the utilizations of ISO/IEC/IEEE 15288 in the context of SoS," 2019.

ISO/IEC/IEEE 21841, "Taxonomy of SoS," 2019.

Maier, M. W. "Architecting Principles for System of Systems." *Systems Engineering.* John Wiley & Sons, Vol. 1(4), pages 267-284, 1998.

Mission Architecture Style Guide, Version 1.0, January 6, 2025. https://ac.cto.mil/wp-content/uploads/2025/01/U-Mission-Architecture-Style-Guide-Final_07Jan2025.pdf

Mission Engineering. Web page. Office of the Under Secretary of War for Research and Engineering, Mission Engineering. [Accessed December 11, 2025]. https://ac.cto.mil/mission-engineering/

Modular Open Systems Approach (MOSA) Reference Frameworks in Defense Acquisition Program, May 2020. https://www.cto.mil/wp-content/uploads/2023/06/MOSA-Framework-2020.pdf

Modular Open Systems Approach (MOSA). Web page. Office of the Under Secretary of War for Research and Engineering, Systems Engineering and Architecture. [Accessed October 24, 2025]. https://www.cto.mil/sea/mosa/

National Security Agency (NSA) Cybersecurity Information Sheet (CIS), Advancing Zero Trust Maturity Throughout the Network and Environment Pillar v1.0, March 2024. https://media.defense.gov/2024/Mar/05/2003405462/-1/-1/0/CSI-ZERO-TRUST-NETWORK-ENVIRONMENT-PILLAR.PDF

Office of the Secretary of War, “Reforming the Joint Requirements Process to Accelerate Fielding of Warfighting Capabilities,” November 7, 2025.

Office of the Secretary of War, “Transforming the Defense Acquisition System into the Warfighting Acquisition System (WAS) to Accelerate Fielding of Urgently Needed Capabilities to Our Warriors,” November 7, 2025.

SD-24, “Value Engineering: A Guidebook of Best Practices and Tools.” February 2025.

Software Developmental Test and Evaluation in DevSecOps Guidebook, Office of the Under Secretary of Defense for Research and Engineering, January 27, 2025. https://www.cto.mil/wp-content/uploads/2025/01/Software_DTE_DEVSECOPS_GB_Jan2025_Signed.pdf

Software Engineering for Continuous Delivery of Warfighting Capability (SWE Guide). Office of the Under Secretary of Defense for Research and Engineering, 2025. https://www.cto.mil/wp-content/uploads/2025/08/SWE-Guide-July2025-secured-1.pdf

Systems Engineering and Architecture (SE&A). Website. https://www.cto.mil/sea/

Systems Engineering and Architecture (SE&A) Library. Web page. Office of the Under Secretary of War for Research and Engineering, Systems Engineering and Architecture. [Accessed October 24, 2025]. https://www.cto.mil/sea/pg/

Systems Engineering Guide for Systems of Systems (SE Guide for SoS). Director, Systems and Software Engineering, Deputy Under Secretary of Defense for Acquisition and Technology, Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, 2008.

Systems Engineering Guidebook. Office of the Under Secretary of Defense for Research and Engineering, February 2022. https://www.cto.mil/sea/pg/

Under Secretary of War for Acquisition and Sustainment (USW(A&S)) Memorandum, “Establishment of Department-Wide Modular Open Systems Approach Policy and Minimum Expectations,” September 21, 2026.

Warfighting Acquisition University (WarU) Glossary. https://www.waru.edu/glossary

System of Systems Engineering Guidebook

September 2026

Office of the Under Secretary of War for Research and Engineering
Office of the Assistant Secretary of War for Mission Capabilities
Office of the Deputy Assistant Secretary of War for Multi-Domain Joint Operations
Systems Engineering and Architecture | Developmental Test, Evaluation, and Assessments
3030 Defense Pentagon
Washington, DC 20301
osd-sea@mil.mil
https://www.cto.mil/sea/

Distribution Statement A. Approved for public release. Distribution is unlimited.
DOPSR Case # 26-T-2752.