INTERNATIONAL JOURNAL OF MANAGEMENT AND ORGANIZATIONAL RESEARCH

System-of-Systems Engineering Management: A Review of Modern History and a Path Forward

Ahmed Abdullah Baeroom [1*], Hayder M Mansoor [2], Laila Alghallabi [3], Mohammed El fatih MEKNACI [4]
[1] College of Civil and Transportation Engineering, Hohai University, China
[2] Department of Civil Engineering, Azad Islamic University, Qazvin, Iran
3 School of International Development and Cooperation (SIDC), University of International Business and Economics, Beijing, China
[4] School of Architecture, Southeast University, Nanjing, China

* Corresponding Author: **Ahmed Abdullah Baeroom**

Article Info

ISSN (online): 2583-6641
Impact Factor (RSIF): 8.56
Volume: 05
Issue: 03
May-June 2026
Received: 17-03-2026
Accepted: 16-04-2026
Published: 15-05-2026
Page No: 78-86

Abstract
This review article confronts the escalating complexity inherent in engineering and managing Systems of Systems (SoS), where multiple independent and complex systems coalesce to create a new, more capable whole. The authors argue that while systems engineering has developed architectures for uncertainty, the specialized domain of SoS Engineering (SoSE) still lacks a coherent management framework to keep pace with relentless technological change. Through a comprehensive analysis of SoS literature, the paper establishes a foundational theory: that SoS can be effectively defined by their distinguishing characteristics and, crucially, can be conceptualized as a network of interrelated components. This network perspective allows for the adaptation of proven "best practices" from network management. Building on this premise, the authors construct a novel SoSE management framework by tailoring the established FCAPS model—which encompasses Fault, Configuration, Accounting, Performance, and Security management—to the unique demands of SoS. The practical application and validity of this proposed framework are then rigorously demonstrated through an in-depth case analysis of a well-documented, real-world SoS: the Integrated Deepwater System, illustrating its utility in guiding better understanding, engineering, and management within the SoSE domain.

DOI: https://doi.org/10.54660/IJMOR.2026.5.3.78-86

Keywords: System-of-Systems (SoS), SoS Engineering (SoSE), Management Framework, FCAPS Model, Network Management, Integrated Deepwater System

1. Introduction

The technological landscape of the twenty-first century is characterized by an unprecedented degree of interconnectedness. The most critical capabilities upon which modern society depends—from national security and disaster response to global commerce and urban infrastructure—are no longer delivered by singular, monolithic systems. Instead, they emerge from the collaborative operation of multiple, independently functioning systems that have been integrated to achieve outcomes unattainable by any constituent part acting alone. These complex aggregations are formally recognized as Systems of Systems (SoS).

Consider the architecture of a modern ballistic missile defense system. It comprises early-warning satellites, ground-based radars, command and control nodes, and interceptor missiles, each a complex system in its own right, developed and managed by different organizations, yet all must function as a cohesive whole to detect, track, and neutralize a threat. Similarly, a "smart city" initiative integrates traffic management systems, power grids, public safety networks, and environmental sensors, all owned and operated by distinct entities, to optimize urban living. These examples illustrate the defining paradox of the SoS: it is a collection of systems that are simultaneously independent and interdependent.

Traditional systems engineering (TSE) has provided a robust and mature toolkit for the development of complex, standalone systems. Its methodologies emphasize top-down requirements analysis, hierarchical decomposition, and rigorous lifecycle management, all applied to a system with a clear boundary, a single acquiring authority, and a defined purpose [1]. However, when applied to an SoS, these traditional approaches encounter fundamental limitations. The constituent systems within an SoS retain their own operational and managerial independence, they evolve according to their own timelines and priorities, and their interactions give rise to emergent behaviors that cannot be predicted from the properties of the individual systems alone [2]. TSE, designed for closed systems, struggles to manage this open, dynamic, and politically charged environment.

This struggle has given rise to the specialized field of SoS Engineering (SoSE). Over the past two decades, significant intellectual effort has been dedicated to understanding the unique nature of SoS and developing architectural frameworks to design for uncertainty and interoperability. Frameworks such as the Department of Defense Architecture Framework (DoDAF) and the Unified Architecture Framework (UAF), along with modeling languages like SysML, have advanced the community's ability to describe and specify complex SoS architectures [3]. This body of work represents the "architectural turn" in SoSE, a critical step in learning how to *design* these intricate wholes.

Yet, a critical gap remains. While the field has made strides in the *architecture* of SoS, it has not developed a parallel, coherent framework for the *management* of SoS throughout its entire lifecycle. Management, in this context, refers to the continuous, dynamic process of directing, monitoring, and optimizing the SoS to ensure it delivers its intended capabilities, adapts to changing environments, and manages the relationships between its constituent parts. Current literature often eloquently describes the myriad management challenges inherent in SoS—configuration control across independent systems, performance measurement at the SoS level, and the allocation of shared resources—but rarely offers a prescriptive, integrated, and adaptable management model to address them [4]. This "management gap" is the central problem this paper seeks to address.

This paper posits that a path forward lies in a fundamental reconceptualization of the SoS. We argue that to manage an SoS effectively, it must be viewed not merely as a "system of systems," but fundamentally as a **network**. This perspective shifts the focus from the internal complexities of each constituent system to the structure, dynamics, and health of the interconnections between them. By abstracting each constituent system into a node and each interface or information exchange into a link, we can leverage a rich body of proven management theory and practice from a mature and highly related field: telecommunications network management.

The core contribution of this paper is the novel adaptation of the FCAPS model—the International Organization for Standardization's (ISO) foundational framework for network management—to the unique demands of SoSE. FCAPS, which stands for Fault, Configuration, Accounting, Performance, and Security management, provides a comprehensive and time-tested taxonomy of management functions [5]. By tailoring each of these five domains to the SoS context, we construct a holistic management framework that addresses the full spectrum of SoS operational and evolutionary challenges.

The validity and utility of this proposed framework are then demonstrated through an in-depth case study of the U.S. Coast Guard's Integrated Deepwater System program. Deepwater, one of the most ambitious and well-documented SoS acquisitions in history, serves as a cautionary tale of management failure. By applying our SoSE-FCAPS lens to the Deepwater program, we illustrate how its myriad problems—from performance shortfalls and integration failures to cost overruns and safety issues—can be systematically diagnosed and understood as failures across the five core management domains. This analysis confirms the framework's diagnostic power and its potential as a tool for guiding future SoS endeavors.

The paper proceeds as follows. Section 2 provides a modern history of SoSE, tracing its evolution and focusing on the persistent challenge of management. Section 3 establishes our foundational theory, defining the key characteristics of an SoS and justifying the network perspective. Section 4 details our proposed SoSE management framework, mapping each FCAPS domain to the SoS context. Section 5 applies this framework to the Integrated Deepwater System case study. Section 6 discusses the implications of our findings for researchers and practitioners and suggests directions for future research, followed by a brief conclusion in Section 7.

2. A Modern History of SoSE: The Quest for a Management Framework

The formal study of Systems of Systems is a relatively recent phenomenon, emerging from the recognition in the late 20th century that a new class of engineering challenges was not being adequately addressed by existing methodologies. This section traces the intellectual history of the field, charting its evolution from initial attempts at definition, through a period of intense focus on architecture, to the present-day recognition of a critical gap in management practice.

2.1. The Foundational Era: Defining the Problem Space (1990s-2000s)

The intellectual groundwork for SoSE was laid in the 1990s, primarily by researchers grappling with the increasing complexity of defense and aerospace systems. The seminal work of Mark W. Maier provided the first enduring and widely accepted definition of an SoS. Maier argued that the defining characteristic of an SoS is not its size, but the operational and managerial independence of its components [6]. He posited that for a system to be considered an SoS, its constituent systems must be able to operate effectively on their own and, crucially, must continue to be managed independently to fulfill their own purposes, even after being integrated into the larger whole.

Building on Maier's foundational insights, Sage and Cigliano synthesized the literature to articulate a set of five distinguishing characteristics that have become the standard litmus test for identifying an SoS [2]. These are:

1. **Operational Independence:** Constituent systems can operate independently and are useful in their own right.
2. **Managerial Independence:** Constituent systems are independently acquired and managed, often by different stakeholders with different priorities.
3. **Geographical Distribution:** Constituent systems are often physically distributed, interacting primarily through information exchange.
4. **Evolutionary Development:** The SoS is never fully

complete; it evolves over time in response to changing needs, technological advancements, and the independent evolution of its constituent systems.

5. **Emergent Behavior:** The SoS as a whole exhibits behaviors and capabilities that are not present in any individual constituent system and are often unpredictable from them.

This era was crucial for *defining the problem*. It established a shared vocabulary and a conceptual framework for understanding why SoS were fundamentally different from large-scale but monolithic systems. The focus was on description and classification, setting the stage for the development of engineering approaches tailored to this new domain.

2.2. The Architectural Turn: Designing for Uncertainty (2000s-2010s)

With a clearer understanding of the nature of SoS, the research community and practitioner base turned their attention to the question of *how* to engineer them. The dominant theme of this period was architecture. Recognizing that top-down, requirements-driven design was ill-suited for the dynamic and decentralized nature of SoS, researchers and practitioners focused on developing frameworks and languages to describe, model, and analyze SoS architectures. This led to the proliferation of architecture frameworks, many driven by government and defense organizations. The Department of Defense Architecture Framework (DoDAF) and its UK counterpart, the Ministry of Defence Architecture Framework (MoDAF), provided standardized viewpoints for describing the operational, systems, and technical aspects of a defense enterprise [3]. These frameworks were later harmonized into the Unified Architecture Framework (UAF), which aims to provide a more generic and comprehensive modeling approach for complex systems [7].

Concurrently, the Systems Modeling Language (SysML) emerged as a powerful graphical notation for specifying, analyzing, and verifying complex systems, including SoS. SysML provided a means to model structure, behavior, requirements, and parametrics, offering engineers a common language to capture architectural designs [8].

The intellectual contribution of this "architectural turn" cannot be overstated. It moved the field from abstract description to tangible engineering practice. It provided tools to model interfaces, analyze information flows, and explore architectural alternatives before committing significant resources. The focus was on *designing for uncertainty*—creating flexible, adaptable architectures that could accommodate the independent evolution of constituent systems and the emergence of new capabilities. The work of researchers like Dr. Judith Dahmann, particularly her work on the "SoS Navigator," further bridged the gap between architecture and the dynamic processes of SoS development and evolution [9].

2.3. The Persistent Gap: The Management Challenge (2010s-Present)

Despite the significant progress in architectural modeling, a persistent and critical gap remains. Architecture frameworks excel at *description* and *design*, but they are not, in themselves, *management frameworks*. They do not prescribe how to continuously monitor, direct, and optimize an SoS once it is operational, nor do they provide a structured approach to the ongoing challenges of coordinating independent stakeholders, allocating shared resources, or adapting to the unplanned evolution of constituent systems.

A review of the literature from the past decade reveals a recurring theme: the identification of management challenges without the prescription of integrated management solutions. For instance, configuration management in an SoS is frequently cited as a significant problem. When constituent systems are managed independently, changes made by one organization to serve its own purposes can have unforeseen and detrimental effects on the capabilities of the larger SoS [10]. Similarly, performance management at the SoS level is fraught with difficulty. Traditional performance metrics are often defined at the level of the constituent system, and developing and measuring Key Performance Parameters (KPPs) for emergent, SoS-level behaviors remains a major challenge [11].

Other persistent management challenges include:

- **Fault Management:** Diagnosing the root cause of a failure in an SoS is complex because a problem may originate in one constituent system but manifest as a capability loss in another, or emerge from a failure in the interface between them.
- **Accounting/Resource Management:** Constituent systems often share common resources, such as communication bandwidth, data repositories, or even physical assets. Managing access, prioritizing usage, and resolving contention in the absence of a central authority is a significant management function.
- **Security Management:** Ensuring the security of an SoS is exponentially more complex than securing a single system. The integration of multiple systems, each with its own security posture and policies, creates a vastly expanded attack surface and introduces complex issues of trust and interoperability [12].

The literature is rich with case studies and conceptual papers that highlight these individual challenges. For example, studies of the Federal Aviation Administration's NextGen air transportation system illustrate the immense difficulty of managing the evolution of a safety-critical SoS with hundreds of independent stakeholders [13]. Analyses of multinational military coalitions repeatedly underscore the challenges of security and information sharing across systems with different classification levels and national ownership [14].

What is missing, however, is a holistic, integrated framework that brings these disparate management functions together. Practitioners are left with a laundry list of challenges but no coherent model for how to establish a management function that addresses them all in a coordinated manner. This is the void that this paper seeks to fill. The field has advanced from defining the beast, to designing its skeleton, and now needs a robust set of operational and management disciplines—a "physiology"—to manage it effectively throughout its life. This requires looking beyond the boundaries of traditional systems engineering for inspiration.

3. Deconstructing the Beast: A Foundational Theory of SoS

Before constructing a management framework, it is essential to establish a clear and operationalizable understanding of the object to be managed. This section synthesizes the defining characteristics of an SoS and, based on this synthesis,

introduces a powerful new lens through which to view it: the network perspective.

3.1. Defining Characteristics: More Than Just "Big" Systems

Building on the foundational work of Maier, Sage, and Cigliano, an SoS can be robustly defined by the following five interrelated characteristics. These are not merely taxonomic labels; they are the fundamental properties that drive the need for a specialized management approach.

- **Operational Independence:** Each constituent system within an SoS is a useful entity in its own right. A cutter in the Deepwater program, for example, can perform patrol and search-and-rescue missions even when not connected to the broader Deepwater command and control network. This means the SoS manager cannot assume total control over the system's internal operations.
- **Managerial Independence:** The constituent systems are acquired, owned, and managed by different organizations, each with its own goals, budgets, and timelines. In Deepwater, the cutters, aircraft, and C4ISR systems were developed by different prime contractors (Lockheed Martin, Northrop Grumman) and managed by different branches within the Coast Guard. The SoS manager must operate through influence and negotiation, not command and control.
- **Geographical Distribution:** The components of an SoS are often widely dispersed geographically, communicating primarily through information networks. This distribution makes centralized, physical oversight impossible and places a premium on the management of information exchanges and network links.
- **Evolutionary Development:** An SoS is never a finished product. It evolves continuously as new constituent systems are added, old ones are retired, and existing ones are upgraded to meet their own stakeholders' needs. This evolution is not centrally planned but is an emergent property of the entire SoS ecosystem. The management framework must therefore be dynamic and adaptive, not static.
- **Emergent Behavior:** The most important characteristic of an SoS is that its overall capability is greater than, and different from, the sum of its parts. The value of the Deepwater program was not in having a cutter, a helicopter, and a radio; it was in their ability to work together to perform missions like drug interdiction more effectively. This emergent behavior is the primary reason for creating the SoS, but it is also the most difficult property to manage, as it cannot be traced back to a single component.

These five characteristics collectively paint a picture of an entity that is distributed, dynamic, politically complex, and whose most important properties are not located in any single place. To manage such an entity, we need an abstraction that captures these essential features.

3.2. The Network Perspective: Seeing the Forest for the Trees

We propose that the most powerful and practical abstraction for managing an SoS is to view it as a **network**. This perspective is not merely a metaphor; it is a formal reframing that aligns perfectly with the defining characteristics of an SoS and, crucially, opens the door to a rich body of established management theory and practice.

In this view:

- Each constituent system is abstracted into a node. This abstraction allows us to treat the internal complexity of the cutter, the helicopter, or the radar as a "black box." For management purposes, we are less concerned with the intricate details of its internal combustion engine or avionics software than we are with its ability to function as a reliable participant in the larger network.
- Each interface, information flow, resource exchange, or physical interaction between constituent systems is abstracted into a link. These links are the connections that make the SoS more than just a collection of parts. They are the pathways through which data is shared, commands are issued, and emergent behavior is enabled. The health, capacity, and security of these links are paramount.

This network perspective directly addresses the five defining characteristics:

- **Operational & Managerial Independence:** By treating each system as a node, we explicitly acknowledge its autonomy. We manage its *participation* in the network, not its internal operations.
- **Geographical Distribution:** A network is, by definition, a structure for connecting distributed entities. Network management is fundamentally about managing connections across distance.
- **Evolutionary Development:** Networks are inherently dynamic. Nodes can join or leave, links can be established or broken. The network perspective embraces this dynamism as a normal state of affairs.
- **Emergent Behavior:** Emergence arises from the *structure* and *dynamics* of the links between nodes, not from the nodes themselves. By focusing on the network, we focus on the very substrate from which emergent capabilities are born. To manage emergence, we must manage the network.

The power of this abstraction is that it reframes the SoS management problem from "How do we control a collection of independent systems?" to "How do we manage a network of interconnected nodes to ensure it delivers its desired capabilities?" This reframing is not just semantic. It allows us to import decades of cumulative knowledge and practice from the field of telecommunications network management, a domain that has been grappling with remarkably similar challenges—faults, configuration, performance, security—in large-scale, distributed, heterogeneous networks for over forty years.

4. A Path Forward: An SoSE Management Framework Inspired by Network Management

Having established the utility of the network perspective, we now turn to the core contribution of this paper: the construction of a novel, comprehensive management framework for SoSE by adapting the proven FCAPS model from network management.

4.1. Introducing the FCAPS Model

The FCAPS model is the International Organization for

Standardization's (ISO) framework for network management, defined as part of the Telecommunications Management Network (TMN) model [5]. It provides a conceptual taxonomy of the five primary functional areas required to manage a telecommunications network. For decades, it has served as the foundation for network management platforms, protocols (like SNMP), and organizational structures in the telecommunications and IT industries. Its five domains are:

- **Fault Management:** The goal is to detect, isolate, and correct abnormal operations or faults in the network. This involves monitoring for alerts, logging errors, running diagnostics, and initiating recovery procedures to ensure network availability and reliability.
- **Configuration Management:** This involves tracking and managing the configuration of network devices and the network topology. It includes inventory management, software version control, and the ability to remotely configure devices to ensure they are operating correctly and consistently.
- **Accounting Management:** This function deals with tracking network usage for purposes of billing, quota management, and cost allocation. It enables network operators to understand who is using what resources and to charge for that usage or enforce usage limits.
- **Performance Management:** The aim is to monitor and measure various aspects of network performance, such as throughput, latency, jitter, and error rates. This data is used to ensure that the network is meeting its service level agreements (SLAs) and to identify bottlenecks or trends that might require capacity planning.
- **Security Management:** This encompasses controlling access to network resources, managing user authentication and authorization, and ensuring the confidentiality and integrity of data transmitted over the network. It also involves maintaining security logs and responding to security incidents.

The enduring value of FCAPS lies in its comprehensive and orthogonal categorization. Each domain addresses a distinct and essential aspect of management, and together they cover the entire lifecycle of network operation and evolution.

4.2. Tailoring FCAPS for the System-of-Systems Context

The central thesis of this paper is that these five functional areas are not only relevant but are directly mappable to the challenges of SoSE management. By carefully translating each domain from the language of network devices and protocols to the language of constituent systems and interfaces, we can construct a robust and holistic management framework for SoS. Table 1 presents this mapping.

Table 1: Mapping the FCAPS Model to SoSE Management

FCAPS Domain	Original Network Management Context	Adapted SoSE Management Context
Fault Management	Detect, isolate, and correct faults in network devices and links (e.g., a failed router, a severed cable).	SoS Health & Anomaly Management: Detect, diagnose, and mitigate failures or degraded performance in constituent systems *or* the critical interfaces/links between them. The focus is on the impact of any fault on the overall emergent capabilities of the SoS. The goal is to maintain mission-level effectiveness, not just component uptime.
Configuration Management	Track and manage device configurations (firmware, settings), software versions, and the network topology.	SoS Interface & Evolution Management: Managing the "configuration" of the SoS network itself. This includes version control of all interface protocols and data models, managing the integration and test of new or upgraded constituent systems ("nodes" joining/leaving), and tracking the evolving topology and dependencies of the SoS over time. It is the discipline of controlled evolution.
Accounting Management	Track network resource usage (bandwidth, storage) for billing, quota enforcement, and cost allocation.	SoS Resource & Contribution Management: Tracking the usage and allocation of shared resources across the SoS (e.g., a common data link, a shared sensor network, a central data repository). In non-commercial SoS (like defense), this is less about billing and more about ensuring equitable access, managing contention, optimizing resource utilization, and understanding each constituent system's contribution to the whole.
Performance Management	Monitor network metrics (latency, throughput, jitter) to ensure Service Level Agreements (SLAs) are met.	SoS Capability & Effectiveness Management: Continuously monitoring the emergent behaviors and overall capabilities of the SoS against its intended mission objectives. This involves defining and measuring SoS-level Key Performance Parameters (KPPs). The question is not "Is the network fast?" but "Is the interdiction mission succeeding?"
Security Management	Control access to network devices, manage user authentication, and ensure data confidentiality/integrity.	SoS Trust & Interoperability Security: Managing security policies and trust relationships across a set of independently managed, autonomous systems. This includes ensuring that the interconnection doesn't create new, exploitable vulnerabilities, managing cross-domain access controls, and maintaining data integrity and confidentiality as information flows between systems with different security postures.

This adapted framework, which we will refer to as SoSE-FCAPS, provides a structured and comprehensive view of the management functions required for any large-scale SoS. It moves beyond simply listing challenges to providing a functional taxonomy that can be used to design a management organization, develop supporting tools and processes, and diagnose problems.

The power of this framework lies in its systemic nature. A failure in any one of these domains can degrade or cripple the entire SoS. For example, poor Configuration Management (e.g., allowing incompatible interface versions) will inevitably lead to Performance Management problems (e.g., data corruption, mission delays) and potentially even Security Management vulnerabilities. SoSE-FCAPS encourages managers to see these interdependencies and manage them holistically.

5. Validating the Framework: The Integrated Deepwater System Case Study

To demonstrate the practical utility and validity of the SoSE-FCAPS framework, this section applies it as an analytical lens to a real-world, well-documented SoS: the U.S. Coast Guard's Integrated Deepwater System program. By examining the program's well-known failures through this structured lens, we can illustrate the framework's diagnostic power and its ability to provide a coherent explanation for complex systemic failures.

5.1. Background: A Cautionary Tale of SoS Complexity

In the late 1990s, the U.S. Coast Guard faced a critical problem. Its fleet of cutters and aircraft, many dating back to the 1960s, were aging and increasingly expensive to maintain. In response, it conceived the Integrated Deepwater System program, one of the most ambitious and expensive acquisition programs in its history, with an estimated lifecycle cost of over $24 billion [15].

Deepwater was not intended to be a simple replacement of old assets. It was explicitly conceived as a "system of systems." The goal was to integrate new cutters, new aircraft, unmanned aerial vehicles (UAVs), and a sophisticated Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) network into a single, seamlessly integrated force. The cutter, the helicopter, and the shore-based command center were to be linked by the C4ISR network, enabling them to share real-time data and coordinate operations with an efficiency never before achieved by the Coast Guard.

In 2002, the Coast Guard awarded a single, 25-year system integration contract to a consortium led by Lockheed Martin and Northrop Grumman, a novel approach intended to foster a holistic, SoS perspective from the outset [16]. Despite this innovative contracting strategy and the best intentions of all parties involved, the Deepwater program became a textbook example of SoS management failure. By the mid-2000s, the program was plagued by severe cost overruns, schedule delays, and most damningly, critical performance and safety failures. The most infamous of these was the discovery that the new 123-foot patrol boats, a key asset, had serious design flaws that caused them to crack and leak fuel, rendering them unsafe for the harsh open-ocean conditions they were designed to operate in [17]. A subsequent government investigation revealed widespread mismanagement, and the program was eventually restructured, with the Coast Guard assuming a much stronger in-house systems engineering role [18].

5.2. Applying the SoSE-FCAPS Lens to Deepwater

The traditional post-mortem of the Deepwater program often cites factors like poor oversight, contracting failures, and unrealistic requirements. While accurate, these explanations are somewhat generic. The SoSE-FCAPS framework allows for a more precise, structured, and systemic diagnosis by categorizing the program's problems according to the five core management domains.

5.2.1. Fault Management

The Deepwater program's most spectacular failures fall squarely within the domain of Fault Management. The goal of SoS Fault Management is not just to detect that a constituent system is broken, but to diagnose the impact of that fault on the overall SoS capability and to correct it.

The cracking of the 123-foot patrol boat hulls is a classic example of a fault that was detected too late and whose impact on the entire Deepwater enterprise was catastrophic. The focus of testing and oversight had been on individual components and platforms, not on the emergent behavior of the vessel in its intended operational environment. When the fault was finally detected, it revealed a systemic failure in the Fault Management function. There were no adequate processes in place to continuously monitor the "health" of the cutters in the context of their mission, to detect the early signs of structural stress as a "fault" in the SoS's ability to perform its patrol mission, or to isolate and correct the problem before it became a fleet-wide crisis. The fault was managed as a problem with a single platform, when in reality it was a failure of the entire SoS to deliver its promised capability.

5.2.2. Configuration Management

The Configuration Management failures in Deepwater were equally profound. The SoSE-FCAPS framework defines this as managing the "network configuration"—the interfaces, protocols, and dependencies that link the nodes together.

A key requirement was that the new helicopters be able to operate from the decks of the new cutters. This required careful configuration management of the physical interfaces (landing decks, hangars) and the electronic interfaces (data links, command and control protocols) between the two systems. However, because the cutter and the helicopter were developed by different teams within the consortium, with different schedules and priorities, their configurations drifted apart. By the time both were ready for deployment, it was discovered that they could not operate together effectively [19]. The helicopters were too large for some cutters' hangars, and the data links were incompatible. This was a direct failure to manage the *configuration of the interface*—the very link that was supposed to create the integrated capability. The SoS had been configured as two separate systems, not as an integrated network.

5.2.3. Accounting Management

Accounting Management in an SoS context is about managing shared resources and understanding contribution. In Deepwater, the most critical shared resource was the C4ISR network and its associated bandwidth. All assets—cutters, aircraft, shore facilities—were to be nodes on this common information network.

The management failure here was a lack of insight into and control over how this shared resource was being allocated and used. As different assets were developed, their demands on the C4ISR network likely grew in uncoordinated ways. Was there a clear process for allocating bandwidth between a helicopter streaming video and a cutter transmitting radar data? When the network became congested, whose mission-critical data got priority? The absence of a robust Accounting Management function meant that the consortium could not effectively manage this shared resource, leading to inevitable performance degradation and conflicts. It failed to track and manage the "invisible" resource that made the SoS more than just a collection of platforms.

5.2.4. Performance Management

The Deepwater program's Performance Management failures were foundational. The program, and the Coast Guard, focused heavily on platform-level performance metrics—the top speed of a cutter, the range of an aircraft, the power of a

radar. However, they failed to adequately define, baseline, and monitor SoS-level performance metrics [16].

What was the desired emergent capability? It was not a fast cutter or a long-range helicopter; it was effective drug interdiction, successful migrant rescue, and robust homeland security patrol. These are SoS-level Key Performance Parameters (KPPs). The program did not establish clear, measurable baselines for these capabilities at the outset, nor did it put in place the continuous monitoring systems to track whether the Deepwater network was actually improving them. Without this Performance Management function, the program managers were flying blind. They could report on the delivery of platforms but had no way of knowing if they were actually delivering the promised capability. The discovery that the new cutters could not perform their patrol mission effectively was not just a Fault; it was a catastrophic failure of Performance Management.

5.2.5. Security Management

While less prominent in the public record than the physical failures, Security Management challenges were inherent in the Deepwater concept. Integrating systems from multiple contractors, with different security postures, into a single C4ISR network created a vastly expanded attack surface.

The management challenge was one of **trust**. How do you ensure that the data from a contractor-provided UAV is trustworthy and has not been compromised? How do you manage access controls so that a helicopter pilot has the data needed for a mission but cannot inadvertently access sensitive intelligence systems on a cutter? The very act of creating the network creates new vulnerabilities. A robust Security Management function would have required the establishment of a common security architecture, the continuous monitoring of network traffic for anomalies, and the management of digital certificates and access rights across all constituent systems. The absence of such a function in the program's management structure almost certainly left the Deepwater network more vulnerable than the sum of its parts.

5.3. Analysis and Findings

Applying the SoSE-FCAPS framework to the Integrated Deepwater System yields a powerful and coherent diagnosis. The program's failure was not a single event or a simple mistake. It was a systemic collapse across all five essential management domains. The framework reveals that:

1. **The failures were interdependent.** Poor Configuration Management (incompatible cutter/helicopter interfaces) directly led to a Performance Management failure (inability to execute missions). The absence of robust Fault Management (delayed detection of hull cracks) was compounded by the lack of SoS-level Performance Management (no metrics to flag the capability loss). A failure in one domain cascaded into others.
2. **The program lacked a holistic view.** The management focus was almost exclusively on the constituent systems (the nodes). The Deepwater program managers, the Coast Guard, and the prime contractors failed to establish a parallel focus on managing the *network*—the links, the shared resources, and the emergent properties. They managed the parts, not the whole.
3. **The framework provides a diagnostic checklist.** The SoSE-FCAPS framework could have been used proactively by the Deepwater program office to structure its oversight activities. It could have asked: "Who is responsible for SoS-level Fault Management? What are our SoS-level Performance metrics? How are we managing the configuration of the interfaces? Do we have an Accounting function for shared C4ISR resources? What is our Security Management architecture for cross-system trust?" The absence of clear answers to these questions would have revealed the management gaps long before they manifested as physical and operational failures.

This case study demonstrates that the SoSE-FCAPS framework is not merely an academic exercise. It is a practical and powerful tool for understanding, diagnosing, and potentially preventing the kinds of systemic failures that have plagued complex SoS acquisitions.

6. Discussion: Implications and Future Research

The preceding analysis has introduced a novel management framework for SoSE and validated its diagnostic power through a real-world case study. This section discusses the broader implications of this work for both practitioners and researchers and outlines a roadmap for future research.

6.1. Implications for Practitioners

For program managers, systems engineers, and architects working on complex SoS, the SoSE-FCAPS framework offers several practical benefits:

- **A Structured Management Checklist:** The framework provides a comprehensive and memorable checklist of the five core functional areas that must be addressed in any SoS management plan. It moves beyond ad-hoc management to a structured, systematic approach.
- **A Diagnostic Tool:** As demonstrated with Deepwater, the framework can be used to analyze existing or troubled programs. By mapping observed problems to the five domains, managers can identify underlying systemic issues rather than just treating symptoms.
- **An Organizational Design Template:** The five domains can be used to structure the SoS management organization itself. A program office could establish distinct teams or working groups responsible for SoS Fault, Configuration, Accounting, Performance, and Security, ensuring that each critical function has dedicated ownership.
- **A Communication and Negotiation Tool:** The framework provides a common language for discussions between the SoS manager and the managers of the independent constituent systems. Instead of vague demands for "better cooperation," the SoS manager can frame requests in terms of specific management domains, e.g., "We need to establish a joint Configuration Management process to govern interface changes."

6.2. Implications for Researchers and Future Research

This paper opens several avenues for future scholarly investigation. The proposed framework is a conceptual starting point, and significant work is required to operationalize it fully.

- **Development of SoSE-FCAPS Metrics and Tools:** Each of the five domains requires the development of specific metrics and supporting tools.

For example, what are the right metrics for SoS-level Performance Management? How can we measure the "health" of an SoS in the Fault Management domain? Research is needed to define quantifiable indicators for each domain and to develop the modeling and simulation tools to collect and analyze them.

- **Application to Different SoS Types:** Maier and others have identified different categories of SoS, such as *directed* (centrally managed), *acknowledged* (managed with agreed-upon goals), and *collaborative* (with no central management) [20]. Future research should explore how the SoSE-FCAPS framework must be tailored for each of these contexts. The management of a directed SoS will look very different from that of a collaborative one, and the framework must be flexible enough to accommodate this.
- **Integration with Architecture Frameworks:** How does this management framework relate to and integrate with existing architecture frameworks like UAF? UAF excels at describing the *static structure* of an SoS. SoSE-FCAPS provides a framework for managing its *dynamic behavior* over time. Research could explore how to use UAF models as the "configuration baseline" for the Configuration Management function, or how to link performance metrics in the framework back to operational viewpoints in UAF.
- **Automation and AI in SoS Management:** The scale and complexity of future SoS will likely overwhelm purely human management. Future research should investigate the role of automation and artificial intelligence in supporting each FCAPS domain. Could AI algorithms be used for real-time Fault Detection and diagnosis in an SoS? Could machine learning optimize resource allocation in the Accounting domain? Could automated reasoning tools help manage the complexity of Configuration Management across hundreds of evolving systems?

7. Conclusion

This paper has addressed a critical gap in the field of System-of-Systems Engineering: the absence of a coherent, integrated management framework to guide the continuous operation and evolution of these complex entities. We have argued that while the field has made significant progress in architectural design, it has not developed a parallel body of knowledge for management that can keep pace with the relentless technological and operational changes that characterize modern SoS.

Our central contribution is the proposal of a novel management framework derived from a fundamental reconceptualization of the SoS as a **network**. This perspective allows us to leverage the proven principles of network management. By carefully adapting the ISO's FCAPS model—Fault, Configuration, Accounting, Performance, and Security management—to the unique context of SoSE, we have constructed a comprehensive and structured framework, **SoSE-FCAPS**, that addresses the full spectrum of management functions required for any large-scale SoS.

The validity and utility of this framework were demonstrated through an in-depth case study of the U.S. Coast Guard's Integrated Deepwater System. The application of the SoSE-FCAPS lens revealed that the program's well-documented failures were not a collection of isolated mistakes, but a systemic collapse across all five management domains. The framework provided a structured and coherent diagnosis, highlighting the interdependence of the failures and the program's fatal focus on managing individual systems at the expense of managing the network between them.

The path forward for SoSE management lies in embracing this holistic, network-centric view. For practitioners, the SoSE-FCAPS framework offers a practical checklist, a diagnostic tool, and a template for organizational design. For researchers, it opens a rich agenda for future work in metrics development, domain-specific tailoring, integration with architecture frameworks, and the application of AI. As the systems we build to serve society become ever more interconnected and interdependent, the ability to manage them effectively will cease to be a specialized skill and will become a fundamental necessity. The SoSE-FCAPS framework provides a robust foundation for meeting that challenge.

References

1. Blanchard BS, Fabrycky WJ. Systems Engineering and Analysis. 5th ed. Upper Saddle River (NJ): Prentice Hall; 2010.
2. Sage AP, Cuppan CD. On the systems engineering and management of systems of systems and federations of systems. Inf Knowl Syst Manag. 2001;2(4):325-345.
3. Department of Defense. DoD Architecture Framework Version 2.02. Washington (DC): DoD Deputy Chief Information Officer; 2010.
4. Dahmann J, Rebovich G, Lane JA. Systems engineering for capabilities. CrossTalk J Def Softw Eng. 2008;21(11):4-9.
5. International Telecommunication Union. ITU-T Recommendation M.3400: TMN Management Functions. Geneva: ITU; 1997.
6. Maier MW. Architecting principles for systems-of-systems. Syst Eng. 1998;1(4):267-284.
7. Object Management Group. Unified Architecture Framework (UAF) Domain Meta Model (DMM) Version 1.1. Needham (MA): OMG; 2019.
8. Friedenthal S, Moore A, Steiner R. A Practical Guide to SysML: The Systems Modeling Language. 3rd ed. Waltham (MA): Morgan Kaufmann; 2014.
9. Dahmann J, Kelley M. The role of systems engineering in the acquisition of systems of systems. Monterey (CA): Acquisition Research Program, Naval Postgraduate School; 2016.
10. Lane JA, Valerdi R. Synthesizing SoS concepts for use in cost modeling. Syst Eng. 2007;10(4):297-308.
11. Baldwin WC, Sauser BJ. Modeling the characteristics of system of systems. In: 2009 IEEE International Systems Conference; 2009. p. 1-6.
12. Bodeau D, Graubart R. Cyber Security Engineering for Systems of Systems. Bedford (MA): MITRE Corporation; 2011.
13. National Research Council. NextGen Air Transportation System: Progress and Challenges. Washington (DC): National Academies Press; 2013.
14. Alberts DS, Hayes RE. Understanding Command and Control. Washington (DC): CCRP Publication Series; 2006.
15. Government Accountability Office. Coast Guard:

Deepwater Program Acquisition Schedule Update Needed. Washington (DC): GAO; 2004. Report No.: GAO-04-695.
16. O'Rourke R. Coast Guard Deepwater Program: Background, Oversight Issues, and Options for Congress. Washington (DC): Congressional Research Service; 2007.
17. Government Accountability Office. Coast Guard: Progress Being Made on Deepwater, but Risks Remain. Washington (DC): GAO; 2007. Report No.: GAO-07-460.
18. Department of Homeland Security, Office of Inspector General. The Coast Guard's Deepwater Program. Washington (DC): DHS OIG; 2007. Report No.: OIG-07-39.
19. Eklund N. The Deepwater debacle: a case study in systems engineering failure. INCOSE Int Symp. 2008;18(1):1562-1576.
20. Dahmann J, Baldwin KJ. Understanding the current state of US defense systems of systems and the implications for systems engineering. In: 2008 2nd Annual IEEE Systems Conference; 2008. p. 1-7.

How to Cite This Article
Baeroom AA, Mansoor HM, Alghallabi L, Meknaci MEF. System-of-Systems Engineering Management: A Review of Modern History and a Path Forward. Int J Manag Organ Res. 2026 May-Jun;5(3):78-86.
doi:10.54660/IJMOR.2026.5.3.78-86.