


Article

Ontology-Based Architecture Process of System-of-Systems: From Capability Development to Operational Modeling

Yimin Feng ¹, Qiang Zou ^{1,2,*}, Chenchu Zhou ³, Yusheng Liu ^{1,*} and Qibo Peng ⁴¹ State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310027, China² Fifth Electronic Research Institute of Ministry of Industry and Information Technology (MIIT), MIIT Key Laboratory of Industrial Software Engineering Application Technology, Guangzhou 510610, China³ Xi'an Aerospace Propulsion Institute, Xi'an 710100, China⁴ China Astronaut Research and Training Center, Beijing 100094, China

* Correspondence: qiangzou@cad.zju.edu.cn (Q.Z.); yslu@cad.zju.edu.cn (Y.L.)

Abstract: System-of-systems (SoS) architecture is crucial in managing complex and interconnected systems. However, the description and modeling of SoS architecture pose significant challenges and require a structured and organized approach. In this study, a metamodel for SoS architecture that considers both structural and behavioral perspectives is defined. The metamodel is then mapped to ontologies that are enhanced with a flow-based extension to characterize architecture views. On this basis, an SoS capability ontology (SoSCO) and an SoS operational ontology (SoSOO) are built with factors, relationships, and flows. A four-step architecture process for developing capabilities and a five-step architecture process for operational modeling are provided based on the ontologies. The proposed approach is applied in a search and rescue case study, demonstrating its ability to improve operability in the early design stage. The process is implemented using the Unified Architecture Framework (UAF) so that various stakeholders and engineers can better understand and develop an SoS.

Keywords: system of systems engineering; ontologies; model-based system engineering; architecture modeling; UAF



Citation: Feng, Y.; Zou, Q.; Zhou, C.; Liu, Y.; Peng, Q. Ontology-Based Architecture Process of System-of-Systems: From Capability Development to Operational Modeling. *Appl. Sci.* **2023**, *13*, 5419. <https://doi.org/10.3390/app13095419>

Academic Editors: Jinzhi Lu, Jérôme Morio, Weiping Wang, Tao Wang and Meigen Huang

Received: 17 February 2023

Revised: 3 April 2023

Accepted: 23 April 2023

Published: 26 April 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The advancement of high technology, particularly in information technology, has led to a significant increase in the frequency and closeness of connections and interactions between systems. As a result, system of systems (SoS) [1,2] and SoS engineering (SoSE) [1,2] have emerged as new areas of research in various fields such as military [3], aerospace [4], and health care [5], among others. The concept of architectural design in civil engineering has inspired the integration of architecture into SoSE to manage complexity and aid in system development and integration [6]. Architecture is the conceptual model that describes the fundamental properties of a system in its environment, including its components, relationships, and principles for design and evolution [7,8]. More particularly, an SoS architecture encompasses not solely its constituent systems (CSs) but also the interconnections and communication processes among them via their interfaces [9]. An Architecture Framework (AF) is a set of conventions and common practices established to guide the description of architecture within a specific domain [7]. In essence, the AF comprises many viewpoints (metamodel) covering various stages of the system life cycle and generating architecture views (models) to address the concerns of stakeholders [10].

AFs have been developed to model the complex relationships in SoSs and describe their architecture [11]. However, AFs are complex and time-consuming to create and implement, requiring significant resources such as time, money, and expertise [12]. The values of architecture descriptions are still under debate due to the construction of architecture views (or simply, views) and the development process of architecture viewpoints (or simply,

viewpoints). The construction of views is often based on predefined scenarios and lacks analytical support, with vague or incomplete elements such as capability taxonomy, capability dependency, operational performer taxonomy, and operational activity flow present in early design stages [13]. The “Fit-for-Purpose” principle, introduced by the Department of Defense Architecture Framework (DoDAF) [8], simplifies architecture design and reduces development costs. However, successfully implementing this principle depends on clarifying relationships among views. While experienced experts may have more flexibility in applying this principle, less experienced engineers may encounter difficulties due to the lack of straightforward architectural processes.

Proposing an architecture development process for SoSs is challenging due to significant gaps in current modeling practices. Firstly, modeling languages lack well-defined semantics for SoS metamodels, such as Systems Modeling Language (SysML) [14], which limits their rigor [15]. Secondly, the complexity of SoSs requires representing architecture through multiple perspectives from different domains [16] (e.g., strategic, operational, and resource), posing the risks of misunderstandings and difficulties in bridging these domains. Finally, even within a single domain, selecting the appropriate views and determining their order is challenging due to the variation in model kinds (e.g., taxonomy, structure, connectivity, and process) [16].

Although SysML has limitations in describing SoS architecture, it can be enhanced by incorporating ontologies and reasoning as a complement [17,18]. Ontology is a formal and rigorous approach to knowledge representation, which can provide precise and unambiguous semantics for terms [19]. Several scholars and institutions have developed ontologies to depict different domains and applications by integrating the elements of a system’s architecture [20,21]. Building ontologies is a promising way to create a knowledge repository for SoS architectures. At the same time, a metamodel provides the foundation for defining basic concepts and relationships necessary for ontology creation and reasoning [22]. Despite differences in architectural elements across domains, they can be correlated with the formal semantics of ontologies. Views developed using different model kinds can be represented through various flows and relationships, including generalization, composition, dependency, connector, message, and transition. Accordingly, the selection and the sequence of views can be specified based on the organization of these relationships and flows.

Motivated by the observation above, SoS metamodels and ontologies are proposed to provide semantics for a shared understanding. The novelty of our work lies in introducing an ontology-based approach with a flow-based extension to characterize the SoS strategic and operational views of AFs. This approach facilitates the understanding and adoption of architecture frameworks by SoS engineers. The significance of our work lies in providing an explicit comprehension of the relationships between different views through flow-based ontologies. By outlining the architectural process required to connect these views, our approach provides engineers with easy-to-use steps, consequently improving the efficacy of architecture frameworks in practical scenarios.

This paper is organized as follows. Section 2 provides a review of related work. Section 3 introduces SoS metamodels, the SoS capability ontology (SoSCO) based on factors and relationships, and the SoS operational ontology (SoSOO) based on relationships and flows. Section 4 describes the design steps of the strategic architecture. Section 5 illustrates the design steps of the operational architecture. Section 6 discusses the advantages and disadvantages of the proposed approach. Section 7 concludes this study and summarizes future works.

2. Related Work

2.1. Architecture Framework

AFs serve different purposes and are defined using metamodels. Widely used AFs include Zachman [23], the Open Group Architecture Framework (TOGAF) [24], and the Department of Defense Architecture Framework (DoDAF) [8]. Although the Zachman

framework does not explicitly propose the concept of “enterprise architecture”, it is a pioneering multi-perspective framework emphasizing the hierarchical and multi-perspective nature of complex systems’ cognition. TOGAF defines an enterprise as “a collection of organizations with a common goal”. Therefore, an SoS can also be considered a type of enterprise, and AFs are an efficient way to handle SoSs. While the Zachman framework and TOGAF define concepts with ambiguity, DoDAF takes data as the core and defines DoDAF Metamodel (DM2) for different situations. However, DoDAF is tailored by the Department of Defense to its military doctrine and may not apply directly to other countries or civilian domains. Governments and organizations such as the UK and NATO have proposed architecture frameworks, including the Ministry of Defense Architecture Framework (MODAF) [25] and NATO Architecture Framework (NAF) [26], respectively, for their military domains. Matching metamodels and defining concepts that effectively communicate between AFs is challenging due to their differing contents [27]. The Unified Architecture Framework (UAF) inherits and develops DM2, is compatible with existing AFs, and provides a more standardized approach to architecture description. UAF extends the application from the military domain to the general enterprise domain [28]. However, the UAF’s metamodel comprises over 80 views with complex concepts and relationships, which challenges selecting appropriate views and organizing them in proper order. Thus, metamodel should be tailored for modeling SoS views in practice.

2.2. SoS Architecture Development

DoDAF’s “six steps”, MODAF’s “six steps”, and NAF and TOGAF’s Architecture Development Method (ADM) guide how to build the system architecture. However, the specific modeling process lacks clarity, leading to low operability and difficulty for engineers to understand and practice architecture development. UAF version 1.2 [29] presents a UAF-oriented enterprise architecture guideline that provides comprehensive guidance for refining architecture by defining steps to create views. However, hundreds of specific steps are introduced in the nine significant steps defined by the UAF-oriented Enterprise Architecture Guidelines [30], which leads to complexity and ambiguity in applying AFs to model SoS architecture due to many-to-many logical relationships between steps and unclear execution sequences.

Dandashi and Hause [31] discussed using UAF for architecture modeling with only two architecture views. Eichmann et al. [32] proposed an SoS development approach using UAF with the V-model development process. Zhang et al. [33] introduced a UAF-based development method for air traffic management. However, the transition between the domains of AFs is ambiguous or missing, and the selection and order of architecture views in a single domain depend mainly on engineers’ experience. Thus, the architectural approach’s operability and practicality still require improvement.

2.3. Ontology

SysML furnishes a user-friendly graphical syntax, exemplified by the Block Definition Diagram (BDD) that models system concepts and relationships [34], as well as the Internal Block Diagram (IBD) that illustrates intricate system architecture and component interconnections [35]. Nevertheless, it is notable that the language is deficient in formal semantics [36]. Ontology is a formal and rigorous approach to knowledge representation, which can provide precise and unambiguous semantics for terms [19]. Based on description logic, web ontology language (OWL) is commonly used in building an ontology. Wardhana et al. [17] proposed a transformation process to concert a SysML requirement diagram into an OWL ontology file, primarily focusing on requirements in the context of System engineering (SE). However, as SoSE involves multiple independent systems working together as a more extensive system, the transformation process must account for the additional complexity arising from various elements beyond requirements. Zhu et al., introduced an approach to SoS mission modeling and analysis using ontology technology [37]. However, this ontology-based approach that focuses only on mission analysis is difficult to bridge

the strategic and operational domains of the SoS architecture. Baek et al. [22] developed a conceptual metamodel to represent SoS ontologies, which could be a meaningful way to analyze SoS characteristics instead of architectures and hence, cannot be applied to the design stage of SoS architecture.

3. Development of SoS Metamodels and Ontologies

Stakeholders use artifacts, such as diagrams and models, to facilitate communication and understanding in system development [38]. This development process involves activities performed by different organizations, including conceptualization and goal definition. Architecture Frameworks (Afs) are utilized to identify and define necessary artifacts through interrelated metamodels [10]. Developing a metamodel for SoS ontologies can enable engineers to obtain overall domain knowledge and facilitate communication between CS-level managers and engineers, providing a domain-general representation of an SoS [22]. This section presents an SoS metamodel for comprehending the mission structurally and behaviorally. The metamodel is then represented in OWL language, enriched through flow-based extensions.

3.1. Mission-Centric SoS Metamodel Based on DMM

The traditional SysML requirement diagram decomposes requirements in SE, but its use in analyzing the SoS mission should consider the SoS context [39]. SoS mission management guides the entire process of UAF development [28]. Architecture development stages should interact with mission management to ensure domains align with the mission and to update missions iteratively. The decomposition of SoS missions can be achieved by mission-related attributes [39] or the ontology-based mission decomposition method [37]. While detailed sub-missions can be derived from top-level missions, over-complicating the mission phase by adding too many attributes for formalization and decomposition should be avoided. Furthermore, analyzing relevant attributes in the mission phase alone may be insufficient, requiring the support of multiple perspectives.

We propose that one of the intentions of SoS Afs is to translate the SoS mission into SE requirements. To enhance consistency among various SoS stakeholders, UAF defines multiple domains, including strategic (St) and operational (Op), which can be utilized to refine mission analysis. The St domain's capability element is critical in an SoS architecture [40] because it assists in developing solutions-based capabilities [41], and architecture-based capability engineering aids in conceptualizing the capabilities expected to be achieved by the SoS architecture [42]. Therefore, the proposed metamodel in this study supports SoS architecture modeling by describing the Op domain from structural and behavioral perspectives, and capabilities in the St domain bridge these two perspectives.

The UAF Domain Meta Model (DMM) provides the foundation for implementing UAF and is widely used in many applications [10,33,43]. It comprises six elements: Individual, Type, Abstract, Tuple, Enumeration, and External Type, and their meanings are based on concepts proposed in the International Defense Enterprise Architecture Specification (IDEAS) [44]. The proposed SoS metamodel, based on DMM, indicates that missions can be achieved by capabilities from both structural and behavioral perspectives, as depicted in Figure 1. In the SoS metamodel, the asterisks appearing at either end of an association relationship serve to denote the multiplicity of the relationship, indicating the number of instances of one element that can be related to one or more instances of another element.

The representation of a mission in the metamodel, denoted by Type, is linked to other elements through three distinct sets of relationships defined by Tuple. These relationships include trace, refine, and satisfy, which respectively outline the capabilities required to achieve the mission, operational activities needed for behavioral analysis, and resource allocation from a structural perspective. To represent the constituent systems (CS) of the SoS, the Performer is used, which can be physical resources (ResourcePerformer) or logical resources (OperationalPerformer). As SoS is a collective term for multiple CSs, there is no single CS in the metamodel to represent the concept of SoS. Instead, the metamodel

investigates how CSs implement their capabilities to comprehend and describe the SoS. The Process can be captured as an OperationalActivity to represent a logical process or function specified within the context of a ResourcePerformer capable of performing it.

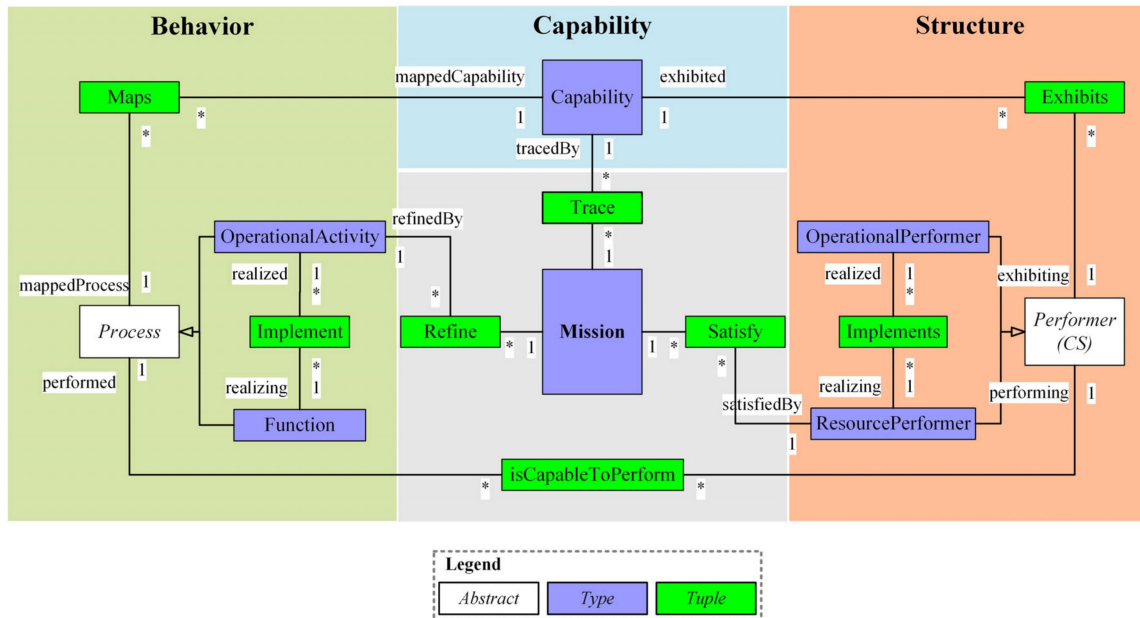


Figure 1. A simplified version of the SoS metamodel.

3.2. OWL-Based Representation of SoS Metamodel

The proposed SoS metamodel comprises elements classified into Abstract, Type, and Tuple. Therefore, the corresponding OWL terms for these element types must be specified in the ontology space, as shown in Table 1. The Protégé ontology editing software (version 5.5.0) [45] models the example elements presented in the SoS metamodel, as shown in Figure 2.

Table 1. SoS metamodel to OWL representation.

Metamodel Term	OWL Term	Example Elements in the SoS Metamodel
Type	Class	Capability, Mission, OperationalPerformer, OperationalActivity
Abstract	Class	Process, Performer
Tuple	Object property	Exhibits, isCapableToPerform, mapsToCapability, trace

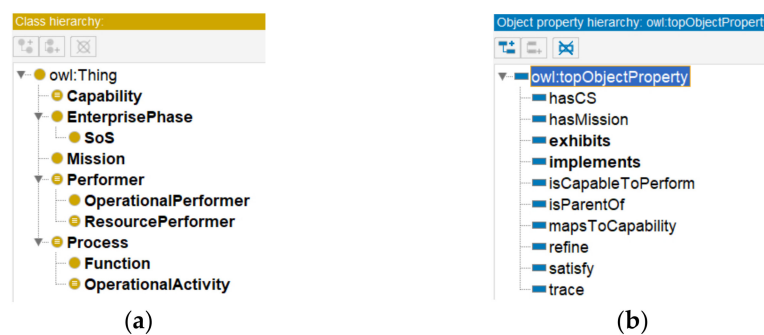


Figure 2. OWL representation. (a) OWL-based representation of the Type and Abstract, (b) OWL-based representation of the Tuple.

The Type is transformed into a class in ontology, as shown in Figure 2a, where an inheritance relationship links the “Performer” class and the “OperationalPerformer” class.

Similarly, the Abstract is transformed into a class in ontology, exemplified by the “Process” and its sub-elements “Function” and “OperationalActivity”. While “Process” is an abstract class and cannot be instantiated in practical cases, serving only for analysis and reasoning purposes. However, the sub-classes “Function” and “OperationalActivity” can be instantiated, representing these instances as individuals in the ontology.

A Tuple means a relationship or flow between two elements and can be directly represented in OWL, which focuses on tuple relationships, such as “A contains B”. Therefore, the Tuple in the metamodel is transformed into an object property in ontology, as illustrated in Figure 2b. For example, the relationship between “Capability” and “Mission” can be represented as an object property named “trace”.

To address the fact that the concept of an SoS is not explicitly represented in the SoS metamodel, an SoS class is added to the ontology using the “EnterprisePhase” stereotype from the UAF DMM. The interconnection between the SoS and its constituent CSs is established through the “hasCS” object property, while the “hasMission” object property is employed to depict the connection between the SoS and its mission.

Based on the above analysis, Figure 3 shows the SoS ontologies modeled in Protégé.

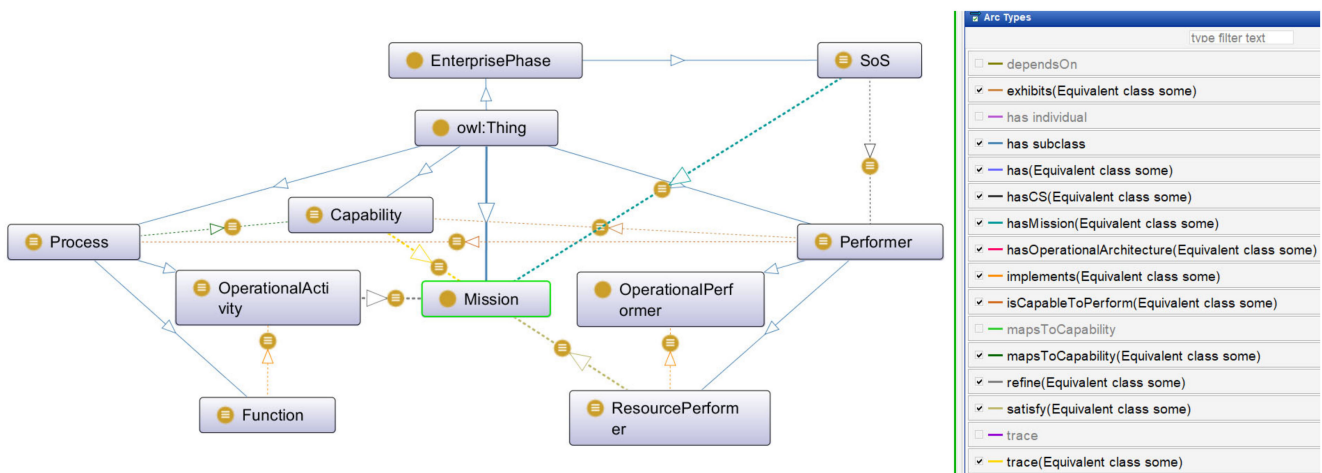


Figure 3. OWL-based representation of SoS ontologies.

3.3. Extending SoS Ontologies Based on Flow

The proposed metamodel and ontology offer fundamental concepts for comprehending SoS but require supplementary semantics to aid the development of architecture views. This section addresses this gap by incorporating relations, flows, and factors into the SoS ontologies. The rationale and related concepts are presented, followed by establishing the SoS capability ontology (SoSCO) and SoS operational ontology (SoSOO) for storing and processing the proposed relationships and flows.

3.3.1. Rationale and Related Concepts

SoS is distinguished from a single system by the managerial and operational independence of the CSs [46]. To fully describe and model SoS, strategic domain views are needed to account for “managerial independence”, and operational domain views are required to address “operational independence” [47]. This study focuses on both strategic and operational domains.

Selecting architecture views and their sequence for a domain is challenging due to various model kinds (e.g., taxonomy, structure, connectivity, and process), capturing unique descriptions of SoS architecture. Flow-based functional representation [46] is a common approach in SE to characterize functions as transformations of flows. However, analyzing SoS architecture requires considering multiple domains and model kinds, and various flows and relationships must be described to reflect the main concerns of the different model kinds.

This study proposes six model kinds and corresponding relationships and flows for them, as shown in Table 2. Instead of solely relying on the model kind’s name, engineers can identify the focus of architecture views by examining the proposed relationships and flows. For example, a generalization relationship is defined as a hierarchical structure whereby a specialized (child) element is developed based on a more generalized (parent) element. The proposed context flow visually links general elements, enhancing the semantic interpretation of the generalization relationship. The context flow does not provide any specific properties of the flow but serves a schematic function within the relationship. A composition represents a whole–part relationship and is a type of aggregation. A dependency relationship between elements, similar to supporting functions [38], represents prerequisites that enable other elements. A connector flow summarizes exchanges of information, systems, personnel, and energy, among other things. A message flow defines specific communication between elements during an interaction. An edge flow signifies the flow of objects or information between activities, while a transition denotes a change in the state of the system or object being modeled.

Table 2. Relationships and flows in selected model kinds.

Model Kinds	Relationships	Flows
Taxonomy (Tx)	Generalization	Context
Structure (Sr)	Composition	/
Connectivity (Cn)	Dependency	Connector
Sequences (Sq)	/	Message
Process (Pr)	/	Edge
State (St)	/	Transition

This study defines relationships and flows based on the Tuple from IDEAS for architecture modeling in the strategic and operational domains. Figure 4 illustrates three relationships for capturing capabilities in the strategic domain and another three relationships and five flows for the operational domain. As illustrated in Figure 1, the performer exhibits capability, represented by activities capturing a logical process. The specific relationships/flows that describe the operational performer are OperationalContext, PerformerGeneralization, PerformerComposition, OperationalConnector, and OperationalMessage, while those used to describe the operational process are ActivityComposition, OperationalActivityEdge, and Transition. Three questions should be answered to construct AFs for SoS architecture based on the relationships and flows in the early design phase:

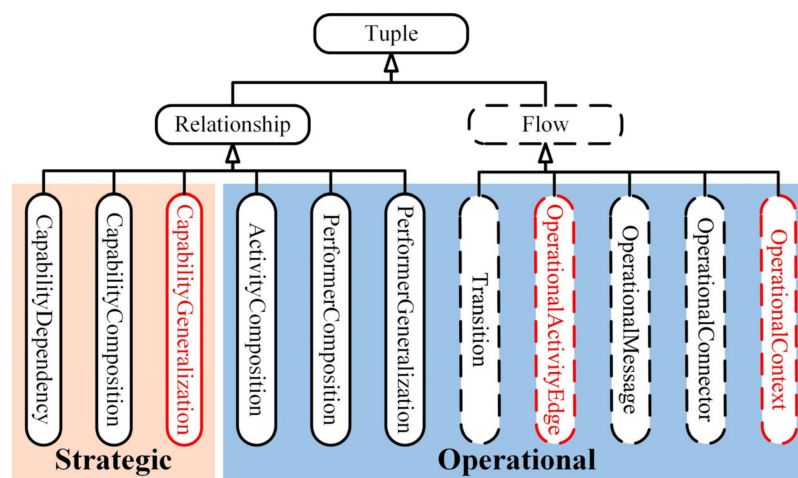


Figure 4. Flow-based representation in strategic and operational domains.

1. How to identify the general and specialized elements of the generalization relationship in the strategic domain?
2. How to identify the context flow in the operational domain based on the capability analysis?
3. How to identify the edge flow's source, target, and swimlanes?

These questions connect the mission and capability analysis, strategic and operational domains, and operational engineering activity. The relationships/flows that correspond to these questions are highlighted in red in Figure 4 and require specific methods for identification and organization.

3.3.2. SoS Capability Ontology (SoSCO)

The UAF defines capabilities as being realized through combinations of ways and means. SoS metamodels frequently refer to capabilities [48–50], such as the “simple” and “complex” types that describe capability attributes [40]. However, these terms remain vague and require necessary attributes and relationships to support the SoS strategic architecture model process. We propose that SoS missions are external demands, while capabilities are internal abilities that enable SoS to adapt to the external environment. Complementary capabilities related to each other through relationships can trace the mission, with each capability characterized by its attributes. Figure 5 presents the core terminology of SoSCO.

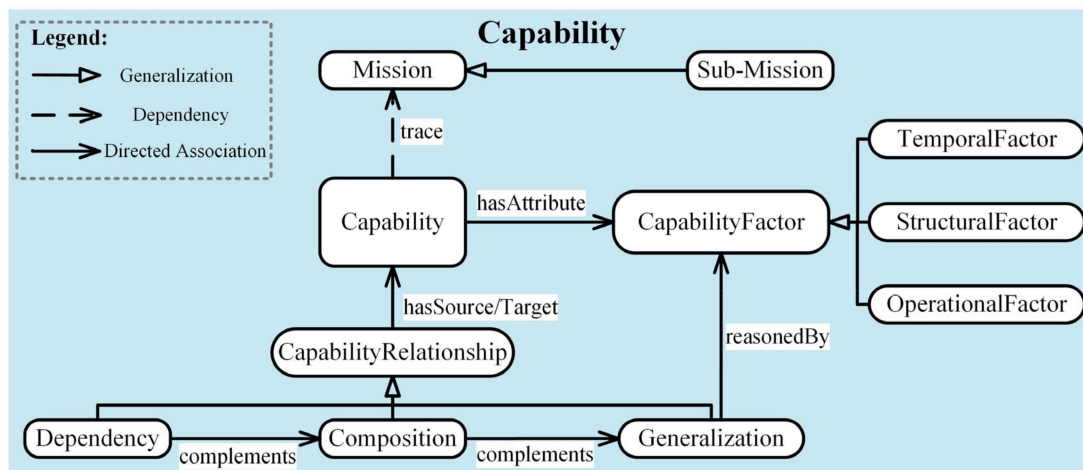


Figure 5. Core terminology of SoSCO.

The sub-mission is inherited from the mission to specify the top-level SoS mission breakdown. The trace relationship illustrates what capabilities should have to realize missions. However, when modeling an SoS with AFs, it is crucial to identify the relationships between capabilities and determine the appropriate modeling order. CapabilityRelationship, which encompasses Generalization, Composition, and Dependency, can be used for this purpose. Additionally, the concepts of CapabilityFactor, including TemporalFactor, StructuralFactor, and OperationalFactor, enrich the semantics of the ontology to reason about the general and specialized elements of the generalization relationship. TemporalFactor is a critical component of ensuring the coherence of capabilities with the strategic phase plan of the SoS. Specifically, TemporalFactor considers time-related factors that may influence the performance of constituent systems within an SoS, such as the duration of operations or the frequency of events. StructuralFactor specifies that capabilities are realized through various combinations of realistic entities. StructuralFactor refers to the physical and logical structure of the constituent systems in an SoS, such as the hardware and software components, the communication protocols, and the data formats. OperationalFactor provides information on the purpose of operational activities and physical functions to enrich the context of the above two factors. After establishing the generaliza-

involving helicopters, naval ships, and civilian sea rescue organizations. A mainstream SoS architecture modeling platform, Cameo Enterprise Architecture, models the SAR case.

The strategic architecture is primarily anchored on the capability defined by the factors and relationships specified in the SoSCO. This section outlines ontology-based modeling steps for strategic architecture, which leverage the SoSCO to identify capabilities that match desired attributes and relationships. Figure 7 illustrates four architectural steps in the strategic domain, where the parallelograms denote elements such as mission and *CapabilityFactor*, and white rectangles represent engineering activities or methods. The green boxes denote the architecture views, while the orange boxes indicate the utilization of *CapabilityRelationship*.

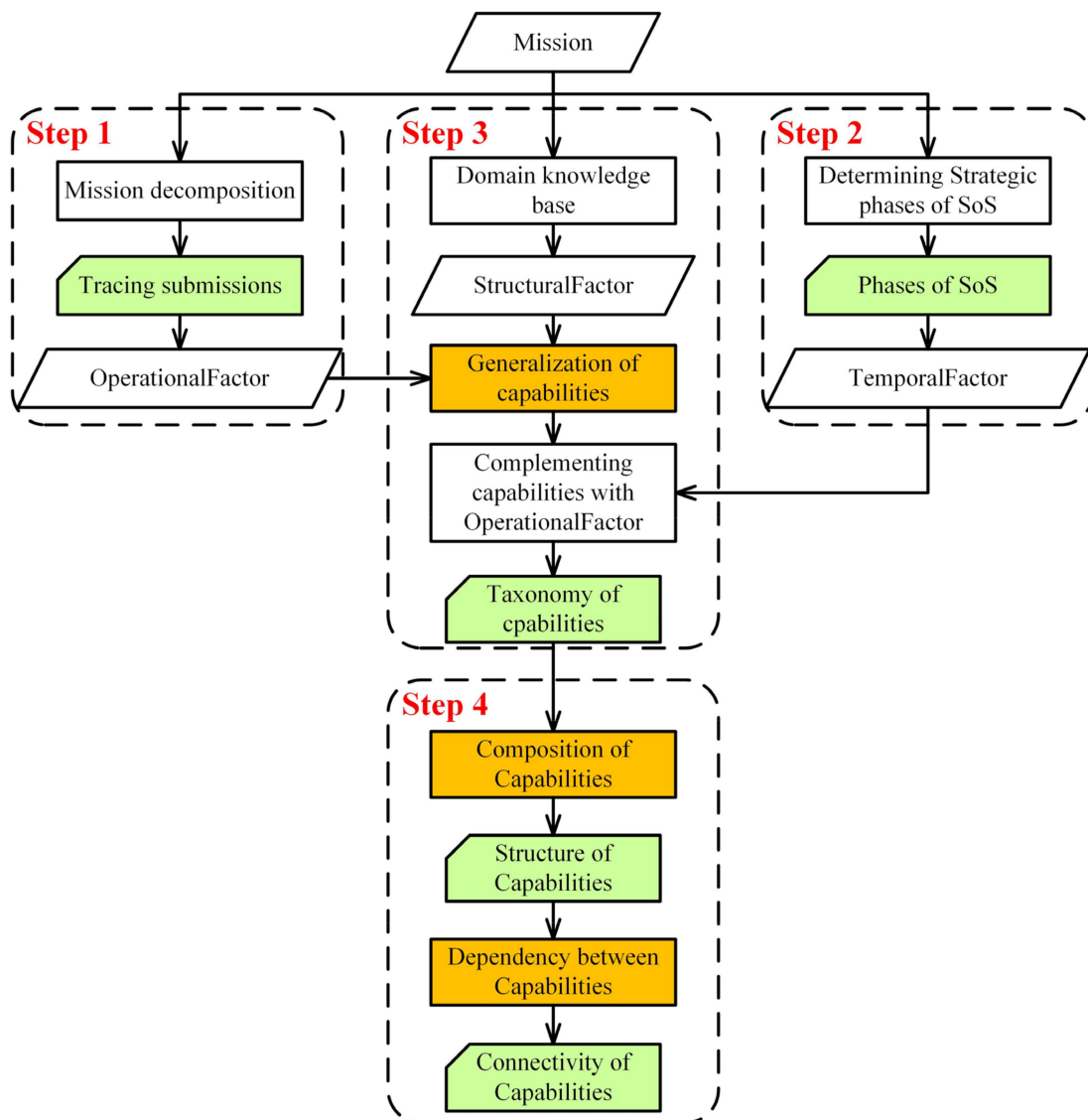


Figure 7. Four-step process of the strategic domain.

4.1. Step 1: Identify Operational Factors with Requirement Analysis

This step aims to connect mission analysis with capability factors to identify the purpose of operational activities and physical functions. It is necessary to gradually decompose the top-level missions and break down complex missions into simple ones. The *OperationalFactor* in the SoSCO can be extracted from sub-missions. Figure 8 displays traceability relationships from strategic domain to missions. A method that creates stereotypes extend-

ing existing meta-classes [53] is applied for modeling SoS missions. The stereotype *Mission* inherits the properties of its super stereotype *Requirement* from SysML.

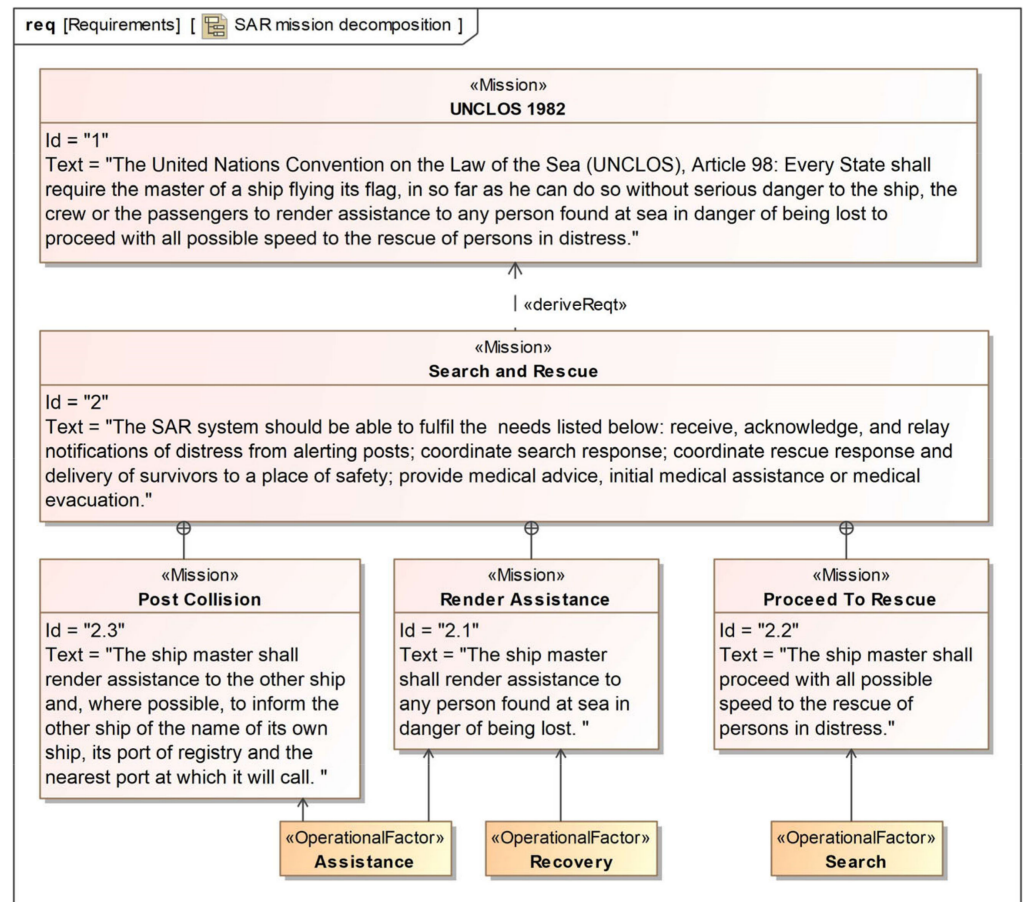


Figure 8. Identify operational factors by tracing sub-missions.

4.2. Step 2: Identify Temporal Factors with Visions and Goals

In this step, the SoS is divided into phases of the development plan, each with specific goals that must be achieved for mission success [16]. The required capabilities to achieve the goals vary across different phases, and the *TemporalFactor* in the SoSCO is used to identify and segment these phases. The number of phases decides the *TemporalFactor* number. Only the number of phases must be determined in the first modeling process. As the SoS architecture is iteratively developed, the models of the phases will be completed by associating them with capabilities, operational architecture, goals, start and end times, and other elements.

The Strategic Process (St-Pr) view of the UAF, implemented by the SysML block definition diagram (BDD), shows the relationship between phases, as illustrated in Figure 9. The SAR SoS has four goals, which can be assigned in three different time-span phases, with each temporal factor corresponding to a phase. Each strategic phase can be instantiated and associated with different capabilities, operational architecture, and other elements. These elements will be gradually refined in subsequent steps.

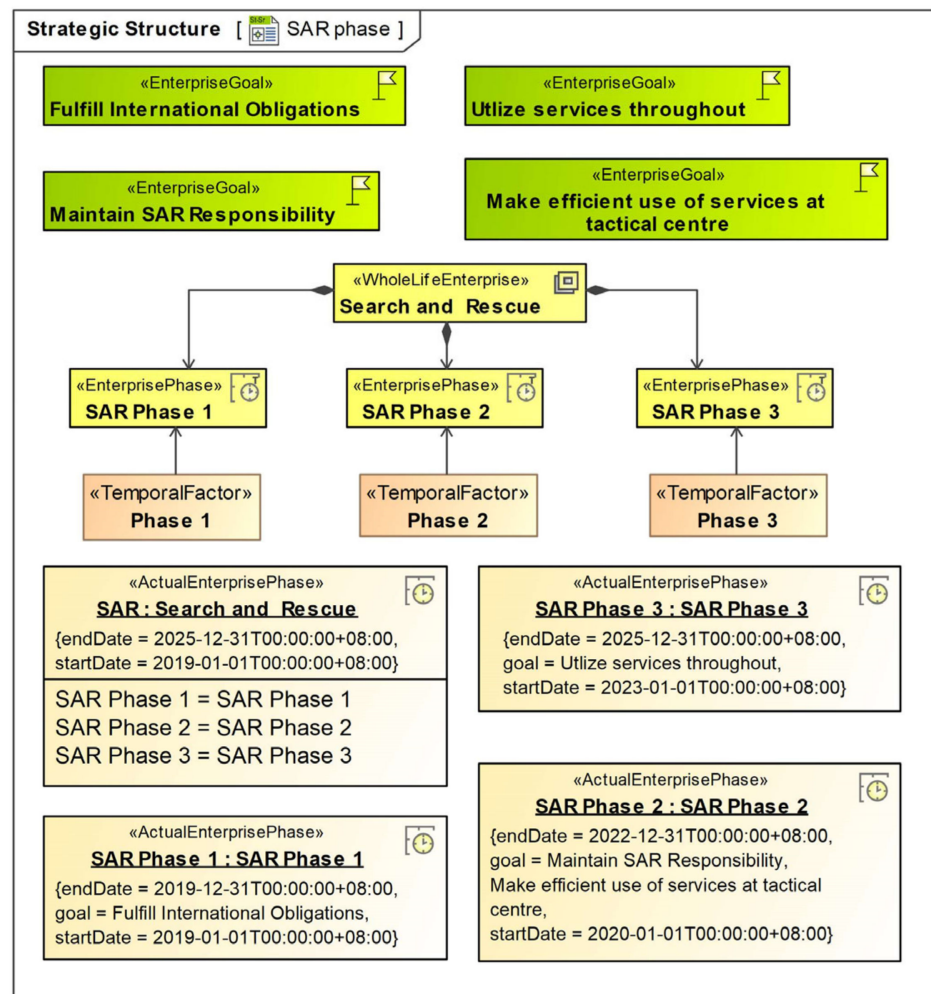


Figure 9. Strategic Processes (St-Pr) of the SoS phases.

4.3. Step 3: Define Capabilities Based on Factors

This step addresses question 1 and involves identifying capabilities by combining the three capability factors using a three-level approach to defining capabilities.

At the top level, basic capabilities are derived from a combination of *TemporalFactors* and *StructuralFactors*. The Semantic Web Rule Language (SWRL) is a rule language that semantic web reasoners can process to enable automated semantic inference [54]. SWRL rules can be specified in an implicit form and are well suited for describing filtering rules [55]. Propositions involved in the rules are represented as atoms of $P(x,y)$, where P is an object property. In $P(x,y)$, the source element x represents the client, and the target element y represents the supplier. Table 3 shows an example of SWRL rules for creating the generalization of capability, where the atom `correspondsToCapability (?sos, ?c)` represents a transformation of OWL class type from *CapabilityFactor* to *Capability*. Figure 10 illustrates the excerpt facts of SoSCO in SAR after reasoning with SWRL rules in Protégé with a Pellet reasoner [56]. SAR includes SAR phase 1 and SAR phase 2 as *TemporalFactor*, and SAR can be further subdivided into maritime and land-based SAR based on *StructuralFactor*. These attributes correspond to various sub-capabilities, and the SoS *Search and Rescue* corresponds to the top-level capability *SAR*. The object property *isParentOf* represents the generalization of capabilities. The capabilities can be modeled in the Strategic Taxonomy (St-Tx) view of the UAF implemented by the SysML BDD, as shown in Figure 11.

Table 3. SWRL rules for the generalization of capabilities.

Comments	SWRL Rules
Generalization reasoned by TemporalFactor	$SoS(?sos) \wedge TemporalFactor(?tf) \wedge owns(?sos, ?tf) \wedge correspondsToCapability(?sos, ?c) \wedge$ $correspondsToCapability(?tf, ?tc) \rightarrow isParentOf(?c, ?tc)$
Generalization reasoned by StructuralFactor	$SoS(?sos) \wedge StructuralFactor(?sf) \wedge owns(?sos, ?sf) \wedge correspondsToCapability(?sos, ?c) \wedge$ $correspondsToCapability(?sf, ?sc) \rightarrow isParentOf(?c, ?sc)$

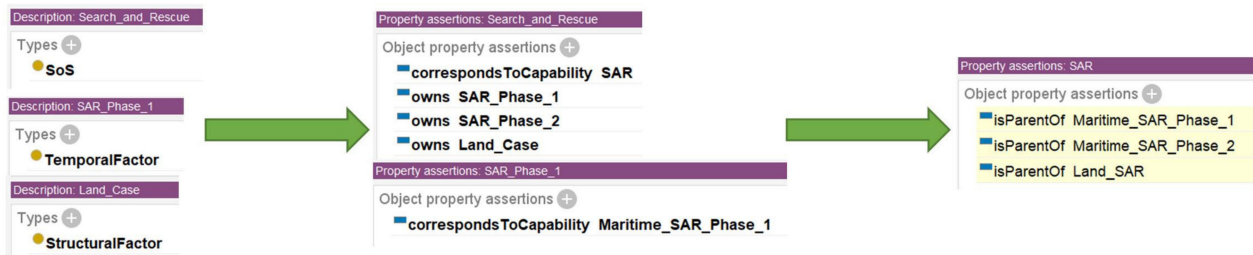


Figure 10. Excerpt of facts of SoSCO.

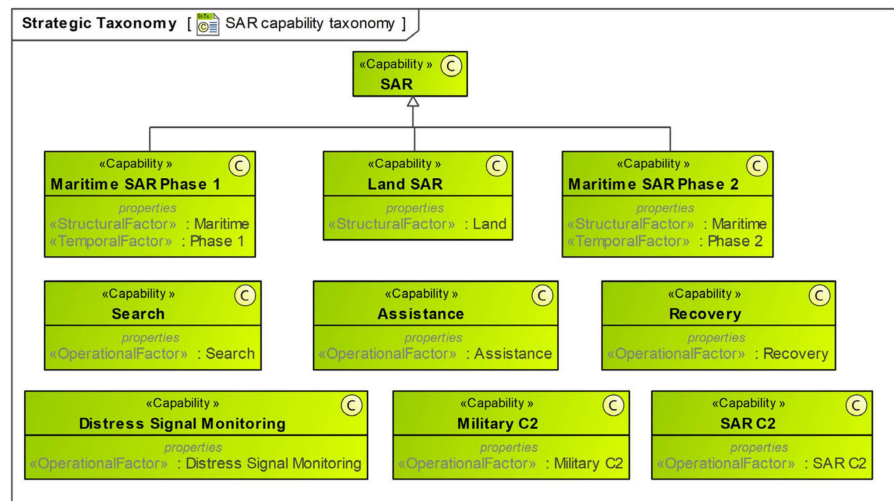


Figure 11. Strategic Taxonomy (St-Tx) of capabilities.

At the middle level, the mission should be decomposed into sub-missions, and the corresponding capabilities characterized by the *OperationalFactor* of the mission view can be traced. Figure 11 shows that the capabilities of search, assistance, and recovery are modeled by verbs based on operational factors of the St-Pr view.

At the bottom level, the captured capabilities are supplemented by additional *OperationalFactors* based on domain knowledge to provide a more comprehensive context. On the one hand, capabilities captured based on sub-mission may lack the information necessary for explaining their relationship with other operational factors. For example, Figure 11 shows that the C2 center should control the search capability and track information from the distress signal monitor. On the other hand, capabilities captured based on *TemporalFactor* and *StructuralFactor* also require additional capabilities based on operational factors. For example, maritime SAR has more capabilities than land-based SAR.

4.4. Step 4: Organize Capabilities Based on Their Relationships

This step involves organizing and representing the capabilities through the composition relationship to establish the top-level capability and sub-capabilities that must be achieved together to attain the overall capability. To this end, a cluster of capabilities based on *OperationalFactor* can complement the ones with the same *TemporalFactor* or *StructuralFactor* in the generalization relationship. The Strategic Structure (St-Sr) view of the

UAF, implemented by the SysML BDD, can be used to display the class diagram of the capability clusters.

To support the analysis of the SoS context, capability correlations need to be identified. Dependencies between capabilities demonstrate that one capability cannot achieve its full potential without the presence of other capabilities [29]. Based on the capability structure obtained in the St-Sr view, domain experts can establish the dependency relationship between capabilities using knowledge about the relationships between *OperationalFactors*. We propose using the Strategic Connectivity (St-Cn) view of the UAF implemented by the SysML internal block diagram (IBD) to address the logical grouping of capabilities within a specific context and their dependencies.

This Strategic Connectivity (St-Cn) view helps group capabilities and display their dependencies, as shown in Figure 12, where SAR command and control (C2) depends on the military C2 capability. Similarly, the assistance, search, and recovery capabilities depend on the SAR C2 capability, which relies on the distress signal monitoring capability. The SysML IBD defines capabilities in a specific context, and the dependencies are scoped accordingly.

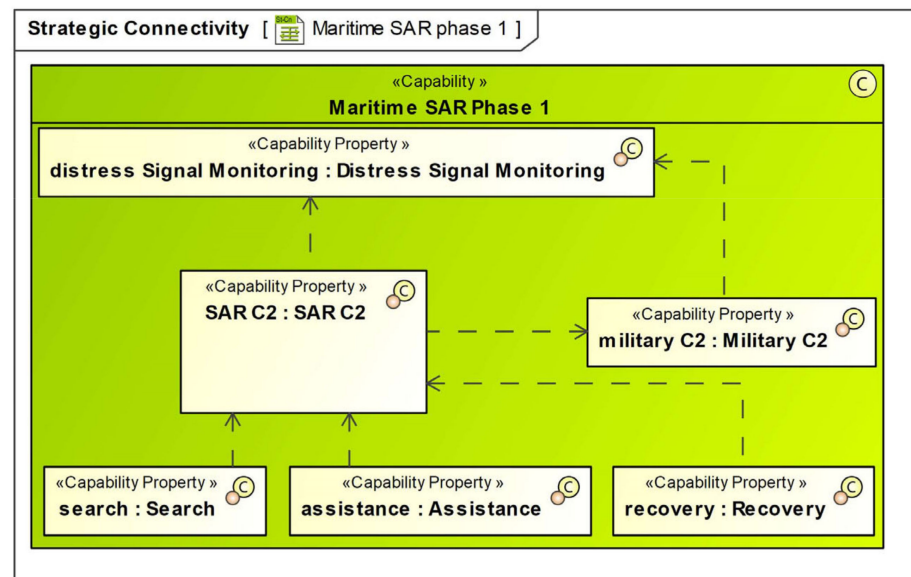


Figure 12. Strategic Connectivity (St-Cn) of capabilities.

5. SoSOO-Based Architectural Steps and Application Example

The operational viewpoints should leverage the outcomes of the strategic viewpoints to identify the operational performers needed to execute the designated capabilities for the SoS [57]. The operational architecture describes the context, operational structure, behavior, and interchanges required to exhibit capabilities. It is imperative that the operational architecture links the capabilities in the strategic domain and explains how to describe the operational structure and behavior. With the SoSOO, a 5-step architecture process for the operational architecture can be devised, as depicted in Figure 13. The white rectangles indicate the engineering activities or proposed methods, while the blue-filled boxes represent the views capturing relationships, and the boxes with dashed lines represent the views capturing flows. The following subsections provide a detailed account of the architecture process and associated view products.

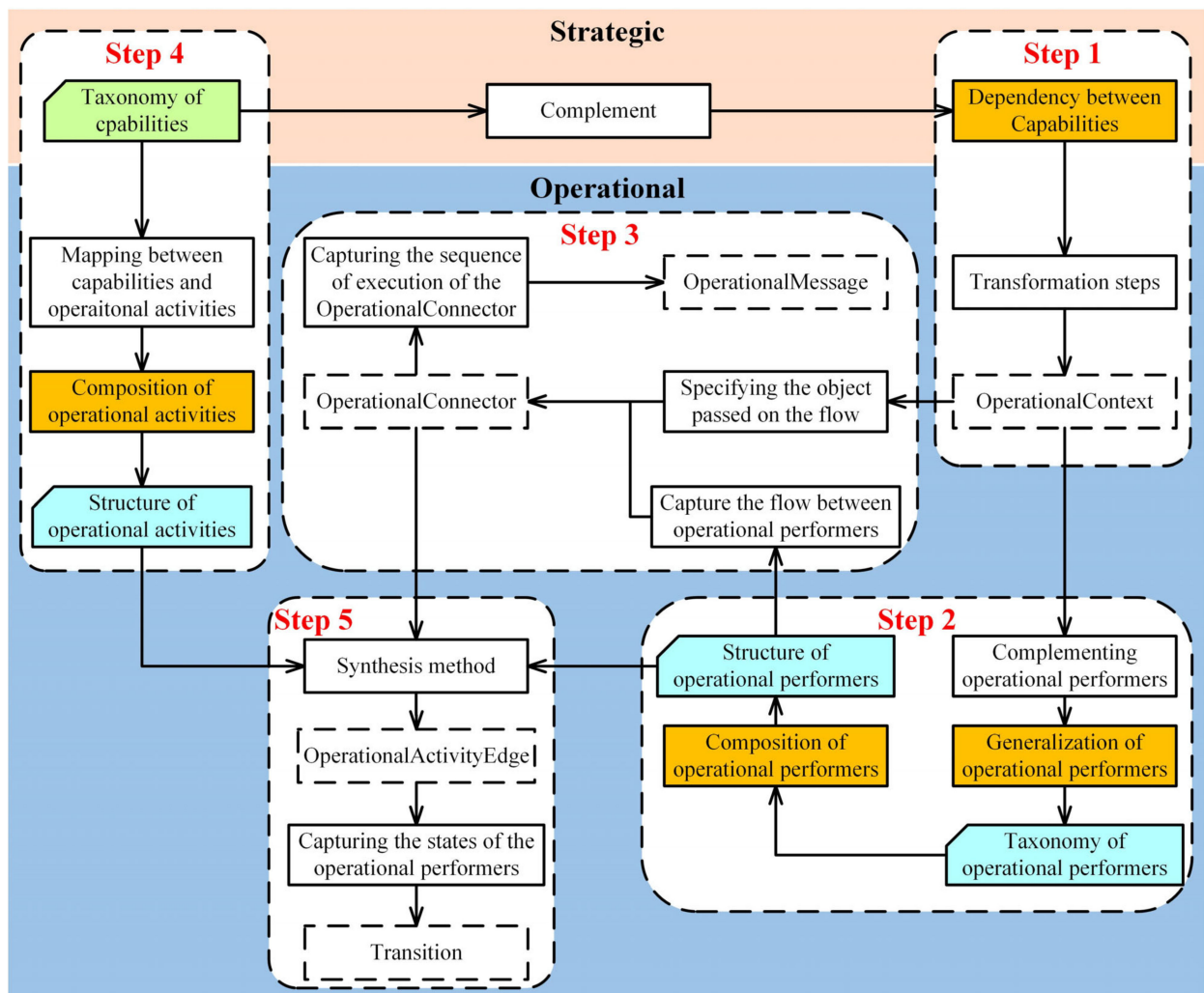


Figure 13. Architectural steps of the operational domain.

5.1. Step 1: Bridge Strategic Domain and Operational Domain by Structural Elements

This step, which addresses question 2, aims to clarify the structure of the SoS context, where the *OperationalContext* is a visual connection between high-level operational concepts. Dependencies between capabilities mean the prerequisites that enable other capabilities, and the supporting functions similar to dependencies can be used for designs based on causal semantics [58] or temporal relations [55]. Accordingly, dependency can be used to reason about *OperationalPerformer* (who) and *OperationalActivity* (how). The transformation method from *Dependency* in SoSCO to *OperationalContext* in SoSOO involves four steps:

1. Use the depth-first search (DFS) algorithm to search for each capability through dependency until the one that does not depend on any other capability is found.
2. Cycle from the deepest to the shallowest capabilities and supplement the context of capabilities by adding the required *OperationalPerformer* and *OperationalContext* between performers. Note that the *OperationalContext* only schematically conveys data/control without an explicit type.
3. When the complement process returns to the outermost capabilities (i.e., those not relied upon by any other capability), list the performers of these capabilities according to the specific operational scenario.
4. Because the *OperationalPerformer* for different capabilities may overlap, merge the above performers and arbitrary connection flows to obtain a high-level operational concept that can be represented as a picture to facilitate understanding of the SoS context.

Figure 14 illustrates the context of maritime rescue by showing the SAR operation involving a yacht in distress at sea. The *OperationalContext* is modeled as a dashed line with an open arrow pointing from flow sources to targets. The diagram shows that the monitoring unit receives the yacht's distress signal and sends it to the C2 center, which coordinates operations between the helicopter, the navy ship, and the rescue boat. Each model element may include a graphical depiction and a brief description of the *OperationalContext* between the elements. The *OperationalContext* can represent abstract conceptual relationships and will be refined in subsequent diagrams.

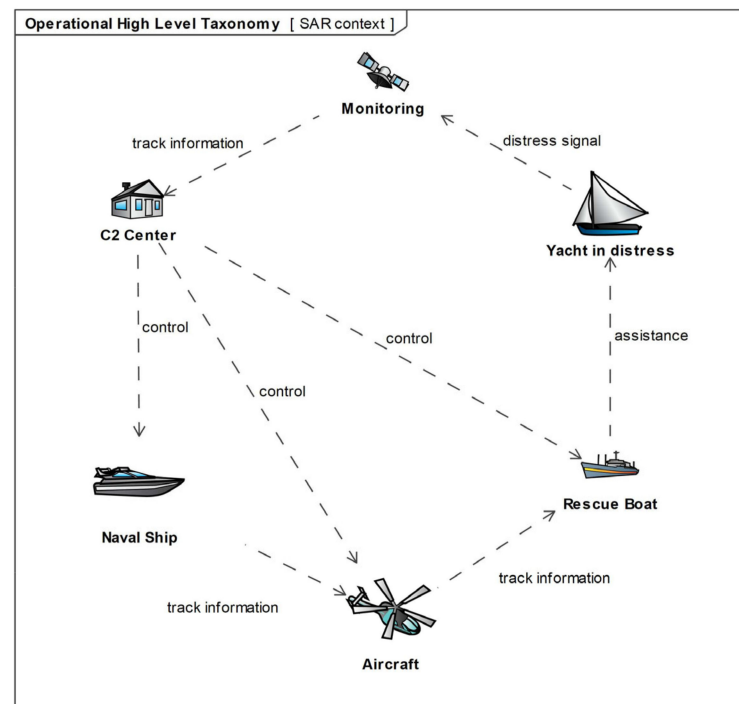


Figure 14. Operational Taxonomy (Op-Tx) of SoS context with graphics.

5.2. Step 2: Organize Operational Performers Based on Their Relationships

To enhance the understanding of various stakeholders, logical and physical information is included in the graphical elements at both ends of the *OperationalContext*. Therefore, complementary performers are created based on the *StructuralFactor* in the SoSCO to complete the logic further. Besides, to refine the *TemporalFactor* mentioned in the SoSCO, relevant performers can be added through the generalization relationship. The Operational Taxonomy (Op-Tx) view of the UAF, implemented by the SysML BDD, can be used to model the performers. For instance, Figure 15 illustrates the taxonomy of operational performers for the SAR phases. As the high-level operational concept diagram describes the primary context of the SAR, complementary performers should be created to enrich the context, e.g., a safe place represents what follow-up measures should be taken after victims are located. Besides, relevant performers such as “Searcher ph2” and “Searcher ph3” are added to refine the analysis of the search team from a time perspective. A searcher’s primary role is to locate and evaluate the status of missing persons or objects, while the primary focus of a rescuer is to deliver prompt aid and care to individuals in distress. Models with the stereotype of *OperationalArchitecture* can contain models whose stereotype is *OperationalPerformer*. The Operational Structure (Op-Sr) view of the UAF implemented by the SysML BDD can be used to model composition relationships to support a specific set of operational performers.

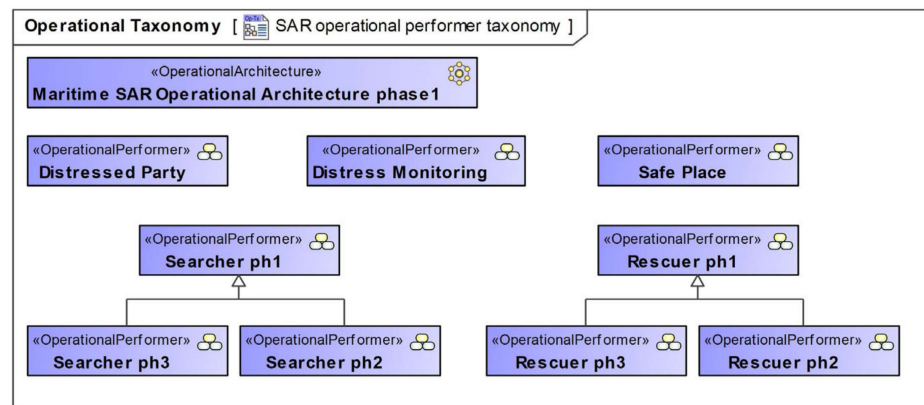


Figure 15. Operational Taxonomy (Op-Tx) of operational performers.

5.3. Step 3: Capture Structural Flows among Performers

To further refine *OperationalContext* logically, we propose using the Operational Connectivity (Op-Cn) view of the UAF implemented by the SysML IBD to specify the type and object name of the object passed on the flow. As the *OperationalContext* conveys objects without an explicit type, Figure 13 shows that the *OperationalConnector* defines the flow objects exchanged by the *OperationalPerformers* that generate and consume the stream objects. Additionally, *OperationalContext* is complemented by the generalization and composition of performers, with the result that additional *OperationalPerformers* on which the flows pass also should be captured through the *OperationalConnector*.

Figure 16 presents the Operational Connectivity (Op-Cn) perspective, which illustrates the stakeholders and their corresponding information exchange interactions involved in the SAR operation. This view builds on the various types of *OperationalPerformer* defined in the Op-Tx and Op-Sr views. These diagrams underscore the crucial role of information exchange and demonstrate the interactions among *OperationalPerformers*. For instance, the distressed party sends the distress signal as information to the search and rescue team. Other interactions can be exchanged between the *OperationalPerformers*, such as equipment and energy.

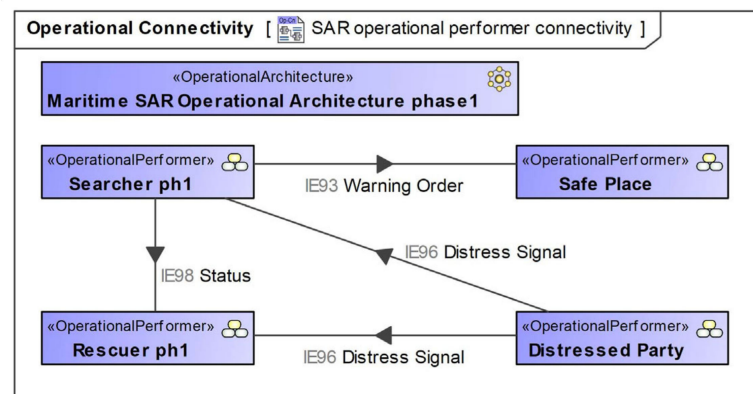


Figure 16. Operational Connectivity (Op-Cn) of operational performers.

The *OperationalMessage* represents the order of execution of the *OperationalExchange* in the operational interaction scenario, which is captured by the Operational Sequences (Op-Sq) view of the UAF implemented by the SysML sequence diagram (SEQ). The Op-Sq view defines time-based behavioral scenarios between operational elements. Figure 17 shows the sequence of interactions for the SAR operation depicted in the Op-Cn view. For example, as the Op-Cn view shows that the search team will transmit status to the rescue team and warning order to the safe place, the order in which these two behaviors can be known in the Op-Sq view.

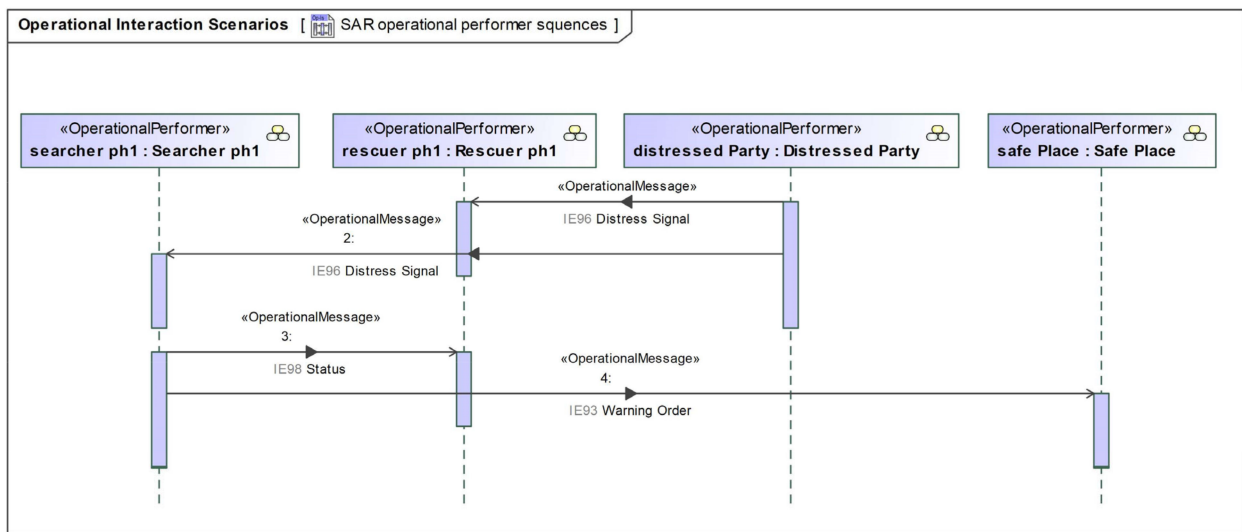


Figure 17. Operational Sequences (Op-Sq) of operational performers.

5.4. Step 4: Bridge Strategic and Operational Domains by Behavioral Elements

This step links strategic and operational domains by mapping between capabilities and activities. A capability is realized once the activity that utilizes it occurs. *ActivityComposition* represents that an activity can be composed of several lower-level activities to meet the required capabilities collectively. The Operational Processes (Op-Pr) view describes the operations generally conducted by the different operational performers. Figure 18 shows that a search activity corresponding to the search capability includes monitoring health and finding victims when a distress signal is received.

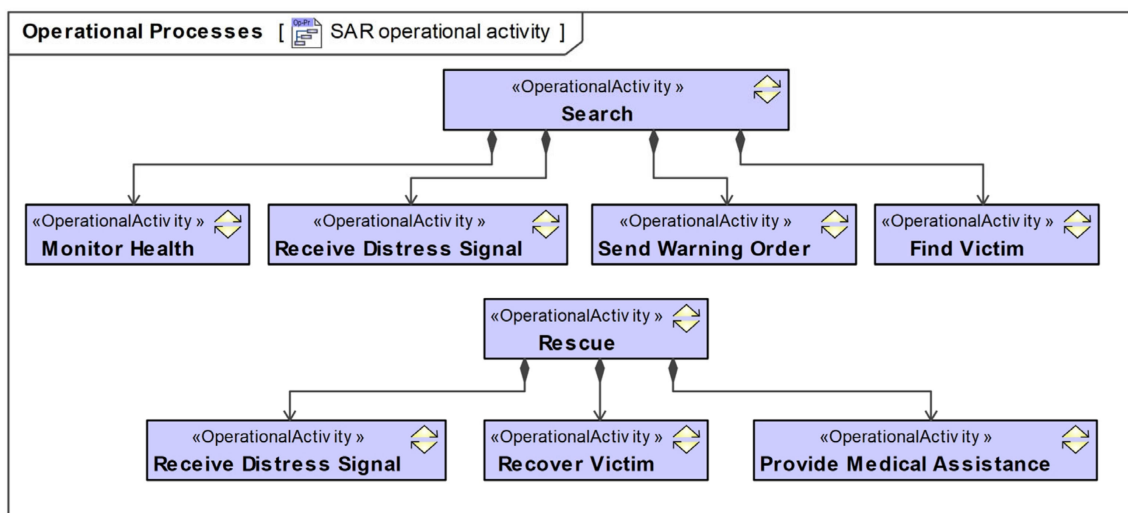


Figure 18. Operational Processes (Op-Pr) of operational activities.

5.5. Step 5: Capture Behavioral Flows among Activities Based on the Synthesis Method

This step addresses question 3 by describing the behavior of operational performers based on the *OperationalActivityEdge*. To achieve this, we propose a synthesis approach that involves the following steps:

1. Transforming performers from the Op-Sr view into the heads of partitions within the SysML activity diagram (ACT) to group all actions (usage of activities) performed by the same performers.

2. Transforming activities from the Op-Pr view into actions contained in the partitions of the ACT.
3. Representing operational connectors from the Op-Cn view by object flows (full lines) between the partitions in the ACT. Control flows (dashed lines) within each partition should be informed by domain knowledge.

The Operational Processes Flow (Op-Pr-Fl) view identifies performers based on the Op-Sr view, describes operational activities based on the Op-Pr view, and determines input/output flows between activities based on the Op-Cn view. Figure 19 illustrates the execution of the search activity, involving four performers engaged in a series of activities, from discovering victims to their placement.

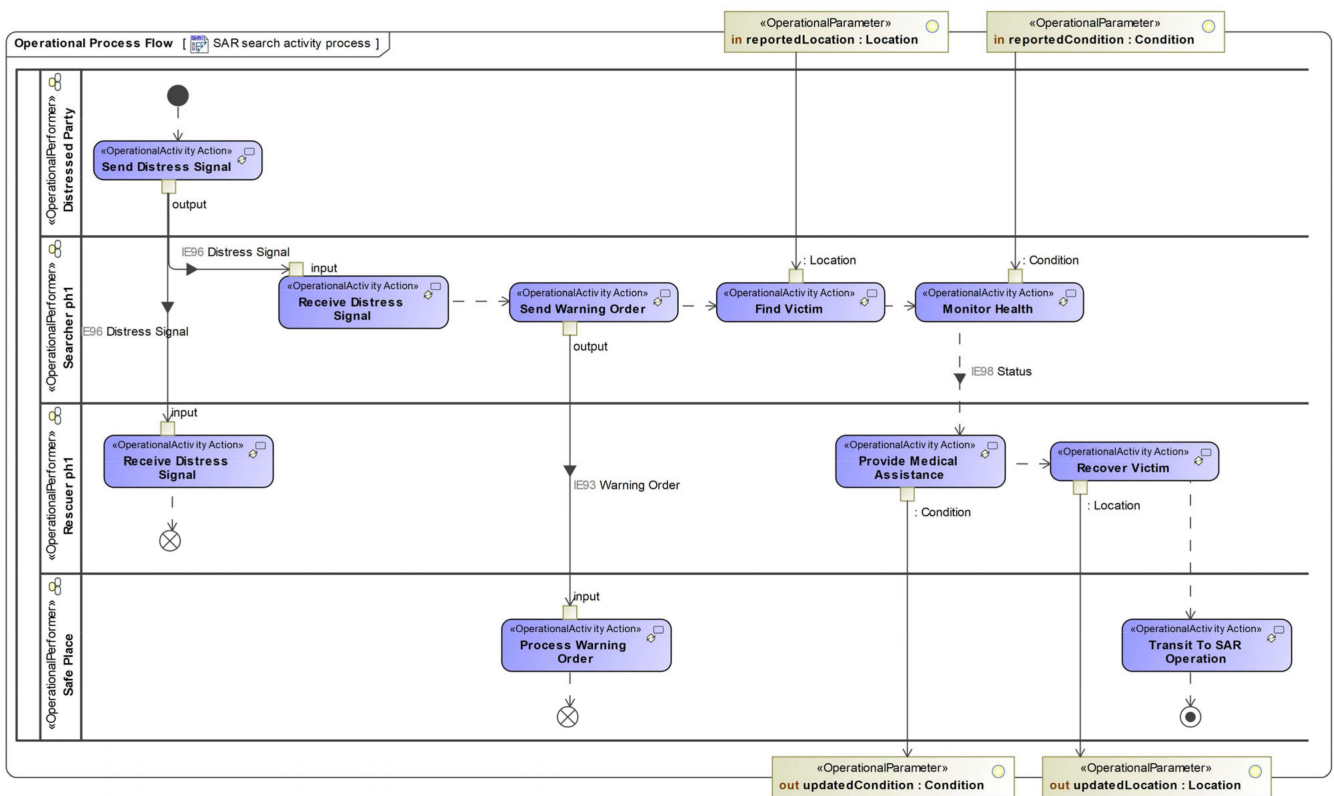


Figure 19. Operational Processes Flow (Op-Pr-Fl) of operational activities.

To capture the state-based behavior of operational performers, we propose using the Operational States (Op-St) view of the UAF implemented by the SysML state machine diagram (STM), as described in the SoSOO. This view enables us to model the operational states of *OperationalPerformer*, including the behaviors that occur in these states, the *Transitions* between states, and the events and guards that cause these transitions. Figure 20 provides an example of the Op-St view, depicting the operational states of the “Searcher ph1” in the Op-Pr-Fl view. In this example, the search node waits for a distress signal, and once received, the assignment is validated, and the searcher node transitions to searching for the victim.

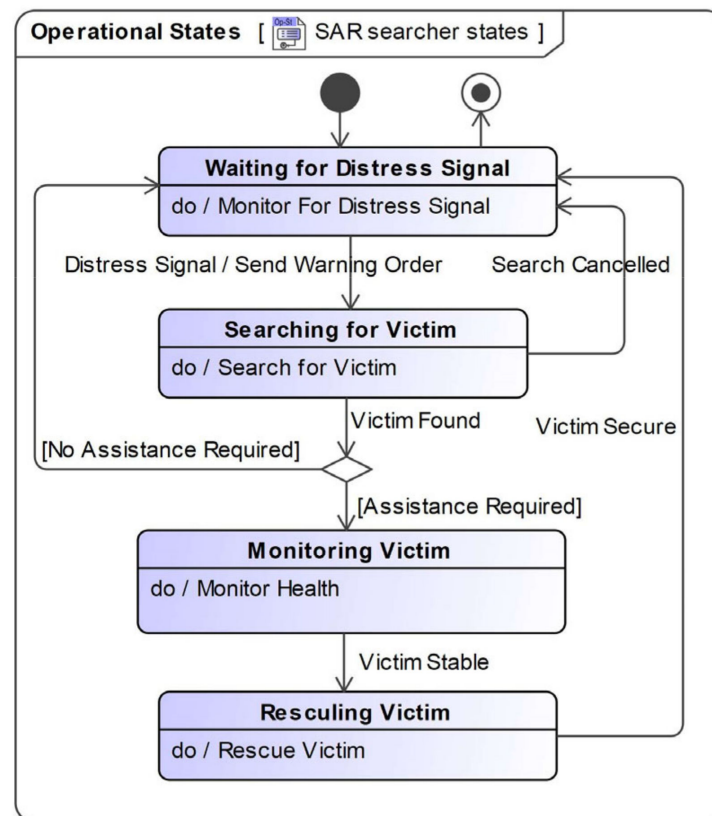


Figure 20. Operational States (Op-St) of operational performers.

6. Discussion

This study presents an ontology-based development approach to developing SoS architecture that covers strategic and operational domains. The approach provides a systematic presentation of multiple perspectives and construction procedures to assist in developing architecture models. Although the strategic domain is addressed first, it is closely inter-related with the operational domains of operations, resources, projects, and others. Thus, after analyzing the other domains, revisiting the strategic domain to revise traceability and conduct further analysis becomes necessary. For instance, insufficient domain knowledge in the strategic domain can be improved by constructing the operational architecture. The operational high-level concept diagram can help domain experts comprehend the mission decomposition. The traceability mechanism enables the appropriate implementation of capabilities and facilitates analyzing the effects of changes [43]. This study presents a series of architectural steps that enable traceability among various entities, including mission and capability, capability and operational role, and capability and operational activity. However, the traceability between assigned capability and corresponding CSs may require additional UAF domains, such as resource (Rs), criteria/standard (Sd), and actual resources (Ar), which could be considered in future work.

Compared with the ontology-based method for the breakdown of SoS missions [18], which uses the UAF at the model level before reasoning based on ontology, this study develops ontology at the metamodel level, thus providing more flexibility for the SoS modeling based on AFs. Ontologies can help to avoid ambiguities, facilitate information exchange between modules and external software, and provide a shared understanding of the design process [59]. In contrast to an ontology-based common conceptual reference used as a foundation for aircraft design processes [59], we employ the UAF metamodel to develop an ontology with greater detail on the SoS architecture. This enriched ontology extends the relationships and flows, facilitating the proposal of architecture steps. Unlike the semantic approach [60] that seeks to resolve regulatory ambiguity in aircraft design

and development through process mapping, Unified Modeling Language (UML), and ontological modeling, we present a set of specific architectural steps that offer simplicity and ease of use. The conceptual methodology for SoS development provides six steps to select the preferred SoS [61] but does not implement the modeling process by any architecture description language. This paper provides detailed steps based on ontology using the UAF views implemented by SysML. The flow-based functional analysis is applied at the system level [58], and the flow-based design process is expected to be used in individual standalone systems [55]. This study introduces architectural steps at the SoS-level, which can be upstream of the system design in SE.

The SoS modeling process in this paper is considered to develop SoS architecture based on AFs [13]. The development of architecture models involves selecting from more than 80 views supported by AFs and building them in a proper sequence. Therefore, the essential model elements in views such as mission, capability, and operational aspects should be explicitly described, and their organization and consistency should also be carefully considered. In particular, organizing elements meaningfully ensures explicit relationships between views within viewpoints. Better consistency among elements can improve communication and understanding among stakeholders. Therefore, relationships between views can be more explicit, and the modeling sequence can be more precise. Table 4 compares the proposed approach in this paper with three other studies regarding the descriptions of architectural elements and the consistency between them. It can be seen that other studies do not explicitly consider the connection between mission and capability. This paper proposes three factors to analyze and bridge mission and capability. While other methods describe capabilities and operations, the transition between them is not explicit. Flow-based representation and related modeling procedures are proposed in this paper to enrich the organization of the operational domain and strengthen the links between strategic and operational domains. In addition, some studies use implicit organizational relationships between views within viewpoints, which may result in a confusing modeling order. With the increase in the number of models, they may not be fully leveraged in subsequent models, resulting in decreased efficiency of architecture model development. To this end, this paper highlights the importance of architecture model development steps to avoid duplication of efforts and ensure efficient development.

Table 4. Comparison of view-based modeling approaches of SoS architecture.

	Mission and Role [39]	Break down SoS Needs [18]	Decision Patterns [13]	This Study
Description of Mission	+	o	o	+
Description of Capability	-	-	o	-
Organization of Capability	o	-	o	+
Description of Operational	+	+	+	+
Organization of Operational	-	o	-	+
Consistency between Mission and Capability	o	o	o	+
Consistency between Capability and Operational	o	-	-	+

Legends: + means: Abundant content, - means: Moderate content, o means: Implicit or no content.

However, there are still some limitations to the proposed approach. Firstly, the proposed process mainly focuses on strategic and operational architectures, neglecting the resources (physical) architecture, which is crucial for implementing the operational architecture and explaining the assignment of the operational roles to CSs. Additionally, this paper does not consider the measure of performance (MoP) and the measure of effect (MoE), which can be addressed using other UAF model kinds, such as the states, information, parameters, and constraints. Secondly, although the proposed modeling process reduces the gap between strategic and operational architecture, some elements of architecture, particularly those pertaining to operational architecture, are known a priori. It still requires domain experts to provide domain knowledge, which should be formally captured in the ontology base to support SoS design based on reasoning [18]. Finally, the proposed process

needs to be applied to more domains and case studies since the SAR case appears to be a directed SoS. In contrast, other SoS types, including acknowledged, collaborative, and virtual, exhibit higher degrees of autonomy of CSs, leading to a complexity that may be difficult to handle by the SysML-based modeling alone. Therefore, agent-based modeling may complement SysML-based modeling [62].

7. Conclusions

This study proposes an ontology-based modeling process for architecting SoS in the early design phase of the SoS to translate SoS missions into strategic and operational architectures through specific modeling steps based on the UAF views implemented by SysML. The novelty of this work lies in providing a novel perspective to model strategic and operational domains of AFs for an SoS architecture based on relationships and flows. The contributions of this study are summarized as follows:

1. The metamodel for the SoS is customized using the UAF DMM, which provides a comprehensive view of the mission from both structural and behavioral perspectives. Mapping rules are then proposed to enable the construction of an ontology from the metamodels, thereby enhancing the semantics for developing SoS architecture.
2. The proposed SoSCO and SoSOO rely on the relationships and flows between architectural elements, thus enabling a shared comprehension of the strategic and operational architectures of the SoS.
3. The detailed modeling steps of SoS strategic and operational architectures provide a structured approach to defining capabilities, bridging strategic and operational domains, and identifying the behavioral process of operational activities. These steps enable the selection and construction order of architecture views for AFs during early design.

Continued refinement and enhancement of the metamodel, ontologies, and modeling process can lead to more sophisticated approaches for designing SoSs.

Future efforts can focus on two areas: (1) Validation and evaluation through analysis of UAF model types (i.e., constraints and parameters) and domains (i.e., criteria/standard and actual resources); (2) Establishment of a comprehensive knowledge base to support SoS architecture design, such as knowledge-based reasoning for automating the matching process from operational architecture to resource architecture.

Author Contributions: Conceptualization, Y.F., Q.Z. and C.Z.; methodology, Y.F., Y.L. and Q.P.; writing original draft preparation, Y.F. and Q.Z.; writing—review and editing, C.Z., Q.P. and Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China (grant number 2018YFB1700901); the National Science Foundation of China (grant numbers 61873236 and 62102355); the fund from the Fifth Electronic Research Institute of Ministry of Industry and Information Technology (grant number HK07202200877); and the Civil Aerospace Technology Pre-Research Program (grant number D020101).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Maier, M.W. Architecting principles of systems-of-systems. In Proceedings of the 6th International Symposium, Chiang Mai, Thailand, 14–17 October 1996.
2. Nielsen, C.B.; Larsen, P.G.; Fitzgerald, J.; Woodcock, J.; Peleska, J. Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions. *ACM Comput. Surv.* **2015**, *48*, 1–41. [[CrossRef](#)]
3. OUSD (AT&L). *Systems Engineering Guide for Systems of Systems*; Pentagon: Washington, DC, USA, 2008.

4. DeLaurentis, D.A.; Crossley, W.A.; Mane, M. Taxonomy to guide systems-of-systems decision-making in air transportation problems. *J. Aircr.* **2011**, *48*, 760–770. [CrossRef]
5. Okami, S.; Kohtake, N. Transitional complexity of health information system of systems: Managing by the engineering systems multiple-domain modeling approach. *IEEE Syst. J.* **2017**, *13*, 952–963. [CrossRef]
6. Fang, Z. System-of-Systems Architecture Selection: A Survey of Issues, Methods, and Opportunities. *IEEE Syst. J.* **2021**, *16*, 4768–4779. [CrossRef]
7. *ISO/IEC/IEEE 42010; Systems and Software Engineering—Architecture Description*. ISO/IEC/IEEE: Geneva, Switzerland, 2011; Volume 2011, pp. 1–46.
8. U.S. Department of Defense. DoD Architecture Framework (DoDAF) ver 2.02. Available online: <https://dodcio.defense.gov/Library/DoD-Architecture-Framework/> (accessed on 17 February 2023).
9. DeLaurentis, D.; Raz, A.; Guariniello, C. MBSE for System-of-Systems. In *Handbook of Model-Based Systems Engineering*; Madni, A.M., Augustine, N., Sievers, M., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 1–29.
10. Shirvani, F.; Beydoun, G.; Perez, P.; Scott, W.; Campbell, P. An Architecture Framework Approach for Complex Transport Projects. *Inf. Syst. Front.* **2021**, *23*, 575–595. [CrossRef]
11. INCOSE. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*; INCOSE: San Diego, CA, USA, 2015.
12. Lock, R.; Sommerville, I. Modelling and analysis of socio-technical system of systems. In Proceedings of the 2010 15th IEEE International Conference on Engineering of Complex Computer Systems, Oxford, UK, 22–26 March 2010; pp. 224–232.
13. Fang, Z.; Jin, W. Exploring Decision Patterns for Supporting DoDAF based Architecture Design. In Proceedings of the 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Prague, Czech Republic, 9–12 October 2022; pp. 2993–2999.
14. OMG. Systems Modeling Language Specification Version 1.6. Available online: <https://www.omg.org/spec/SysML/1.6> (accessed on 22 April 2023).
15. Fitzgerald, J.; Larsen, P.G.; Woodcock, J. *Modelling and Analysis Technology for Systems of Systems Engineering: Research Challenges*; INCOSE: San Diego, CA, USA, 2012.
16. Object Management Group. Unified Architecture Framework (UAF) Domain Metamodel v1.1. Available online: <https://www.omg.org/spec/UAF/1.1/DMM/PDF> (accessed on 17 February 2023).
17. Wardhana, H.; Ashari, A.; Sari, A.K. Transformation of SysML Requirement Diagram into OWL Ontologies. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 106–114. [CrossRef]
18. Knöös Franzén, L.; Staack, I.; Krus, P.; Jouannet, C.; Amadori, K. A Breakdown of System of Systems Needs Using Architecture Frameworks, Ontologies and Description Logic Reasoning. *Aerospace* **2021**, *8*, 118. [CrossRef]
19. Lin, J.; Fox, M.S.; Bilgic, T. A requirement ontology for engineering design. *Concurr. Eng. Res. Appl.* **1996**, *4*, 279–291. [CrossRef]
20. Bermudez-Edo, M.; Elsaleh, T.; Barnaghi, P.; Taylor, K. IoT-Lite: A lightweight semantic model for the Internet of Things. In Proceedings of the 2016 INTL IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (Uic/Atc/Scalcom/Cbdcom/Iop/Smartworld), Toulouse, France, 18–21 July 2016; pp. 90–97.
21. Sales, D.C.; Becker, L.B.; Koliver, C. Informatics. The systems architecture ontology (SAO): An ontology-based design method for cyber-physical systems. *Appl. Comput. Inf.* **2022**. ahead of print. [CrossRef]
22. Baek, Y.-M.; Song, J.; Shin, Y.-J.; Park, S.; Bae, D.-H. A meta-model for representing system-of-systems ontologies. In Proceedings of the 6th International Workshop on Software Engineering for Systems-of-Systems, Gothenburg, Sweden, 29 May 2018; pp. 1–7.
23. Zachman, J.A. A framework for information systems architecture. *IBM Syst. J.* **1987**, *26*, 276–292. [CrossRef]
24. The Open Group. TOGAF-The Open Group Architecture Framework Version 9.2. Available online: <https://www.opengroup.org/togaf> (accessed on 28 March 2023).
25. Ministry of Defence. Ministry of Defence. Ministry of Defense Architecture Framework. Available online: <https://www.gov.uk/mod-architecture-framework> (accessed on 28 March 2023).
26. North Atlantic Treaty Organization. NATO Architecture Framework Version 4. Available online: https://www.nato.int/cps/en/natohq/topics_157575.htm (accessed on 28 March 2023).
27. Odukoya, K.A.; Whitfield, R.I.; Hay, L.; Harrison, N.; Robb, M. An Architectural Description For The Application Of Mbse In Complex Systems. In Proceedings of the 2021 IEEE International Symposium on Systems Engineering (ISSE), Virtual Conference, 13 September–13 October 2021; pp. 1–8.
28. Object Management Group. Unified Architecture Framework (UAF) Version 1.0. Available online: <https://www.omg.org/spec/UAF/1.0/> (accessed on 17 February 2023).
29. Object Management Group. Unified Architecture Framework (UAF) Version 1.2. Available online: <https://www.omg.org/spec/UAF/1.2/Beta1/About-UAF/> (accessed on 17 February 2023).
30. Object Management Group. Enterprise Architecture Guide for UAF. Available online: <https://www.omg.org/cgi-bin/doc?dtc/21-12-13.pdf> (accessed on 17 February 2023).
31. Dandashi, F.; Hause, M.C. UAF for system of systems modeling. In Proceedings of the 2015 10th System of Systems Engineering Conference (SoSE), San Antonio, TX, USA, 17–20 May 2015; pp. 199–204.

32. Eichmann, O.C.; Melzer, S.; God, R. Model-based Development of a System of Systems Using Unified Architecture Framework (UAF): A Case Study. In Proceedings of the 2019 IEEE International Systems Conference (SysCon), Orlando, FL, USA, 8–11 April 2019; pp. 1–8.
33. Zhang, Y.; Xing, C.; Wang, Q.; Zhu, J. A UAF Based Method for Next Generation Air Traffic Management System Development. In *Advances in Guidance, Navigation and Control, Proceedings of the 2020 International Conference on Guidance, Navigation and Control, ICGNC 2020, Tianjin, China, 23–25 October 2020*; Springer: Singapore, 2022; pp. 4053–4063.
34. Mori, M.; Ceccarelli, A.; Lollini, P.; Frömel, B.; Brancati, F.; Bondavalli, A. Systems-of-systems modeling using a comprehensive viewpoint-based SysML profile. *J. Softw. Evol. Process* **2017**, *30*, e1878. [\[CrossRef\]](#)
35. Browning, C.M.; Deal, J.; Mayes, S.; Arshad, A.; Rich, T.C.; Leavesley, S.J. Excitation-scanning hyperspectral video endoscopy: Enhancing the light at the end of the tunnel. *Biomed. Opt. Express* **2021**, *12*, 247–271. [\[CrossRef\]](#) [\[PubMed\]](#)
36. Graves, H. Integrating SysML and OWL. In Proceedings of the OWL: Experiences Directions, Chantilly, VA, USA, 23–24 October 2009.
37. Zhu, W.; He, H.; Wang, Z. Ontology-Based Mission Modeling and Analysis for System of Systems. In Proceedings of the 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Exeter, UK, 21–23 June 2017; pp. 538–544.
38. Friedenthal, S.; Moore, A.; Steiner, R. *A Practical Guide to SysML: The Systems Modeling Language*; Morgan Kaufmann: Burlington, MA, USA, 2014.
39. Cherfa, I.; Belloir, N.; Sadou, S.; Fleurquin, R.; Bennouar, D. Systems of systems: From mission definition to architecture description. *Syst. Eng.* **2019**, *22*, 437–454. [\[CrossRef\]](#)
40. Koutsopoulos, G.; Henkel, M.; Stirna, J. An analysis of capability meta-models for expressing dynamic business transformation. *Softw. Syst. Model.* **2021**, *20*, 147–174. [\[CrossRef\]](#)
41. Jamshidi, M. *System of Systems Engineering: Innovations for the 21st Century*; John Wiley & Sons: Hoboken, NJ, USA, 2008; pp. 1–591.
42. Sage, A.P.; Rouse, W.B. *Handbook of Systems Engineering and Management*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
43. Morkevicius, A.; Bisikirskiene, L.; Bleakley, G. Using a systems of systems modeling approach for developing Industrial Internet of Things applications. In Proceedings of the 2017 12th System of Systems Engineering Conference (SoSE), Waikoloa, HI, USA, 18–21 June 2017; pp. 1–6.
44. IDEAS Group. International Defence Enterprise Architecture Specification. Available online: https://en.wikipedia.org/wiki/IDEAS_Group (accessed on 28 March 2023).
45. Musen, M.A. The protégé project: A look back and a look forward. *AI Matters* **2015**, *1*, 4–12. [\[CrossRef\]](#)
46. Maier, M.W. Architecting principles for systems-of-systems. *Syst. Eng.* **1998**, *1*, 267–284. [\[CrossRef\]](#)
47. Tirone, L.; Guidolotti, E.; Fornaro, L. A Tailoring of the Unified Architecture Framework’s Meta-Model for the Modeling of Systems-of-Systems. *Proc. INCOSE Int. Symp.* **2018**, *28*, 1691–1705. [\[CrossRef\]](#)
48. Hongyue, H.; Weixing, Z.; Ruiyang, L.; Qiaoyu, D. An executable modeling and analyzing approach to C4ISR architecture. *J. Syst. Eng. Electron.* **2020**, *31*, 109–117.
49. Dridi, C.E.; Benzadri, Z.; Belala, F. System of Systems Engineering: Meta-Modelling Perspective. In Proceedings of the 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE), Budapest, Hungary, 2–4 June 2020; pp. 000135–000144.
50. Qi, Y.; Wang, Z.; Dong, Q.; He, H. Modeling and verifying SoS performance requirements of C4ISR systems. *J. Syst. Eng. Electron.* **2015**, *26*, 754–763.
51. Franzén, L.K.; Schön, S.; Papageorgiou, A.; Staack, I.; Ölvander, J.; Krus, P.; Amadori, K.; Jouannet, C. A System of Systems Approach for Search and Rescue Missions. In Proceedings of the AIAA Scitech 2020 Forum, Orlando, FL, USA, 6–10 January 2020.
52. Gao, Y.; Liu, H.; Niu, F.; Tian, Y.; Wang, J.; Cheng, W. Search and rescue system-of-systems influence degree evaluation of aviation equipment based on simulation. *Sci. Rep.* **2022**, *12*, 22384. [\[CrossRef\]](#)
53. Cao, Y.; Liu, Y.; Paredis, C.J. System-level model integration of design and simulation for mechatronic systems based on SysML. *Mechatronics* **2011**, *21*, 1063–1075. [\[CrossRef\]](#)
54. Horrocks, I.; Patel-Schneider, P.F.; Boley, H.; Tabet, S.; Grosz, B.; Dean, M. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Memb. Submiss.* **2004**, *21*, 1–31.
55. Cao, Y.; Liu, Y.; Ye, X.; Zhao, J. An Automated Approach for Execution Sequence-Driven Software and Physical Co-Design of Mechatronic Systems Based on Hybrid Functional Ontology. *Comput.-Aided Des.* **2021**, *131*, 102942. [\[CrossRef\]](#)
56. Sirin, E.; Parsia, B.; Grau, B.C.; Kalyanpur, A.; Katz, Y. Pellet: A practical OWL-DL reasoner. *J. Web Semant.* **2007**, *5*, 51–53. [\[CrossRef\]](#)
57. Torkjazi, M.; Davila-Andino, A.J.; Alghamdi, A.; Zaidi, A.K. UAF Strategic Planning for Enterprises. *IEEE Access* **2022**, *10*, 123549–123559. [\[CrossRef\]](#)
58. Yuan, L.; Liu, Y.; Sun, Z.; Cao, Y.; Qamar, A. A hybrid approach for the automation of functional decomposition in conceptual design. *J. Eng. Des.* **2016**, *27*, 333–360. [\[CrossRef\]](#)
59. Gómez-Rodríguez, Á.; Poveda-Villalón, M.; García-Castro, R.; Gómez-Pérez, A.; Cuerno-Rejado, C. Towards the Use of Ontologies in Remotely Piloted Aircraft Systems Conceptual Design: Opportunities and Challenges. In Proceedings of the AIAA Scitech 2021 Forum, Virtual Event, 11–15 and 19–21 January 2021.

60. Cartile, A.; Marsden, C.; Liscouet-Hanke, S. Demonstrating a semantic approach to clarifying regulatory ambiguity in aircraft design and development using process mapping, UML, and ontological modeling. In Proceedings of the AIAA SCITECH 2023 Forum, National Harbor, MD, USA, 23–27 January 2023.
61. Mokhtarpour, B.; Stracener, J. A Conceptual Methodology for Selecting the Preferred System of Systems. *IEEE Syst. J.* **2017**, *11*, 1928–1934. [[CrossRef](#)]
62. Baldwin, W.C.; Sauser, B.; Cloutier, R.J.P.C.S. Simulation approaches for system of systems: Events-based versus agent based modeling. *Procedia Comput. Sci.* **2015**, *44*, 363–372. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.