



**HAL**  
open science

# A Unified Architecture Framework supporting SoSs Development: Case of the Aircraft Emergency Response System-of-Systems

Charaf Eddine Dridi, Zakaria Benzadri, Faiza Belala

## ► To cite this version:

Charaf Eddine Dridi, Zakaria Benzadri, Faiza Belala. A Unified Architecture Framework supporting SoSs Development: Case of the Aircraft Emergency Response System-of-Systems. International Journal of Organizational and Collective Intelligence (IJOICI), 2023. hal-04227367

**HAL Id: hal-04227367**

**<https://hal.science/hal-04227367>**

Submitted on 3 Oct 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A Unified Architecture Framework supporting SoSs Development: Case of the Aircraft Emergency Response System-of-Systems

Charaf Eddine Dridi<sup>1</sup>[0000-0001-5724-8187], Zakaria Benzadri<sup>1</sup>[0000-0002-9199-0657], and Faiza Belala<sup>1</sup>[0000-0002-4563-4061]

Constantine2 - Abdelhamid Mehri University LIRE Laboratory, Constantine, Algeria  
{charafeddine.dridi, benzadri.zakaria, faiza.belala}@univ-constantine2.dz

**Abstract.** One of the major challenges in large-scale systems is the effective architectural design of complex constituents employing rigorous modeling methodologies. The engineering of Systems-of-Systems (SoSs) constitutes another critical issue that requires the definition of multiple viewpoints that are dedicated to various concerns of stakeholders. To address this challenge, this paper contributes to the definition of a reusable framework handling the design of SoSs' architectures by adopting a conceptual model of architecture framework "ISO/IEC 42010:2011 standard for Systems and Software Engineering-Architecture Description". The proposed framework extends this standard by using well-defined software development processes, to identify and implement the different architectural viewpoints. Besides, these processes are used in a way to take advantage to manage and document a set of diagrams given by a UML profile, and then, to verify that the parts of the architecture form a consistent whole. In this context, we define four main viewpoints dedicated to the various stakeholders and which are essential to allow them to describe and implement different SoSs. To guide the control and coordination of the development tasks, this framework provides again a development processes model that allows the stakeholders to explicitly design the viewpoint they want using an SoS-UML profile. This latter is a design tool that implements these viewpoints in a modeling environment together with the corresponding dedicated development processes. The results we get on the concrete design of the case study we considered are encouraging and go beyond the results of existing frameworks.

**Keywords:** SoS · SoSE process · Multi-Viewpoints Architecture Framework · SoS-UML Profile · IEEE 42010

## 1 Introduction

In recent years, SoSs have experienced an increasing evolution and interest from the computer science community. SoSs are not designed in a top-down way, they have been designed to integrate multiple independent functional systems into a larger system in several important application domains. However, these

Constituent-Systems (CSs) are becoming larger, more complex, and difficult to develop as well. Examples of these large-scale systems can be found in several fields such as Robotics, Avionics, Military, Intelligent systems (Smart-Grids, Smart-Cities, Smart-Homes, etc.). An SoS is characterized by offering new functionalities to users that cannot be offered by its CSs, but emerging from their combination. The CSs making up an SoS are independent, geographically distributed, developed with different technologies and intended for several platforms.

As SoSs get more and more complex, engineers need to pay a lot of attention to early-stage design decisions rather than focusing on implementation and writing code. This will facilitate the process of developing such large-scale systems. Additionally, Architecture Frameworks are a recent discipline in Software Engineering (SE) that consider Architectural Viewpoints as first-class entities in software development. The viewpoints have become the major paradigm in which the SE will be able to open a door to the development representation and provide a new way of designing systems. In the context of SoSs, we argue that our architecture framework encompassing the knowledge on how to design SoSs would ease the application of this process and, consequently, produce an SoS of higher quality. A single comprehensive viewpoint of an SoS' architecture is often too complex to be understood and communicated in its most detailed form, showing all the relationships between the various business, structural and behavioral aspects. Therefore, we are seeking to represent by means of one or more architecture viewpoints that together can provide a unified AF of SoS' architecture.

## 1.1 Context and Problematic

The field of SoSs comes up against constraints during the engineering process. The difficulty of modeling SoSs lies in the complexity resulting from the interaction, cooperation and collaboration of their heterogeneous CSs, which each have specific goals to accomplish, different roles to play, and they are not easily interoperable. Independent evolution and dynamic changes can cause these CSs to behave differently. These changes can affect their interactions and communications within the SoS and consequently, it can derail the overall mission of the SoS. Architecting an SoS helps to understand how it works, as well as to master its complexity before its implementation. This offers considerable advantages to the designers of such systems. Besides, it can be seen as the separation of different functional and non-functional needs which are related to SoSs characteristics, SoSs quality attributes, management and oversight, SoSs architecture, design and implementation.

In this new perspective, SoSs development does not follow the normal system development process. As SoSs' capabilities are based on the contributions of the individual CSs, their interdependences make a document-centric development impractical as an exorbitant effort. The development processes refer to activities that can guide an SoS' lifecycles from the system requirements level down to

the software implementation level, and naturally, by coordinating the various processes for the development of a new system [1].

From a SE perspective, design decisions made at the architectural level have a direct impact on the fulfillment of functional and quality requirements of SoSs development. At this stage, the SoS' Stakeholders identify functional and non-functional characteristics through the use of their own theoretical backgrounds, notations and environments. In addition, the SoSs' architectures are still created without the support of a systematic processes and traditional design approaches do not adequately support the creation of these types of systems due to their composed nature, their large-scale, their decentralized control mechanism, their evolving environments, and their large number of stakeholders [1].

## 1.2 Motivation and Objectives

To get a handle on this complexity, it is necessary to maintain consistency and coherence between the different viewpoints of different stakeholders, as well as the ability to reconcile and include all their viewpoints before proceeding to the various processes in the SoSE lifecycle. In this context, the paper introduces a Multi-viewpoints approach-based Architecture Framework, that is associated with a set of SoSE development processes and a UML profile dedicated to the SoSs that can facilitate and improve the design of SoSs' architectures. The purpose of this work is to present an AF that intends to foster the systematic development of SoSs' architectures. The main contributions of this paper are manifold:

- The proposal of an AF that encompasses the concepts proposed by “ISO/IEC/IEEE 42010:2011 Systems and Software Engineering-Architecture description” to manage the complexity of SoSs' architectures following a multi-viewpoint approach. However, the standard is a general Meta-Model, therefore we have had to specialize according to SoSE's context. Consequently, our specialization aims at specifying a set of SoSE processes necessary to have a semantic consistency between the different parts of the SoS' architecture specified in the different viewpoints.
- Improving the potential of the AF by integrating systematic processes of developing SoSs' architectures which can have a positive impact on the overall quality of the framework. The design of these processes is based on a consolidated Model-Based SoSE in which the involved processes can be seen as a sequence of connected and dependent activities. It should be noted that to our knowledge, there is no available SoSE process in the literature to support the design of IEEE 42010 standard.
- Defining an SoS-UML Profile to model a set of SoSs' processes. The advantage of this profile is to provide a large number of models to separately capture, describe and organize each of the processes of different viewpoints. The proposed profile is a package of new stereotypes' notations that are extended from existing UML2.0 elements. These new notations will complete the list of standard UML notations by modeling explicitly and appropriately all structural and behavioral aspects of SoSs.

- Demonstration of the work by designing an SoSs’ architecture of an “Aircraft Emergency Response System-of-Systems” (AERSoS). It is therefore the aim of the paper to show that a unified AF for SoSs’ architectures abstractions are needed.

### 1.3 Paper structure

The remainder of this paper is organized as follows: in Section 2, we provide some prerequisites and backgrounds on SoSs and our related works. In section 3, we discuss different approaches adopted to design and analyze these systems. Section 4 gives a top view model of our approach. In Section 5, we show the detailed description of the Multi-viewpoint-based architecture framework. In Section 6, we introduce a UML extension mechanism for modeling the SoS’s architectures, conducted by an illustrative case study. Finally, Section 6 concludes the paper and discusses possible future works.

## 2 Prerequisites

SoSs are an emerging vision for the next-generation systems that are built by interconnecting existing legacy systems. The existing literature provides a rich set of definitions of an SoS. However, there is no an official agreed definition for what is an SoS or what are its main characteristics. In general, the definition of an SoS is mainly based on the properties and application domains. Based on existing works [2–6], we define an SoS as a set of distributed and complex CSs that interact in a network structure to create a large-scale system and perform a unified capability that cannot be provided by any of the CSs.

Moreover, the authors of [2, 5] and others, highlight the five common characteristics which are: (1) operational independence of CSs, (2) managerial independence of CSs, (3) geographic distribution, (4) emergent behavior and (5) evolutionary development. These characteristics are the main distinguishing properties between SoSs and other types of systems. A system that does not have these characteristics cannot be considered an SoS [1].

In addition, an SoS can take four different types. These types are primarily based on the governance, management complexity and the relationships among the CSs in the SoS. Therefore, every SoS can be recognized, treated and classified following one of the following four types of SoSs: Virtual, Collaborative, Acknowledged or Directed.

On the other hand, there is no doubt that today’s SoSs can be found everywhere, and it is easy to see that their applications are increasingly covering a variety of domains. A wide range of studies has already addressed these domains: Transportation [7–10], Healthcare [11, 8], Military Defense [12, 13], Smart City [15, 14], Smart Energy Grids [8, 14], Emergency Management and Response [8]. For more details about SoS’ characteristics, types and application domains, see [16, 1].

To validate our approach, an example of Aircraft Emergency Response System-of-Systems, abbreviated AERSoS, will be considered in order to clarify and concretize the basic elements and the contributions made by SoS-AF methodology. This case study was illustrated in [16] to introduce the notion of Meta-Modeling in the SoSs context. We will adopt it here to encapsulate all the necessary notions in the SoS-AF.

As an illustrative example of an SoS, consider a collection of autonomous and interacted CSs tasked with the prevention of the aircraft from any accident or system failures. To achieve this Goal, *AERSoS* must be designed in such a way that the CSs can interact and perform a unique capability that cannot be provided by any of the CSs. Examples of the *AERSoS*' CSs include an *AircraftEmergenciesSoS*, *EmergencyResponseSoS*, *WarningSoS*, *EvacuationCS*, *TowerControlCS*, *EnginesProtectionCS*, *LandingCS*, etc. see Figure 1.

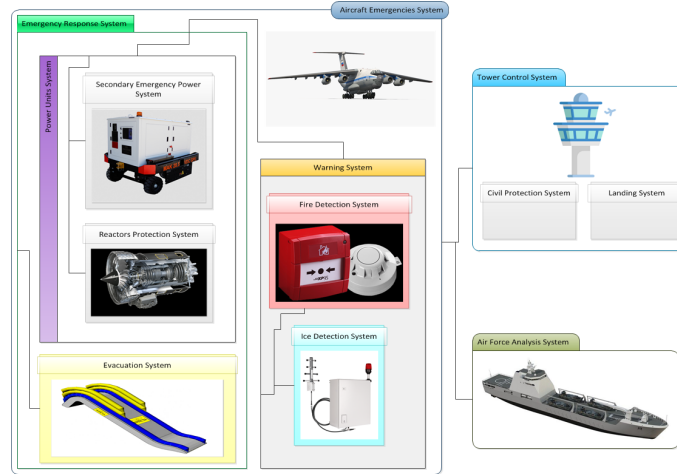
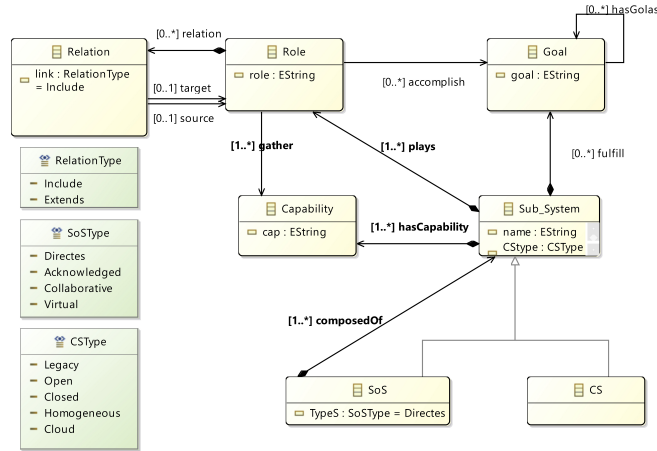


Fig. 1. Aircraft Emergency Response System-of-Systems (*AERSoS*) from [16]).

Each CS of the AERSoS is a system that is specified by a set of entities, which are divided into three types. The first set represents the Roles describing the ideal behavior of CSs through the gathering of the required Capabilities to accomplish the AERSoS global-Goals, the second represents the Capabilities describing the functions provided by each CS in specific Roles to the wider needs of the AERSoS, and finally, the Goals, describing instances of the AERSoS' Roles, that represents sub-Goals of each CS and Global-Goal of AERSoS.

In our previous work[1], we have introduced a Meta-Model called *MeM-SoS*(see Figure 2). We have presented the main notions, concepts and entities constituting an SoS. *MeMSoS* emphasizes a vision-oriented on the hierarchical composition of CSs and highlights the importance of *CS*, *Roles*, *Capabilities*, *Goals* and *Interactions*, etc. (for further details, readers are encouraged to refer

to [16] and [1]). The *MeMSoS* is based on the EMF “Eclipse Modelling Framework”, we used Ecore Modelling Language to specify the abstract syntax of the Meta-Model and carried out within Eclipse Sirius tool.



**Fig. 2.** Overview of the *MeMSoS*, from [16]).

In addition, *MeMSoS* supports a Platform-Independent Model (PIM) level. Therefore, it allows the designers to successively refine the PIM in terms of SoS aspects. In this work, we consider the *MeMSoS* as our reference of concepts since it specifies the necessary concepts for SoSs description, identifies relations between their CSs and defines vocabulary to be mapped to SoS-UML Profile.

### 3 Related Works

Nowadays, there is a growing consensus that the SoS concept is an effective solution to implement and analyze large, complex, autonomous and heterogeneous distributed systems named CS. Systems of systems are collectively interacting to achieve a higher global target that would not be possible to accomplish individually. Their complexity arises from the uncertainty of the behavior: (1) to preserve their performance, (2) to complete the required functions and (3) most importantly to anticipate potential defects.

Actually, SoS engineering (SoSE) has been considered by many researchers and practitioners a fundamental asset in SoS development and deployment, since their applications included numerous modern society solutions. SoSE standardization represents the process of designing, organizing, deploying, and integrating the capabilities of a mix of existing and complex systems to produce desirable results into an SoS. A large number of modeling approaches that address SoS

architectural aspects were identified by recent works and the seven main classes are [1]: MDA, MD, SOA, Ontology, ADL, Bigraph and Hybrid. The following presented works in this section describe a review of the contributions in SoS, SoSE and related subjects. Most of these points are collected from [1]:

- **(Model-Driven Architecture) MDA-based approaches:** the aim of [21] was to define an abstract view with all the possible information in the configuration and deployment processes. A meta-model that represents several possible configurations was also produced. The authors of [22] have adopted an MDE approach to define a DSML that was used to model SoS security architectures. The authors of [23] have defined an SoS profile that extends on the SysML reference meta-model with specific language constructs. They have also introduced an extension of this work in [24]. In the same direction, in our previous paper [1], we have provided an MDA method that simplifies SoSs complexity by increasing their abstraction level.
- **(Model-Driven) MD-based approaches:** authors of [25] have proposed a formalism for relating basic SoS concepts by means of a UML class diagram. They have identified a set of basic concepts to describe a modelling approach for collaborative SoSs. The goal of [26] was to show how SysML models can be used to support a set of needs that are essential for a SoS. In [27], the authors have investigated through a case study in the construction domain the interplay between SoS and CS architectures. The paper [28] has provided an approach to support design activities in the SoS development process.
- **(Services-Oriented Architecture) SOA-based approaches:** the authors of [29] have proposed an approach to assist the SE community during the integration among CSs of a SoS and to use it as a basis for the composition of Directed SoS. The authors of [30] have proposed a service-based architecture, which they named MV-SoSA, that serves as a basis when composing new Mixed-type SoSs. The authors of [31] have realized a modular reconfigurable SoS based on a platform of reusable distributed CSs integrated within a SOA.
- **Ontology-based approaches:** the authors of [32], have described a SE methodology using a UML-like representation of SoS. UML has assisted the authors to develop the required elements of SoS ontologies. The aim of this paper [33], was to provide a method for approaching the first two levels “Needs and boundary conditions” and “SoS Capabilities” of the SoS-process and generating a SoS design space using ontology. The authors [34] have proposed an approach to build an SoS conceptual model and a foundational ontology adapted from DOLCE to depict SoS interoperability context [35]. The authors of [36] have proposed an SoS cyber effects ontology that outlines the requirements for a series of ontologies necessary to model the SoS effects of cyber-attacks.
- **(Architecture Description Language) ADL-based approaches:** this approach [37] suggests a Maude-based formal and executable model where communications and relationships architecture is well-defined. The authors of [38] have presented SosADL, an ADL based on a  $\lambda$ -Calculus with Concur-



rent Constraints specially designed for describing SoS architectures. The extended work [39] enables the description of evolutionary architectures, which maintain emergent behavior supporting dynamic reconfigurations. And in [40], they have focused on the description of SoS architecture to support automated verification.

- **Bigraph-based approaches:** in [41], the authors have proposed a novel methodology based on the formal technique of BRS with an inspiring vision from multi-scale modeling. The authors of [42] and [43] have demonstrated how bigraph-based approaches can engage with SoS through abstract relationships that allow for dynamic interaction. In [44], the authors have presented a tool for bigraph matching and transformation. They have implemented a solution based on an investigation of formal approach reaction rules that have been used to rewrite bigraphs for modeling and simulation of SoS.
- **Hybrid approaches:** in [45] where the authors have exploited different models and in particular executable models from SysML specifications. This work [46] has focused on developing a conceptual meta-model called M2SoS that represents SoS ontologies. The authors of [47] have presented a hybrid modeling method based on service-oriented and ontology-based modeling. The authors of [48], have presented an MDA for service-oriented SoS architecting, modeling and simulation. The authors of [49], have used a hybrid approach with both Colored Petri Nets and Object Process Method modeling languages to create executable architecture models for SoSs.

In the same context [50], the authors have proposed a method to adapt the standard proposed by "ISO 42010 for the software industry" to be adequate to describe the software architectures of SoSs. To this end, the authors have changed the concept of 'System of Interest' by 'CS' and added an element entitled 'System-of-Systems (SoS)'. Then, they proposed to add a relationship link between them labeled 'participates on'.

Based on the previously mentioned approaches, we note that various modeling methods were adopted in the development lifecycle of SoSs; from the system, requirements level down to the software implementation level. The majority of the approaches have some advantages and disadvantages .i.e. all of them are only limited to dealing with some SoS concepts; namely, by taking into account the high-Level SoS requirements, understanding the CSs and their relationships and inter-dependencies, effective mission capability, etc.

Additionally, the adopted modeling methods come from a wide range of backgrounds, ranging from conceptual models to formal methods, hybrid methods, etc. The adopted modeling methods face several challenges: (1) some focus on describing the SoS as a whole, addressing structural organizations, ignoring how the CSs interaction; (2) others express the SoS at different levels of abstraction which is broad enough to cover the different aspects of SoS; (3) more still target the reasoning focusing less on detecting CSs behavior and how the goals and requirements change at runtime; (4) finally, and most importantly these approaches are not to mention a complete SoSE process. In fact, SoSE adopts

a structured, main three-process engineering (Conceptual design and CSs selection, Architectural Design, and Integration and deployment) to develop projects from Analysis through Implementation that permits releasing an efficiently finished SoS, satisfying stakeholders and performs as required.

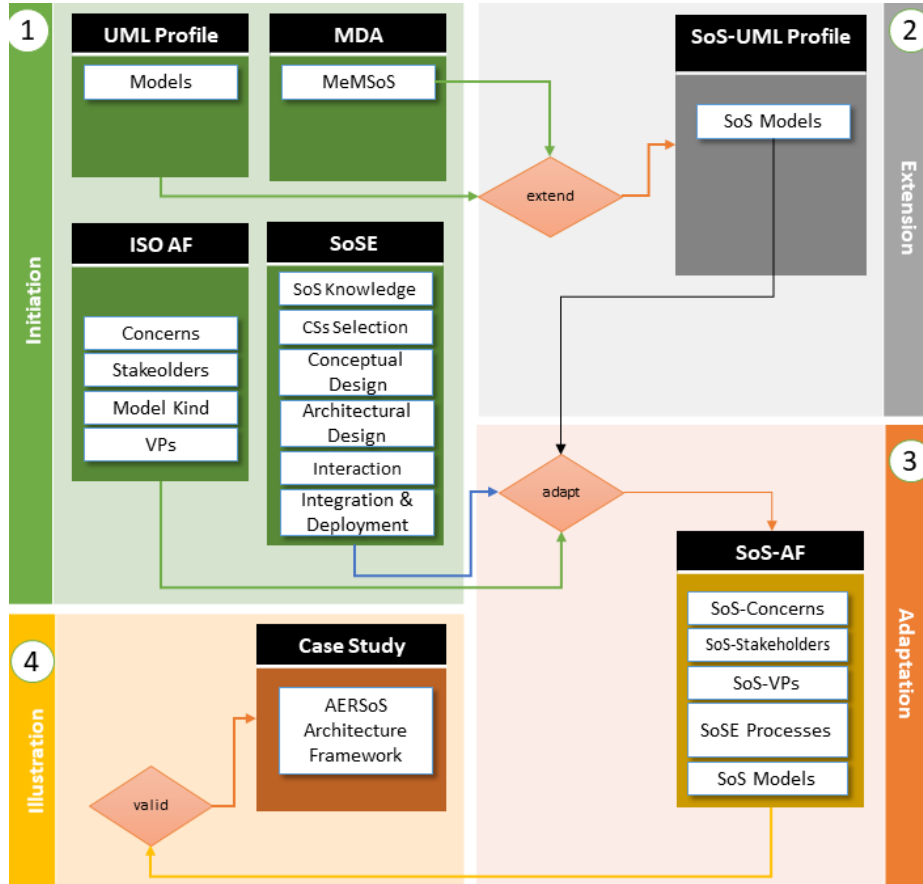
One possible solution for describing an SoSE and SoS architecture is to rely on standards such as ISO/IEC/IEEE 42010 [17]. However, in practice it is much too complicated to understand such standards, as they are considered too high-level to be used in practice, and also too complex to be easily understandable by the SoS designers. The proposal within this paper is to specify a framework named AF-SoS to satisfy the level of architectural maturity using ISO/IEC/IEEE 42010 as context.

## 4 Methodology and principles of the solution

The shortage of academic interest in the architectural standards of SoSs' domain and the lack of a single unified consensus of processes involved in the SoSE lead to the absence of consolidated AF of these types of systems [1]. Besides, Creating and managing a coherent SoS architecture framework is clearly a complex task. Therefore, the main purpose of this work is to follow the roadmap that we previously have suggested in [1] and define a Model-Based SoSE methodology, allowing decision-makers to design informed architectural solutions for the most well-known SoSs challenges. To this end, an Architecture Framework called SoS-AF that describes this methodology is introduced in this paper, it supports the SoSs development and resolves the common issues encountered in the SoSE.

Figure 3 below summarizes the framework, it can depict four different phases:

1. **Initiation phase:** describes the initiation methods and the architectural principles required to create our AF; namely, ISO Standard 42010, Model-Driven Approach (MDA), Unified Modeling Language (UML) profile and SoSE process model. This combination covers and connects different processes in a global framework. As well as it contains both the tools and the methods for constructing and managing SoSs architectures.
2. **Extension phase:** permits to use an MDA-based Meta-Model that we developed in the previous work as a reference to obtain extended UML models. In particular, it proposes a UML Profile-based modeling tool for SoSs called SoS-UML Profile. The latter provides a generic extension mechanism for building UML models in SoSs domain, and offer a way that reflects and refines the specifications of the new framework.
3. **Adaptation phase:** permits obtaining global AF viewpoints of the SoS by incorporating the first two phases. On the one hand, It is essentially based on the adaptation of "ISO/IEC/IEEE42010 standard: Systems and Software Engineering-Architecture Description"[17] to make it suitable to a Model-Based SoS Engineering context. And on the other hand, it highlights the adoption of SoSE processes and demonstrates how SoS-UML Profile can be leveraged to lead the production of SoSs architectures and to govern the



**Fig. 3.** Multi-Viewpoints Framework for SoSs' Architectures.

involved stakeholders. The strength of this methodology lies in migrating SoSE processes from one architecture viewpoint to another and at the same time mitigating common concerns.

- 4. Illustration phase:** involves a set of examples to validate some of SoS-AF related properties. It demonstrates that this work can offer a consistent AF for an SoS' case study (e.g. AERSoS). i.e. it represents a complete and more generic AERSoS-AF that is conducted throughout an orderly SoSE development processes.

The proposed SoS-AF will enable various stakeholders to separately design each process from different viewpoints. i.e, it gives a generic methodology to ensure that the resulting SoSs architecture models will also yield the desired expectations. Moreover, we argue that with this methodology and the SoS-UML profile-based model kind it provides, will offer a map to guide stakeholders towards achieving a unified SoS' AF.

## 5 A Multi-Viewpoints approach for the SoS Engineering

Architecture frameworks are mechanisms widely used in architecting. They establish a common practice for creating, interpreting, analyzing and using architecture descriptions within a particular domain of application or stakeholder community. As a result, their uses include, but are not limited to [18]:

- Creating architecture descriptions.
- Developing architecture modeling tools and architecting methods.
- Establishing processes to facilitate communication, commitments and inter-operation across multiple projects and/or organizations.

The idea is that an AF is a knowledge prefabricated structure that stakeholders can use to organize an architecture description into complementary views [19]. The specification of an AF is one area of the standardization in ISO/IEC/IEEE42010:2011. This standard proposes a conceptual model to describe the terms and concepts pertaining to systems and architecture description. This standard specifies an AF as a composition of multiple Viewpoints (VPs), each VP can be used to address specific concerns of different Stakeholders [17].

We aim to extend this standard to offer a comprehensive guideline to define an SoS-AF. It is specified by enhancing the concepts of the AF description represented in the international standard with the essential processes that an SoS' AF should encompass, as well as an SoS-UML Profile that identifies a set of Model Kind as a way to guide the SoS-AF construction (red rectangles in Figure 4).

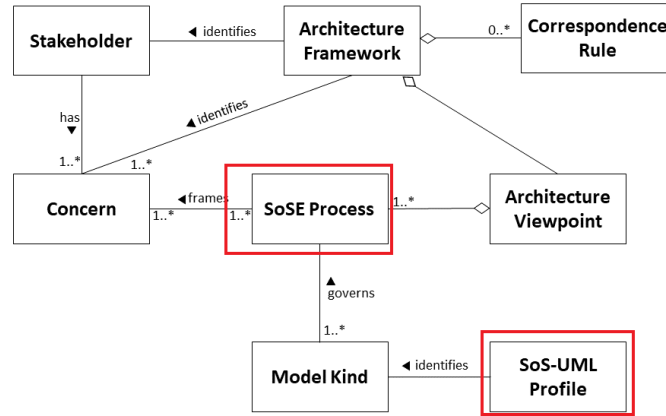


Fig. 4. An extended conceptual model of the SoS-AF.

The proposed SoS-AF conforms to the standard IEEE 42010:2011 and its description is motivated by the different concerns commonly shared by SoSs'

stakeholders across various development processes. Therefore, to form a collection of architecture VPs that constitutes the body of our SoS-AF description, we provide a comprehensive UML profile-based modeling basis for the notion of SoSs that can guide the development of SoSE processes pertaining to the shared concerns of each involved Stakeholder.

Consequently, our SoS-AF model is introduced as a composition of a set of VPs, each VP is created through an aggregation of one or more SoSE processes. A SoSE process is governed by a set of Model Kind appropriate to specific concerns to be addressed by different Stakeholders. These Model kinds are specified by SoS UML Profile. i.e. SoS-UML profile defines a modeling tool that we develop to describe the associated Model kind that governs different SoSE processes and their underlying relations and dependencies. In the following, the SoS-AF, Concerns, Stakeholders, SoSE Process, VPs and Model Kinds are described.

### 5.1 Concerns

Concerns arise throughout the development life-cycle of SoSs, from the CSs knowledge level down to design and implementation level. A concern may appear in many forms, such as stakeholder relationships, SoSs' global objectives, Capabilities, requirements, modeling constraints, CSs' interdependencies, Quality Attributes, design decisions or other issues that pertain to any influence on SoS in its environment. Through our previous studies [16, 1], we were able to extract these concerns by exploring multiple aspects related to SoSE; those concerns are framed in this study according to five SoSE processes as follows.

### 5.2 SoSE processes

SoSE processes have a major stake in the SoS-AF and strong influence in its implementation. Our contribution consists in inspiring from the SoSE processes provided by [20] to guide the SoSs lifecycle processes from CSs knowledge, to design and implementation. We propose to take inspiration from this work by modifying some elements to adapt it to our previous works.

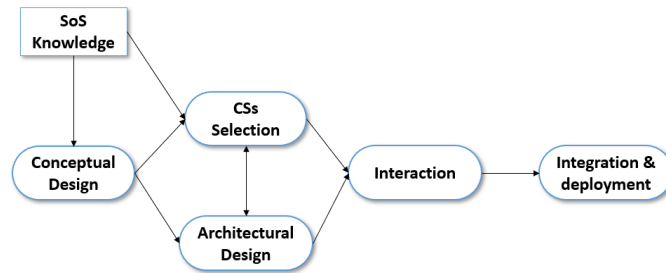


Fig. 5. The proposed SoSE development processes (Adapted from [20]).

SoSE processes express the activities engaged under SoSE from the perspective of one or more Stakeholders to frame specific Concerns, using the conventions established by its models. Each SoSE process will be identified by one or more governing Model Kind that adheres to the conventions of SoS UML Profile. However, as shown in Figure 5, the main SoS development processes involved in our SoS-AF are:

- SoS Knowledge: addresses high-Level SoS requirements and investigates existing CSs that can participate in the SoS.
- CSs Selection: this process consists of choosing a set of CSs and distinguishing their relevant Capabilities and Goals.
- Conceptual Design: the design involves creating a global vision of an SoS, defining the essential relationships and identifying mission capability assessment.
- Architectural Design: represents a global architecture for the SoS' constituents and their possible Roles. It could be developed in parallel with the CSs Selection process.
- Interaction: the different CSs involved in an SoS usually have different Capabilities. Therefore, a large part of the software engineering effort in the SoSE is to design interactions so that the CSs can interoperate.
- Integration & deployment: this process implies that the different CSs involved in the SoS work together and interact through the assigned Roles. Deployment of the system consists of setting up the CSs interactions in the organizations concerned and making it operational.

### 5.3 Viewpoints

A viewpoint in SoS-AF is a selection of relevant aspects of the SoSE processes (and their Stakeholders' concerns); and the representation of that part of an architecture that is expressed in different Model Kind. It is claimed that the SoSE processes form a necessary and sufficient set to meet the needs of SoS-AF. Four main VPs are identified in our proposed SoS-AF:

- Business viewpoint: the VP of Business includes the knowledge and requirement of CSs, and the creation of a rough draft of the SoS requirements including identification of the possible Capabilities, which could be derived from the existing CSs of the application domain.
- Analysis viewpoint: aims to analyze the problem and the addressed solution, it starts by selecting the appropriate CSs involved in providing the required SoS capabilities and then by building the conceptual design model, which will offer a global understanding of CSs, their relationships and interdependencies as part of the SoS.
- Design viewpoint: supports architects and designers in the design processes from initial sketch to detailed design. It can deal with the fundamental processes of an SoS development and the discipline of designing such architectures, CSs, Roles, and their collaborations.

- Deployment viewpoint: is a process of merging two or more diverse Roles that are designed to define, control, and monitor complex interactions that extend across SoS and CSs boundaries.

#### 5.4 Stakeholder

Stakeholders are inherently heterogeneous due to multiple users, their VPs, engineering processes, platforms, environments... etc. Stakeholders are individuals, groups or organizations holding Concerns for an SoS. They use SoS-AF Description to understand, analyze and compare SoS' architectures. Each SoSs' concern could be managed by one or more stakeholders; four main stakeholders have been identified in this framework:

- SoS Experts: a group of persons responsible for the Business VP establishment, with strong theoretical knowledge in an SoS application domain and they have the ability to understand the practical implications of the existing CSs that can participate in SoS and can translate SoS Capability Objectives into High-Level SoS Requirements.
- Architects and Designers: their role is vital to the success of both Analysis and Design VPS, they translate the requirements into a demand for CSs Capabilities. i.e. they look at business plans and requirements provided by SoS Experts, analyze the Goals and the Capabilities of the selected CSs, and propose recommendations on the right selection of CSs to achieve the SoS' Global Goal.
- Collaboration Specialists: they are also Analysis and Design experts; they are responsible for understanding CSs and their Capabilities' collaboration. They also look at integrations with existing CSs, interfaces with people and other systems.
- Interactions Engineers: persons responsible for the Deployment VP and they are responsible for specifying communication and interactions between different Roles. They oversee the CSs' Capabilities and their Roles' interactions to facilitate the interaction modeling within an SoS application.

#### 5.5 Model kinds

The proposed SoS-AF must support the modeling of all the concepts and relationships of the SoSs entities, with their different static and dynamic aspects. Thus, we propose an "SoS-UML Profile for SoS-AF". In Figure 6, we summarize the needed diagrams for each VP. The SoS-UML profile introduces a graphical construct to represent the *requirements diagram* and relate it to other Static and/or Behavior diagrams. The static diagrams include the *Goals diagram*, *Domain model diagram* and *Constituent diagrams*. The SoS behavior diagrams are represented by the *Capabilities diagrams*, *Roles Interaction diagrams*, *Capabilities Collaboration diagrams* and *Roles Interfaces diagram*.

The concrete syntax of SoS-UML profile is formulated to graphically represent a set of Model Kind using UML notation for each SoS aspect. This SoS-UML

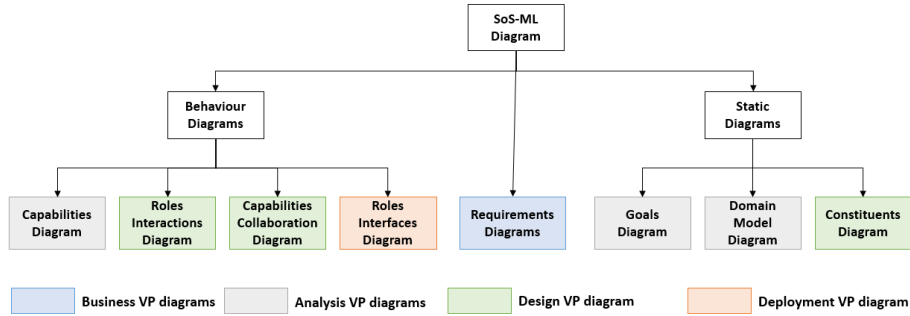


Fig. 6. *SoS-UML Profile's diagrams.*

profile, its diagrams and all their associated concepts will be detailed in the next section.

## 6 SoS-UML Profile: UML Extensions for Modeling SoS-AF

SoS-AF is our proposed architecture framework to support the entire development of SoSs' lifecycles, including its various processes from the requirements' specification process, design, to the implementation process. Hence, the need for a tool allowing the definition of the overall architecture of SoSs, and also the definition of specific models for each involved process (such as UML models) is essential. For this reason, we present a UML2.0 extension tool, denoted SoS-UML profile, for the management of SoS' architectures following the approach proposed in SoS-AF.

A key engineering problem then is to construct an Architecture Framework of the AERSoS (AERSoS-AF) based on the proposed AF-SoS, and this requires to :

- Create different VPs matching the VPs of SoS-AF at high-level abstraction,
- Take advantage of SoSE processes that can be used to manage the AERSoS,
- Use the SoS-UML Profile to abstract their analysis and design from implementation technologies, increase the automation of the development of AERSoS and allow VPs modeling.

To be able to specialize the UML for the SoSs domain, we needed to extend the MeMSoS to integrate new entities and constructions adapted to treat SoSs. Figure 7 shows the new version of MeMSoS (black rectangles: unchanged entities, green rectangle: updated entities and red rectangle: new entities). Consequently, MeMSoS' will represent the definition of SoS-UML Profile below, and defines the available building elements and how they can be assembled.

The new version of MeMSoS offers better support of different VPs modeling, frames Stakeholders concerns and introduces new features. This new version





As well as, we work on an instance of Eclipse which loads the plugins that we generate previously, in order to be able to design a set of examples of our models.

This section presents the abstract/concrete syntaxes of the SoS-UML profile, which is structured into packages labeled by the SoSE development processes' names to make groupings of different aspects, and thus better manage the complexity of each process. Figure 8.(a) shows a screenshot of the different packages in IBM RSA tool that make up the Profile's Meta-Model, it involves six packages namely: *SoS\_Knowledge\_Package*, *CS\_Selection\_Package*, *Conceptual\_Design\_Package*, *Architectural\_Design\_Package*, *Interaction\_Package* and *Integration\_Deployment\_Package*. Additionally, Figure 8.(b) shows their relevant interdependencies as mentioned in the SoSE processes model. In the following, we will demonstrate how we use each package of them as a model kind to design the main concepts of each process in the SoSE development.

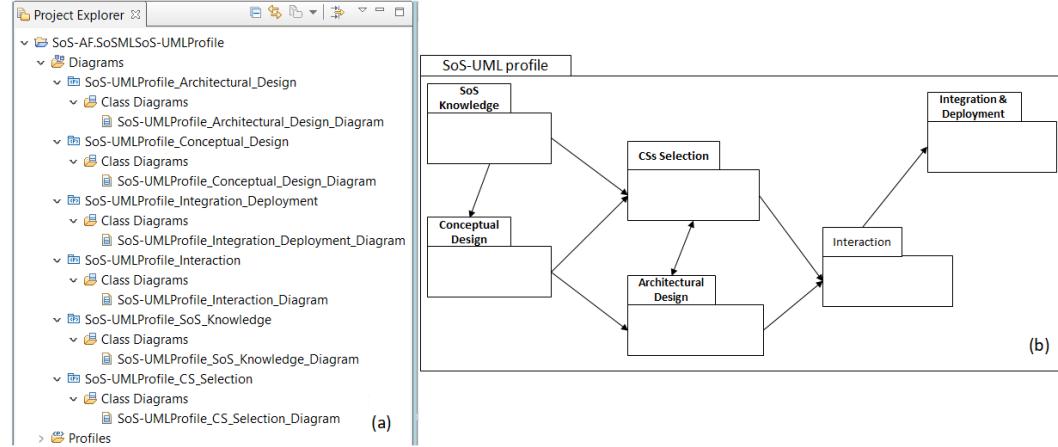
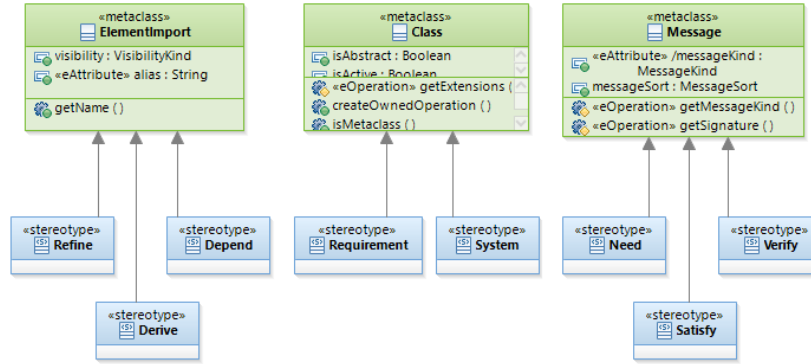


Fig. 8. SoS-UML profile packages.

## 6.1 SoS\_Knowledge\_Package

The SoS\_Knowledge\_Package describes the basic elements needed to describe the SoS Knowledge. The deliverable contained in this package will guide the description of the SoS' Goals and support the identification of its Capabilities during the next process. Particularly, the two stereotypes Requirement and System are the central concepts in this model, and they represent a unit SoS Knowledge process. The latter starts with understanding the desired Requirement and suggesting a set of Systems as various options for achieving that Requirement. It is used for the representation of the Requirement Diagram's Model. The package contents are shown in the next Figure 9.



**Fig. 9.** The structure of the *SoS\_Knowledge\_Package*.

The Requirements Model represents the functionalities or the conditions that an SoS must fulfill based on the contributions of the collaborative CSs. As shown in the package figure, it contains different stereotypes for describing the knowledge and requirement of an SoS, and how they can be related to the necessary entities to gather, organize, analyze and decompose the different existing systems that can participate in an SoS. A part of this diagram, describing the most important stereotypes and the extended meta-classes is shown in Table 1.

**Table 1.** Stereotypes of the *SoS\_Knowledge\_Package*.

Process	Model Kind	UML Diagram	Stereotypes	Description	Meta-class
SoS knowledge	Requirement Diagram	Class Diagram	Requirement	Capability or Goal that must (or should) be performed	Class
			System	Systems, SoS, CS... etc	
			Refine	Clarifies the requirement's meaning or context	ElementImport
			Depend	A requirement uses or depends on other ones	
			Derive	Impose additional sub-requirements	
			Need	Express required systems for a requirement	Message
			Verify	Relate a requirement with a system that verifies it	
			Satisfy	Relate a requirement with a system that satisfies it	

Take the AERSoS case study, requirements can be organized as an ordered tree hierarchical structure. A typical structure may include a top-level requirement for all sub-requirements. By using different relationships, each requirement

within the top-level one may be associated with different systems (SoSs or CS, component, etc.) to describe its scope; for example (see Figure 10), the *«Requirement» HandleAbnormalSituation* which is derived from *«Requirement» EnsureAircraftSafety* can be satisfied by *«CS» AircraftEmergencySystem* using the two relationships *«Derive»* and *«Satisfy»* respectively.

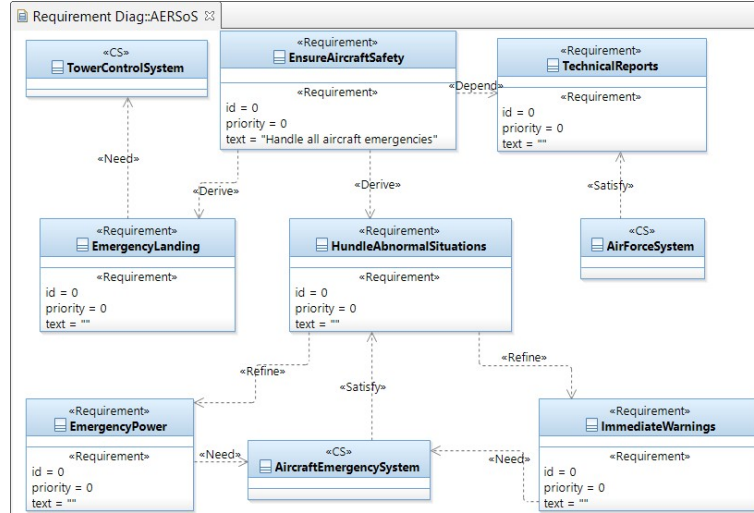


Fig. 10. Requirements diagram for AERSoS.

## 6.2 CSs\_Selection\_Package

The Selection of CSs requires the characterization of the Capabilities in which the CSs will perform to fulfill their Goal, and thus, the SoS' global Goal. Therefore, the main Systems that can participate in the SoS must be defined, described, and documented using the CSs\_Selection\_Package. As shown in the structure of this package that is depicted in Figure 11, the extensions proposed here comprise stereotypes that reflect the entities that constitute the basis for the specification of Systems' Goals (CS\_Goal and SoS\_Goal) and the identification of their relevant Capabilities (CS\_Capability and SoS\_Capability).

The CS\_Selection\_Package uses the output artifacts of the SoS Knowledge process (Requirements) to describe the SoS in terms of Goals and Capabilities. Thus, the notions of CS\_Goal and SoS\_Goal are the central concepts in the Goals Model, representing a unit of the local goals of CSs as well as the SoS global goal in which high-level Goals may be realized through the combination of lower-level Goals. As well as, the Capabilities Model contains the concepts enabling the description of Capabilities which the selected CSs should perform

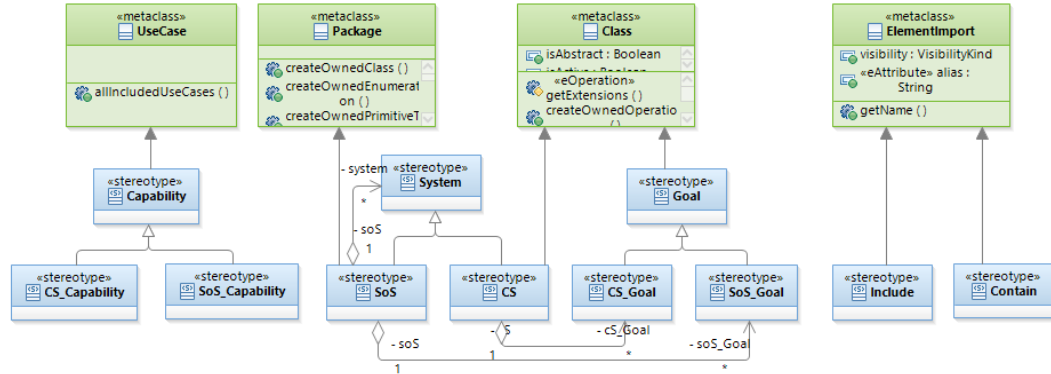


Fig. 11. The structure of the CSs\_Selection\_Package.

Table 2. Stereotypes of the CS\_Selection\_Package.

Process	Model Kind	UML Diagram	Stereotypes	Description	Meta-class
CSs Selection	Goals Diagram	Class Diagram	Goal	Represents objectives of a system	Class
			CS_Goal	Represents objectives of a CS	
			SoS_Goal	Represents objectives of an SoS	
			Include	Split a goal into several sub ones	ElementImport
			Contain	Expresses the capability of having sub-goals	
	Capabilities Diagram	Use Case Diagram	System	Could be any type of System providing a Capability	Package
			SoS	SoS providing the Capability	Class
			CS	CS providing the Capability	
			Capability	Refers to functions provided by any system	
			CS_Capability	Refers to functions provided by any system	
SoS_Capability	Refers to functions provided by an SoS				

to achieve the predefined Goals. The concepts in this package are divided into two diagrams, Goals Diagram and Capabilities Diagram (listed in Table 2).

The first diagram in this package is Goals Diagram where different Goals can be organized as a tree structure in which a high level Goal that represents «SoS\_Goal» may be realized through the combination of lower level Goals «CS\_Goal» of CSs. In addition, relations between them denote sharing of the same common Goals; for example, Figure 12, the «SoS\_Goal» *Aircraft Safety* has three sub-goals, «CS\_Goal» *Safe Landing*, «CS\_Goal» *Safe Flight* and «CS\_Goal» *Accident Report*. For the second diagram (Capabilities Diagram), it can be viewed as a mechanism to capture the SoS Capabilities in terms of the Capabilities of the pre-selected CSs which specify the expected behavior (what), and not the exact method of making it behave (how) of an SoS and thus, it represents a black-box view of the SoS; it is therefore well suited to serve later in Interactions

and Architectural Design Diagrams. Figure 13. represents the AERSoS capabilities, where a set of sub-capabilities (e.g. «CS\_Capability» *GeneratingPower*, «SoS\_Capability» *InsulatingReactors*) for different CSs are required to perform the global-Capability «SoS\_Capability» *ControllingSituation*.

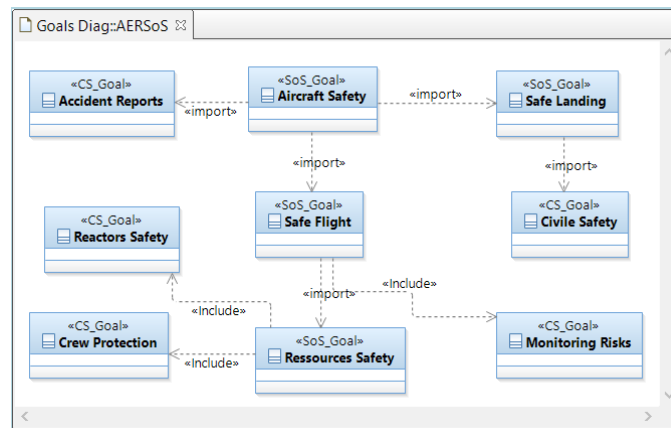


Fig. 12. Goals Diagram for AERSoS.

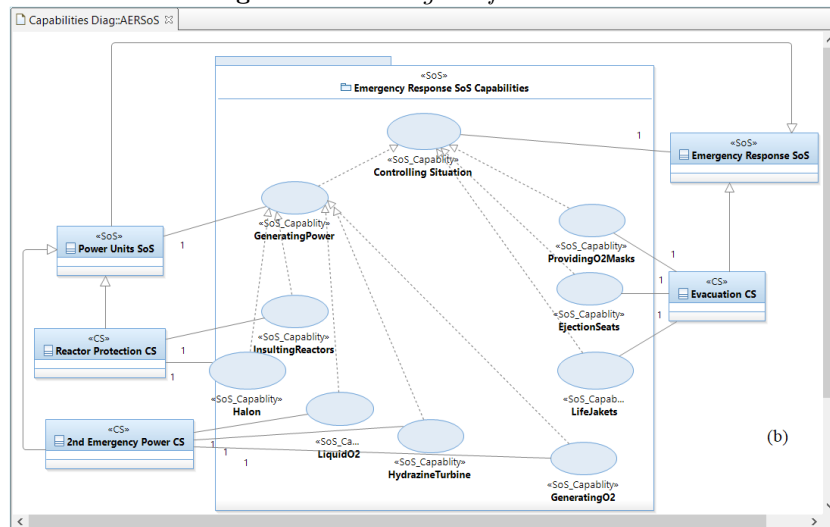


Fig. 13. Capabilities diagram for AERSoS.

### 6.3 Conceptual\_Design\_Package

This package captures the main blocks for designing the Conceptual Design process, the design allows creating a global vision of an SoS, defining the essential

relationships and identifying mission capabilities. The main concepts in this model are Systems, Roles, Capabilities and different relationships types that can offer a global understanding of CSs, their Roles and their interdependencies as part of an SoS. A general structure of the package is depicted in Figure 14.

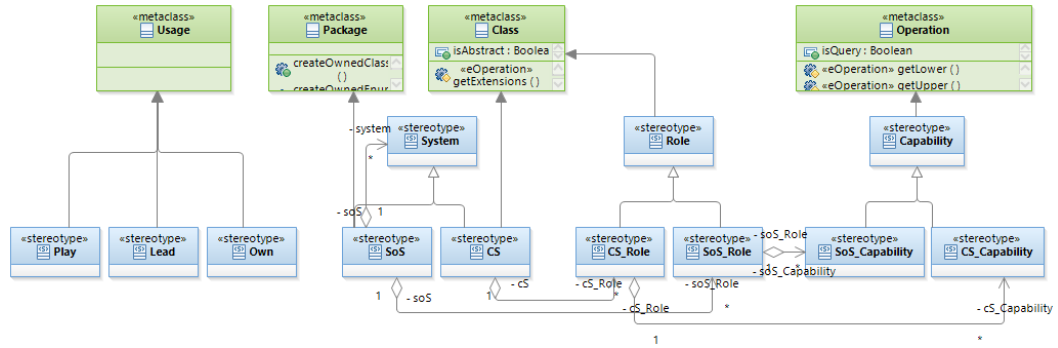


Fig. 14. The structure of the Conceptual\_Design\_Package.

Table 3. Stereotypes of the Conceptual\_Design\_Package.

Process	Model Kind	UML Diagram	Stereotypes	Description	Meta-class
Conceptual Design	Domain Model	Class Diagram	System	Represents the involved System	Class
			SoS	Represents the involved SoS	
			CS	Represents the involved CS	
			Role	Ideal behavior of any type of System	
			CS_Role	Ideal behavior of a CS	
			SoS_Role	Ideal behavior of an SoS	Operation
			Capability	Represents Capabilities of a System in specific Role	
			CS_Capability	Represents Capabilities of a CS in specific Role	
			SoS_Capability	Represents Capabilities of an SoS in specific Role	Usage
			Own	Expresses authority of one system to another	
			Lead	Expresses control or guidance of one system to another	
Play	Associates systems with the required Roles				

The stereotyped concepts in this model can be used to provide a global structure of an SoS to enhance the interaction of its CSs. Consequently, they can be used to describe the CSs, the Capabilities they have to accomplish Goals and the Roles they play within an SoS. In Addition, the *Conceptual\_Design\_Package* defines to which a Sub-System has access to and which Role it can play to solve missions. The concepts of this package are introduces one single diagram called Dmain Model as showed in the Table 3.

At this stage, the Domain Model Diagram is used by stakeholders to design the SoS' characteristics in terms of its structural CSs, behavioral Roles, the internal Capabilities and relationships between the CSs. An example of *PowerUnitsSoS* as it is depicted in Figure 15, the stakeholders can display various kinds of CSs and SoSs that constitute the top-level entities, e.g. «SoS» *PowerUnitsSoS*, «CS» *ReactorsProtectionCS*, etc. their corresponding Roles, e.g. «SoS\_Role» *PowerGenerator* «CS\_Role» *OxiginProvider*, etc. and relationships among them «Play» «Lead», etc.

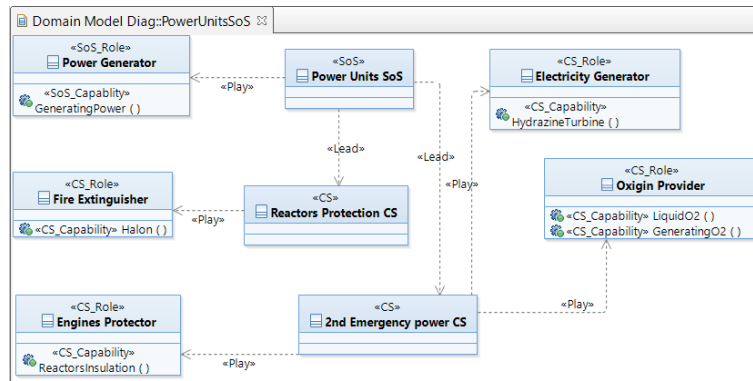


Fig. 15. Domain Model for the *PowerUnitsSoS*.

#### 6.4 Architectural\_Design\_Package

This package which is depicted in Figure 16 presents the concepts to support the Architectural Design decision in every CS' architectures. This is required to propagate the CS' architectural characteristics in the next processes in the lifecycle of an SoS. They manifest the structure of every CS by characterizing which Roles are part and which functions are used by the different stakeholders.

The Constituent Model has the ability to describe the internal structure of every autonomous entity cooperating within the SoS and how the operational and managerial independence can be defined. Additionally, the *Architectural\_Design\_Package* defines which CSs' stakeholders have access to and which functions they can perform. Detailed stereotypes are summarized in the Table 4.



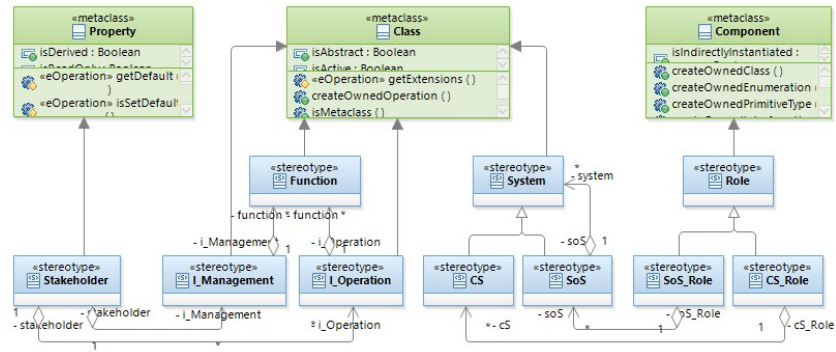


Fig. 16. The structure of the Architectural\_Design\_Package.

Table 4. Stereotypes of the Architectural\_Design\_Package.

Process	Model Kind	UML Diagram	Stereotypes	Description	Meta-class
Architectural Design	Constituent Diagram	Component Diagram	System	Refers to system's class	Class
			CS	Refers to a CS' class	
			SoS	Refers to an SoS' class	
			I_Operation	Independent operations	
			I_Management	Independent management	
			Function	Represents a service	
			Role	Roles of a system	Component
			CS_Role	Roles of a CS	
			SoS_Role	Roles of an SoS	
			Stakeholder	Intervening persons	Property

Figure 17 shows an example of a Constituent Diagram used to model the decomposition of the EvacuationCS and its internal entities such as functions (e.g. «Function» *extinguish*, «Function» *manage*) which are associated to one of the main independence classes: the management class with the stereotype «I\_Management» *CSManagement* and the operation class with stereotype «I\_Operation» *CSOperation*, as well as, each function involves the corresponding stakeholders as attribute, e.g. «Stakeholder» *Manager* and «Stakeholder» *FireFighter*, respectively.

## 6.5 Interaction\_Package

The Interaction\_Package contains the concepts to describe how flexible collaboration and cooperation take place between different CSs in an SoS. This package's model supports the Interactions process by focusing on the quality of the interaction architecture and, as a consequence. We define two major Models: The Capabilities Collaboration Model and the Roles Interactions Model. Both determining the types of Collaborations among Collaborative Capabilities

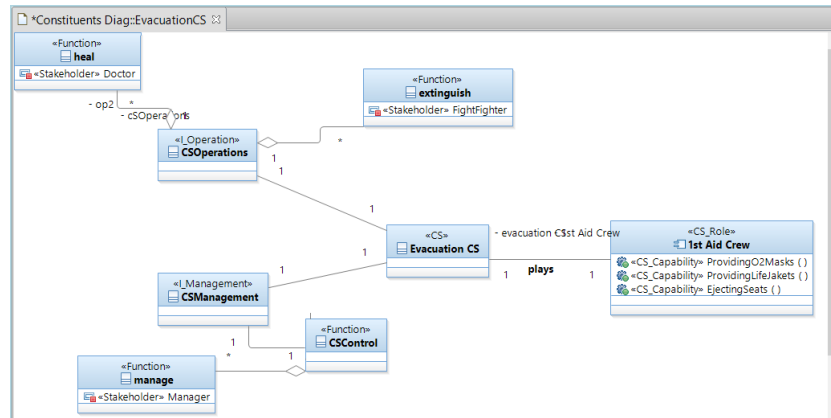


Fig. 17. Constituent diagram for EvacuationCS.

and the interactive Roles. The package’s stereotyped concepts are represented in Figure 18.

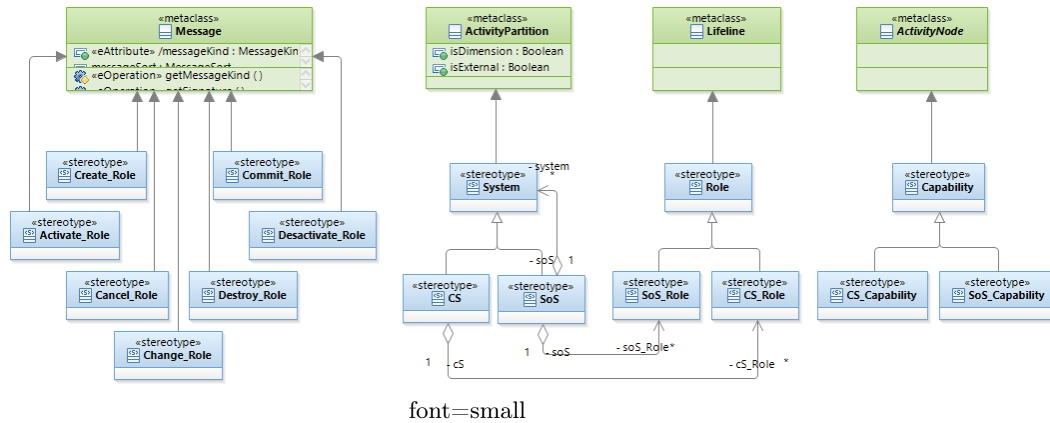


Fig. 18. The structure of the Interaction\_Package.

In the Capabilities Collaboration diagram, we define the stereotypes to describe the internal behavior of different CSs used for fulfilling predefined Goals. The global-Goals of an SoS can be achieved in terms of combining simple Capabilities of its participating CSs. Additionally, the Roles Interactions diagram covers the abstract representations of the collaborative Capabilities of different CSs within an SoS. Moreover, the Role package provides the different relationships that can be used to connect CSs among each other. The identified stereotypes are summarized in Table 5.

Process	Model Kind	UML Diagram	Stereotypes	Description	Meta-class
Interactions	Capabilities Collaboration	Activity Diagram	System	System performing Capabilities	ActivityPartition
			CS	CS performing Capabilities	
			SoS	SoS performing Capabilities	
			Capability	functions provided by a System	ActivityNode
			CS_Capability	functions provided by a CS	
			SoS_Capability	functions provided by an SoS	
	Roles Interactions	Sequence Diagram	Role	Interactive Role of a System	Lifeline
			CS_Role	Interactive Role of a CS	
			SoS_Role	Interactive Role of an SoS	
			Capability	Refers to a Capability of a System	Activation
			CS_Capability	Refers to a Capability of a CS	
			SoS_Capability	Refers to a Capability of an SoS	
			Create_role	Initiating a new Role	Message
			Destroy_role	Finishing a Role	
			Activate_role	Starting a Role	
			Deactivate_role	Interrupting a Role	
Cancel_role	Omitting a Role				
Change_role	Replacing a Role				
Commit_role	Performing a Role				

**Table 5.** Stereotypes of the Interaction\_Package.

The Figure 19 shows an example of Capabilities Collaboration diagram which describes “AssessingRisks” Capability of «SoS»WarningSoS collaborating with other CSs’ Capabilities («CS\_Capability»CO\_sensing and «CS\_Capability»IR\_radiation of «CS»FireDetectionCS) and («CS\_Capability»Transducer\_probe of «CS»IceDetectionCS). This diagram particularly offers a good method to express the flow of capabilities of the «SoS»WarningSoS and how its CSs can collaborate.

We can use the Roles Interactions diagram Figure 20, to show how the different Roles interact within the «SoS»AERSoS when fulfilling the global goal «SoS\_Goal»SafeLanding in case of critical situations in the aircraft. This diagram depicts a collection of interactions between external Roles of different CSs «SoS\_Role»LandingManager, «CS\_Role»EmergencyLandingController, etc. In this case, the Roles represent the specification of a sequence of Capabilities («CS\_Capability»first\_aid, «SoS\_Capability»landing...etc.), that an SoS (or CS) can perform. In addition, the roles represent a path or flows of a sequence of interactions (e.g. «Change\_role»Unplanned\_Landing, «Create\_role»pilot... etc.) that occurs during the execution to accomplish the SafeLanding goal.

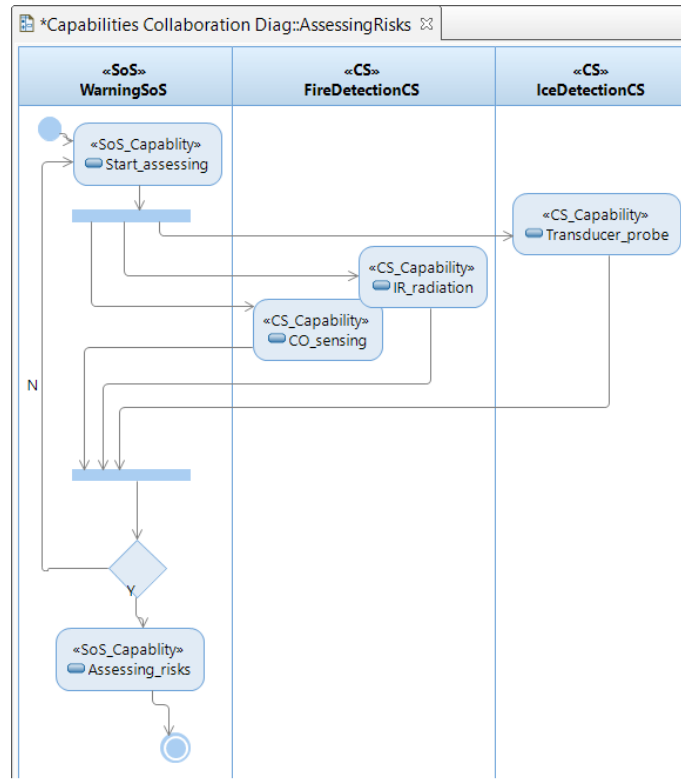


Fig. 19. Capabilities Collaboration diagram for WarningSoS.

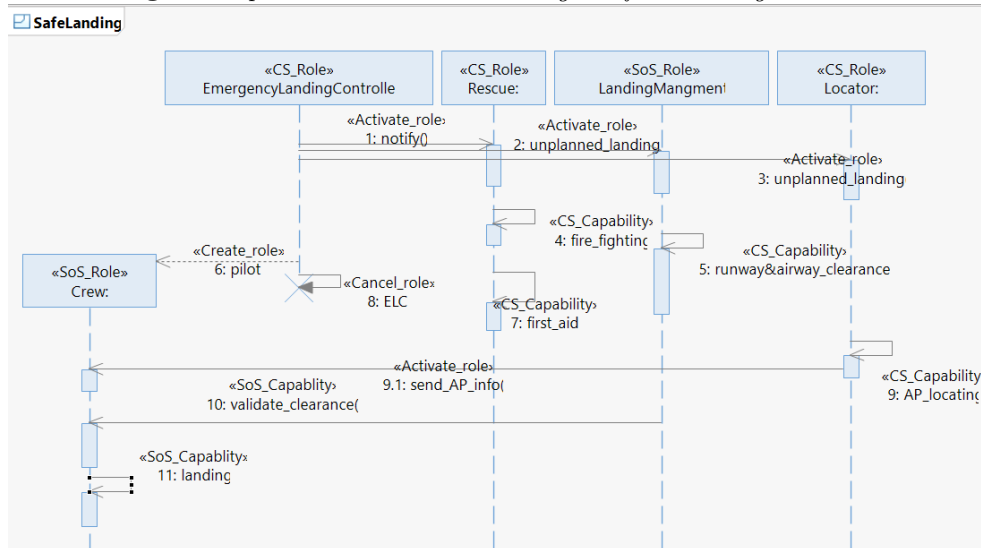


Fig. 20. Roles Interactions diagram for SafeLandingCS' Roles.

## 6.6 Integration\_Deployment\_Package

The Integration\_Deployment\_Package (Figure 21) contains the concepts to describe the process of Deployment, including instances of Roles and the corresponding Interfaces. This model contains the stereotypes of CS' interfaces defined by their Roles and the necessary provided or/and required Interfaces (RUIs for Relied Upon Interfaces).

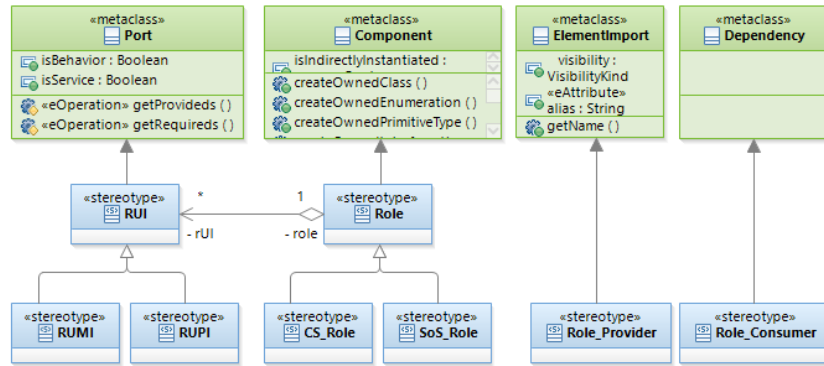


Fig. 21. The structure of the Integration and deployment package.

The different stereotypes in Table 6 included in this diagram describe the aspect of Integration of an SoS itself. In this case, the Integration and deployment diagram describes the physical deployment of different interfaces required or provided by CSs. The interfaces extend the meta-class port to specify the interaction points among CSs supporting the integration of behavior and structure.

Process	Model Kind	UML Diagram	Stereotypes	Description	Meta-class
Integration and Deployment	Roles Interfaces	Component Diagram	System	Integrated System	Component
			CS	Integrated CS	
			SoS	Integrated SoS	
			RUI	Relied Upon Interface	Port
			RUMI	Relied Upon Message Interface	
			RUPI	Relied Upon Physical Interface	
			Role_Provider	Role providing a RUI	ElementImport
			Role_Consumer	Role consuming a RUI	Dependency

Table 6. Stereotypes of the Integration\_Deployment\_Package.

The Figure 22 shows an example of this diagram of PowerUnitsSoS. the stakeholders assemble a set of Relied Upon Message or/ and Physical Interfaces (e.g. «RUMI»I\_OP and «RUPI»I\_RP1) and their associations (e.g. «Role\_Provider»control\_Engines and «Role\_Consumer»provide\_O2) that constitute the basic elements to define how the CSs of one SoS can collaborate among each other to realize the integration of structure and/or behavior of the SoS.

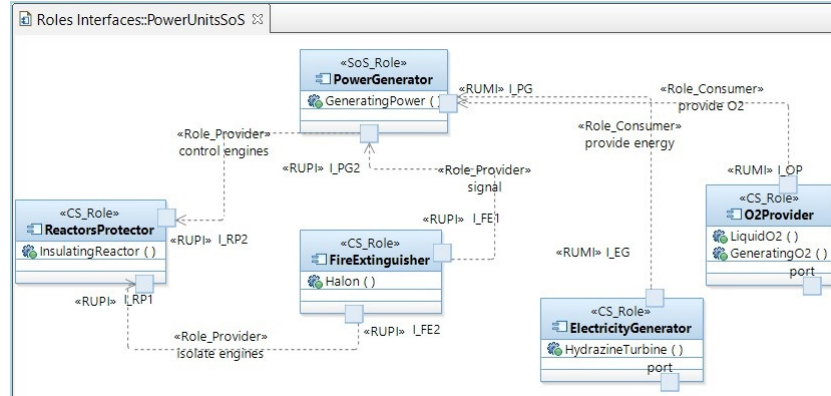


Fig. 22. Roles Interfaces diagram of PowerUnitsSoS.

## 7 Conclusion

In this paper, we presented a multi-viewpoint Architecture Framework called SoS-AF which is understandable and easily manipulated by different stakeholders. This methodology aims to offer the audience of SoSs' Stakeholders the tools to facilitate the task of developing a multi-viewpoint architecture that is managed by a new SoSE processes and documented through the SoS-UML profile's models.

Besides, this approach conforms to a very widespread standard in software architectures community "IEEE 42010" which was designed in order to standardize the definition of Systems and Software Engineering-Architecture description. Specifically, SoS-AF inherits the definitions of the main elements which are part of this standard, and extends them by the two elements "SoSE process" and "SoS-UML Profile". It is based around the construction of multi-viewpoint SoSs' architectures, through the definition of several viewpoints for a given SoS architecture, which is in our example we used an Aircraft Emergency Response System-of-Systems (AERSoS) as a case study.

The first extension of the standard is to integrate the notions contained in the SoSE process. This adoption allows the SoS-AF to pass through several processes. At the end of each process, each one of the involved stakeholders must

raise the level of concretization of his architecture by creating a set of models that better meet his essential concerns. The stakeholders can benefit from the SoS-UML Profile to support the design of all the concepts and relationships of an SoS' CSs. This tool defines structural diagrams: Goals diagram, Domain model diagram and Constituents diagrams. In addition, to behavioral diagrams which aim to represent the dynamic aspects of an SoS: Capabilities diagrams, Roles Interaction diagrams, Capabilities Collaboration diagrams and Roles Interfaces diagrams. The visual syntax allows using diagrams to manage the SoSE development processes by its audience of stakeholders within the SoS-AF's Viewpoints.

To the best of our knowledge, this is the first paper supporting the modeling of SoSs' multi-viewpoints architectures using a SoS-UML Profile to manage the development lifecycle. However, there can be a need for enhancing the SoS-AF by considering some non-functional requirements to provide more comprehensive support. In addition, the proposed SoS-AF, can treat Cyber-Physical Systems (CPS) implicitly as SoSs. Therefore, and as a future enhancement, the SoS-AF also needs to be extended to other instances such as CPSs.

## References

1. Dridi, C. E., Benzadri, Z., & Belala, F. (2020, November). System of Systems Modelling: Recent work Review and a Path Forward. In 2020 International Conference on Advanced Aspects of Software Engineering (ICAASE) (pp. 1-8). IEEE.
2. Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering: The Journal of the International Council on Systems Engineering*, 1(4), 267-284.
3. ISO/IEC/IEEE 15288, Systems and software engineering - System lifecycle processes.
4. Department of Defense (DoD), 2004, Defense Acquisition Guidebook Ch. 4.2.6. "System of Systems Engineering," Washington, DC: Pentagon, October 14.
5. Cocks, D. (2006, July). How Should We Use the Term "System of Systems" and Why Should We Care?. In INCOSE International Symposium (Vol. 16, No. 1, pp. 427-438).
6. Kotov, V. (1997). Systems of systems as communicating structures (Vol.119). HP Labs.
7. DeLaurentis, D. (2005, January). Understanding transportation as a system-of-systems design problem. In 43rd AIAA Aerospace Sciences Meeting and Exhibit (p. 123).
8. Gunes, V., Peter, S., Givargis, T., & Vahid, F. (2014). A survey on concepts, applications, and challenges in cyber-physical systems. *KSII Transactions on Internet & Information Systems*, 8(12).
9. Nielsen, C. B., Larsen, P. G., Fitzgerald, J., Woodcock, J., & Peleska, J. (2015). Systems of systems engineering: basic concepts, model-based techniques, and research directions. *ACM Computing Surveys (CSUR)*, 48(2), 1-41.
10. Jamshidi, M. (2008, December). System of systems-innovations for 21st century. In 2008 IEEE Region 10 and the Third international Conference on Industrial and Information Systems (pp. 6-7). IEEE.
11. Wickramasinghe, N., Chalasani, S., Boppana, R. V., & Madni, A. M. (2007, April). Healthcare system of systems. In 2007 IEEE International Conference on System of Systems Engineering (pp. 1-6). IEEE.

12. Lane, J. A., & Epstein, D. (2013). What is a System of Systems and why should I care?. University of Southern California.
13. Dahmann, J. S. (2015). Systems of systems characterization and types. *Systems of Systems Engineering for NATO Defence Applications (STO-EN-SCI-276)*, 1-14.
14. Assaad, M. A., Talj, R., & Charara, A. (2016, July). A view on Systems of Systems (SoS). In 20th World Congress of the International Federation of Automatic Control (IFAC WC 2017) –special session.
15. Aljohani, T. M. (2018). Analysis of the Smart Grid as a System of Systems. arXiv preprint arXiv:1810.11453.
16. Dridi, C. E., Benzadri, Z., & Belala, F. (2020, June). System of Systems Engineering: Meta-Modelling Perspective. In 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE) (pp. 000135-000144). IEEE.
17. ISO/IEC/IEEE42010, Systems and software engineering — Architecture description, ISO, December 2011.
18. May, I. S. O. (2011). Systems and software engineering–architecture description. Technical report, ISO/IEC/IEEE 42010, 2011.
19. Emery, D., & Hilliard, R. (2009, September). Every architecture description needs a framework: Expressing architecture frameworks using ISO/IEC 42010. In 2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (pp. 31-40). IEEE.
20. Arass, M. E., Ouazzani-Touhami, K., & Souissi, N. (2019). The system of systems paradigm to reduce the complexity of data lifecycle management. Case of the security information and event management. *International Journal of System of Systems Engineering*, 9(4), 331-361.
21. Barbi, E., Cantone, G., Falessi, D., Morciano, F., Rizzuto, M., Sabbatino, V., & Scarrone, S. (2012, July). A model-driven approach for configuring and deploying systems of systems. In 2012 7th International Conference on System of Systems Engineering (SoSE) (pp. 214-218). IEEE.
22. El Hachem, J., Pang, Z. Y., Chiprianov, V., Babar, A., & Aniorte, P. (2016, December). Model driven software security architecture of systems-of-systems. In 2016 23rd Asia-Pacific Software Engineering Conference (APSEC) (pp. 89-96). IEEE.
23. Mori, M., Ceccarelli, A., Lollini, P., Bondavalli, A., & Frömel, B. (2016, January). A holistic viewpoint-based SysML profile to design systems-of-systems. In 2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE) (pp. 276-283). IEEE.
24. Mori, M., Ceccarelli, A., Lollini, P., Frömel, B., Brancati, F., & Bondavalli, A. (2018). Systems-of-systems modeling using a comprehensive viewpoint-based SysML profile. *Journal of Software: Evolution and Process*, 30(3), e1878.
25. Gezgin, T., Etzien, C., Henkler, S., & Rettberg, A. (2012, April). Towards a rigorous modeling formalism for systems of systems. In 2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (pp. 204-211). IEEE.
26. Lane, J. A., & Bohn, T. (2013). Using SysML modeling to understand and evolve systems of systems. *Systems Engineering*, 16(1), 87-98.
27. Axelsson, J., Fröberg, J., & Eriksson, P. (2019). Architecting systems-of-systems and their constituents: A case study applying Industry 4.0 in the construction domain. *Systems Engineering*, 22(6), 455-470.
28. Cherfa, I., Sadou, S., Belloir, N., Fleurquin, R., & Bennouar, D. (2018, June). Involving the application domain expert in the construction of systems of systems. In 2018 13th Annual Conference on System of Systems Engineering (SoSE) (pp. 335-342). IEEE.



29. Vargas, I. G., Gottardi, T., & Braga, R. T. V. (2018, September). An approach to integrate systems towards a directed system-of-systems. In *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings* (pp. 1-7).
30. Braga, R. T. V., Vargas, I. G., & Gottardi, T. (2016). A service-based architecture for virtual and collaborative system of systems. In *X Workshop em Desenvolvimento Distribuido de Software, Ecosystemas de Software e Sistemas-de-Sistemas (WDES), CBSoft Proceedings* (pp. 51-60).
31. Kaur, N., McLeod, C. S., Jain, A., Harrison, R., Ahmad, B., Colombo, A. W., & Delsing, J. (2013, February). Design and simulation of a SOA-based system of systems for automation in the residential sector. In *2013 IEEE International Conference on Industrial Technology (ICIT)* (pp. 1976-1981). IEEE.
32. Osmundson, J. S., Huynh, T. V., & Shaw, P. (2006). Developing Ontologies for Interoperability of Systems of Systems. In *Conference on Systems Engineering Research*.
33. Franzén, L. K., Staack, I., Jouannet, C., & Krus, P. (2019, October). An Ontological Approach to System of Systems Engineering in Product Development. In *FT2019. Proceedings of the 10th Aerospace Technology Congress, October 8-9, 2019, Stockholm, Sweden (No. 162, pp. 35-44)*. Linköping University Electronic Press.
34. Benali, H., Saoud, N. B. B., & Ahmed, M. B. (2014, November). Context-based ontology to describe system-of-systems interoperability. In *2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)* (pp. 64-71). IEEE.
35. Yang, L., Cormican, K., & Yu, M. (2019). Ontology-based systems engineering: A state-of-the-art review. *Computers in Industry*, 111, 148-171.
36. Ormrod, D., Turnbull, B., & O'Sullivan, K. (2015, December). System of systems cyber effects simulation ontology. In *2015 Winter Simulation Conference (WSC)* (pp. 2475-2486). IEEE.
37. Seghiri, A., Belala, F., Benzadri, Z., & Hameurlain, N. (2018, June). A maude based specification for sos architecture. In *2018 13th Annual Conference on System of Systems Engineering (SoSE)* (pp. 45-52). IEEE.
38. Oquendo, F. (2016, June). Formally describing the software architecture of systems-of-systems with SosADL. In *2016 11th system of systems engineering conference (SoSE)* (pp. 1-6). IEEE.
39. Oquendo, F. (2016, November). Formally describing the architectural behavior of software-intensive systems-of-systems with SosADL. In *2016 21st International Conference on Engineering of Complex Computer Systems (ICECCS)* (pp. 13-22). IEEE.
40. Oquendo, F. (2016, June). Pi-Calculus for SoS: A foundation for formally describing software-intensive systems-of-systems. In *2016 11th System of Systems Engineering Conference (SoSE)* (pp. 1-6). IEEE.
41. Gassara, A., Rodriguez, I. B., Jmaiel, M., & Drira, K. (2017). A bigraphical multi-scale modeling methodology for system of systems. *Computers & Electrical Engineering*, 58, 113-125.
42. Stary, C., & Wachholder, D. (2016). System-of-systems support—A bigraph approach to interoperability and emergent behavior. *Data & Knowledge Engineering*, 105, 155-172.
43. Wachholder, D., & Stary, C. (2014, October). Bigraph-ensured interoperability for system (-of-systems) emergence. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 241-254). Springer, Berlin, Heidelberg.

44. Gassara, A., Bouassida, I., & Jmaiel, M. (2017, April). A tool for modeling sos architectures using bigraphs. In *Proceedings of the Symposium on Applied Computing* (pp. 1787-1792).
45. Rao, M., Ramakrishnan, S., & Dagli, C. (2008). Modeling and simulation of net centric system of systems using systems modeling language and colored Petri-nets: A demonstration using the global earth observation system of systems. *Systems Engineering*, 11(3), 203-220.
46. Baek, Y. M., Song, J., Shin, Y. J., Park, S., & Bae, D. H. (2018, May). A meta-model for representing system-of-systems ontologies. In *2018 IEEE/ACM 6th International Workshop on Software Engineering for Systems-of-Systems (SESoS)* (pp. 1-7). IEEE.
47. Zhang, Y., Liu, X., Wang, Z., & Chen, L. (2012). A Service-Oriented Method for System-of-Systems Requirements Analysis and Architecture Design. *JSW*, 7(2), 358-365.
48. Hu, J., Huang, L., Chang, X., & Cao, B. (2014, March). A model driven service engineering approach to system of systems. In *2014 IEEE International Systems Conference Proceedings* (pp. 136-145). IEEE.
49. Wang, R., Agarwal, S., & Dagli, C. H. (2015, April). OPM & color petri nets based executable system of systems architecting: A building block in FILA-SoS. In *2015 Annual IEEE Systems Conference (SysCon) Proceedings* (pp. 554-561). IEEE.
50. Chaabane, M., Rodriguez, I. B., Colomo-Palacios, R., Gaaloul, W., & Jmaiel, M. (2019). A modeling approach for Systems-of-Systems by adapting ISO/IEC/IEEE 42010 Standard evaluated by Goal-Question-Metric. *Science of Computer Programming*, 184, 102305.