System of Systems Modelling: Recent work Review and a Path Forward

Charaf Eddine Dridi, Belala Faiza

System of Systems Modelling: Recent work Review and a Path Forward

1st Charaf Eddine DRIDI, 2nd Zakaria BENZADRI, 3rd Faiza BELALA
Constantine2-Abdelhamid Mehri University
LIRE Laboratory
Constantine, Algeria
{charafeddine.dridi, zakaria.benzadri, faiza.belala}@univ-constantine2.dz

***Abstract*—Systems-of-Systems (SoSs) stand out from monolithic systems, because of their composed nature, their large scale, their decentralized control mechanism, their evolving environments, and their large number of stakeholders. Due to the varied methodologies and domains of applications in existing literature, there does not exist a single unified consensus for processes involved in System-of-Systems Engineering (SoSE). The purpose of this article is to provide a cursory description of the SoS basic concepts on the one hand, and then to analyse the main challenges in its development. Finally, we report the literature review showing various techniques and methods that have been modified from the conventional systems engineering to better fit the needs of SoSs design. We hope the findings of this work may encourage and inform the community researchers of the creation of a more holistic and unified engineering process that is tailored for the demands of these large-scale systems. Thus, the complexity of the SoS development lends itself nicely to a Model-Based Systems Engineering (MBSE) which provides communication and verification that transcends the levels of development. MBSE uses a model or set of models to document and communicate from the system requirements level down to the software implementation level.**

***Index Terms*—Literature Review, System-of-Systems, SoS Engineering, Model-Based Systems Engineering.**

I. Introduction

In the last decades, large-scale systems (LSS) and more recently SoSs appeared as new software technologies that integrate a set of various Constituent System (CSs) from different subfields, and which offer a reliable and more natural alternative to build very LSSs, thus giving some solutions to the current research community issues. The subsystems were not designed to work together, each CS may be of different age and technology. This is one of the many reasons that make the SoSE process quite different from that of the traditional System Engineering (SE) in which software systems can only be implemented from scratch. SoSE not only requires to focus on system specification and verification but also requires additional consideration for overall SoS context, individual CSs characteristics, issues solving and integration process.

The term SoSE process refers to the application of engineering principles to SoS development. It consists of activities for managing the creation of SoS, including identifying and selecting CSs based on stakeholder requirements analysis, designing conceptual architectures, integrating and assembling the selected CSs and updating the SoS as CSs evolve with newer versions over time. SoSE can also significantly reduce development cost, time and improve efficiency, reliability and overall quality of SoS. The literature review shows various methods that have been modified from the conventional systems development to better fit the needs of SoS design. However, each of these methods favors one aspect of SoS over another. To our knowledge, there is so far no consensus to specify the engineering process for this type of LSS. Hence, developing a unified approach to model different aspects of SoS is a very critical research challenge and has aroused great interest in academia and industrial researchers.

The main objective of this work is threefold: (1) provide a cursory description of the SoS basic concepts and analyze the main challenges in its development; (2) report the literature review showing various techniques that have been modified from the conventional SE to better fit the needs of SoS design; and (3) knowing that traditional SE seeks to optimize an individual CS, whereas SoSE seeks to optimize the network of the various interacting and new systems, brought together to satisfy multiple objectives, we propose an effective SoSE methodology allowing decision-makers to design informed architectural solutions for the most SoSs challenges. In such a complex scenario, this paper survey is directed to those who want to approach this complex discipline and contribute to its development.

The remainder of this paper is as follows. section II presents the state-of-the-art in SoS and its main application domains. In section III, challenges for SoS design are analyzed and classified. Then in section IV, several SoS modelling approaches are reviewed, comparing their methods and tools. We analyze in section V the reasons behind the lack of acceptance in these methodologies, as well as the features that could have considered for better acceptance. We conclude in section VI, suggesting new directions for SoSs Engineering.

II. SoS BASICS: definition and application domains

The term SoS is mainly used to describe an integrated force package of interoperable systems acting as a single system to achieve a mission capability [1]. Existing literature offers a rich set of definitions of SoS and their characteristics, among them, [2] defines a SoS as a set of distributed and complex CSs interacting in a network structure, whose CSs are physically

and functionally heterogeneous, and perform a unified capability that contributes to the system function. Moreover [3], defines five main characteristics of SoS: operational/managerial independence, geographic distribution, emergent behavior, and evolutionary development (see Table I).

TABLE I
SOSS MAIN CHARACTERISTICS.

	Description
Operational independence	Every CS works independently to achieve its own individual goals and collaborating with the other CSs to accomplish the SoS global goal.
Managerial independence	Every CS belongs to a specific company and it can be managed independently by the company to which it belongs.
Geographic distribution	CSs are dispersed, i.e. that they can exchange only information with one another and not substantial quantities of mass or energy.
Emergent behaviour	SoS global behaviours are emergent properties of the entire SoS and cannot be localized to any CS.
Evolutionary Evolutionary	The SoS structures, functions and purposes are subject to several requirements, functionalities and evolutions of its CSs.

Relying on the governance and management complexity, the first categorization for SoSs was proposed by Maier in 1990's specifying three SoS types: Virtual, Collaborative, and Directed, a fourth SoS type Acknowledged was identified in [4].The following characteristics of CSs are the main distinguishing characteristics between SoSs and other types of systems.

A system that does not have these five characteristics is therefore not considered to be a SoS. Every SoS can be recognized, treated and classified in accordance with one of the four SoS types. In addition, 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 (see Table II).

III. SOS CHALLENGES

In this section, we introduce the most relevant challenges of SoSs (see Fig. 1).

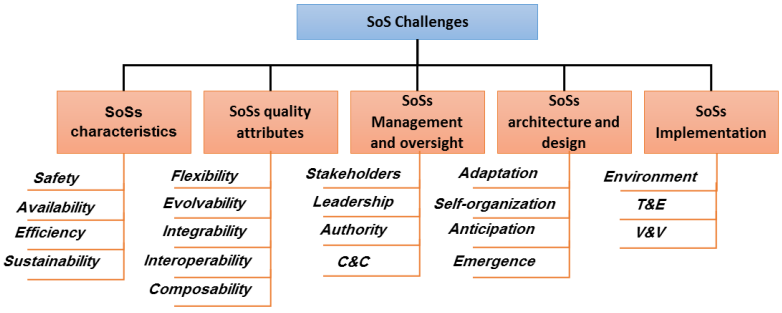

Fig. 1. SoSs Challenges.

A. Challenges behind SoSs characteristics

In general, the complexity of the SoSs characteristics increases with the:

- **Safety:** This property refers to the ability of a SoS to not cause any harm, hazard or risk inside or outside of its. The authors of [5] have indicated that safety is a very complex challenge that can be derived to ten major challenges.
- **Availability:** We can define SoS availability as the property of a SoS and its CSs to be continuously operational even when faults occur.
- **Efficiency:** Refers to a peak level of performance that uses the least amount of resources to achieve the highest number of specified functionalities.
- **Sustainability:** Is a key attribute of robustness that enables a SoS to continue to operate effectively throughout its mission cycle.

B. Challenges in SoSs quality attributes

Q.A are defined as the supplement purpose of SoSs:

- **Flexibility:** Defined as the degree of ease of effecting change(s) to the SoS, in response to external or internal changes, in order to maintain its mission [6].
- **Evolvability:** Refers to the capability of a SoS to incorporate the required design changes to meet new requirements that arise over time [17].
- **Integrability:** Refers to the ability of different CSs parts, linked in a one larger SoS, to significantly communicate with each other.
- **Interoperability:** Refers to the ability of the SoS and its CSs to work together, exchange information and interpret this information to provide specified capabilities [10].
- **Composability:** A highly composable SoS means that it may easily and systematically be combined with other CSs.

C. Management and oversight challenges

This task ensures that stakeholders, leaders, and authorities are used for the intended purposes:

- **Stockholders heterogeneity:** Stakeholders are inherently heterogeneous due to multiple users, their viewpoints, business processes, platforms, environments…etc. [18] [19].
- **Leadership:** Multiple independent stakeholders with their heterogeneity require strong leadership to identify and assess technical and development and application options [19].
- **SoS authorities:** Lack of a single cross-cutting authority in SoS remains a major challenge.
- **Certification and accreditation:** C&A processes in SoS are given by dynamic standards and protocols that are defined by a series of tests, guidelines, policies…etc.

TABLE II
CATEGORIES OF SOSS, THEIR LEVEL OF CENTRAL AUTHORITY, AND THEIR APPLICATION DOMAINS [7] [8] [9].

SoS	Definition	Authority	Application domain
Virtual	This SoS emerges from the interaction between CSs, whereas the objectives are unknown. CSs are independently managed in a distributed and uncoordinated environment where mechanisms to maintain the whole SoS are not evident.	Non existent	-Internet. -Automated high-speed algorithmic trading. -National economies.
Collaborative	The CSs collaborate to fulfil the central objectives. CSs voluntarily collaborate more or less to address shared or common interests.	offers standards to enable the collaboration of CSs.	-Regional area crisis response system [1]. -Public Transport Information [8] [11] [12] [13]. -The Global Financial . -Intelligent Transport . -Internet Engineering Task Force.
Acknowledged	Goals, management, resources, and central authority of the SoS are all recognized, but the CSs still retain their independent management.	is based upon negotiation between CSs and the SoS as a whole.	-Smart City [8] [14]. -NATO Alliance. -SESAR-Single European Sky (EU).
Directed	The system is built for a specific purpose. CSs can have their operational and managerial independence, but their behaviour is subordinated to a central authority and its purposes.	its specific main purposes are evident and drive CSs.	-Health care SoS [8] [15] . -Mars Science Laboratory (MSL). -Military Command and Control [16] [17]. -NexGen – US Air Traffic Management. -Army's Future Combat systems in the US DoD.

D. SoSs architecture and design challenges

Several challenges related to the architecture and the design of SoSs are identified:

- **Dynamic adaptation:** SoS/CSs dynamicity is a major complexity in the run-time of a SoS and it may be caused by many reasons.
- **Self-organization:** It forms the collaboration between the CSs which adapt their internal structure in response to external circumstances.
- **Anticipation needs:** For an effective performance, we need reliable new design and modelling approaches which can be formulated only on the basis of accurate forecasts.
- **Emergence:** It is impossible to understand SoS without thinking that simple CSs in one way or another will give complex behaviours.

E. Implementation challenges

Here we will review some of the most important SoS implementation challenges:

- **Environment changes:** CSs should not be parts of the SoS but a change in any of them can produce a change in the state of SoS [20].
- **Test and evaluation:** Full SoS level testing can be costly and it can be very difficult to create test environments that realistically represent the expected results [19] [21].
- **Verification and Validation:** Verification when applied to the SoS, have issues creating an agreement on the standard. Validation recognizes the essentially subjective nature of the evolutionary goals.

F. Synthesis

SoSE considered as a set of developing process, tools, and methods for designing, re-designing and deploying SoSs, has new characteristics and challenges that particularly distinguish them from other systems in several aspects, by requiring: a high degree of strength of SoSs inherent characteristics, a need for certain QA and other derived characteristics to meet certain non-functional requirements, a need for efficiency and cost-effective architecture and design, especially for SoS dynamicity and evolution, and also a need to introduce a new implementation solution taking into account other related challenges.

These two last requirements are what prompted the software engineering community to explore the need for a comprehensive methodology to analyze, design and build in the engineering process of LSS. Indeed, their complexity lends itself nicely to a model-centric approach, especially a model that can represent the independence of the CSs that comprise the SoS.

Thus, MBSE constitutes a promising approach as it provides communication and verification that transcends the levels of development. It uses a set of models to document and communicate from the requirements level down to the implementation level. The models are connected and dependent on each other so that, changes in one model automatically require the update of the set of models.

IV. STATE OF RESEARCH IN MODELLING SOSS

We present in this section a review and analysis of the modelling approaches that address SoS architectural modelling. Within the area of academic modelling and simulation, several studies have been carried out in the context of modelling SoSs. We classify the approaches found in the literature into seven main classes: MDA, MD, SOA, Ontology, ADL, Bigraph and Hybrid.

A. (Model-Driven Architecture) MDA-based approaches

MDA-1. The aim of [22] was to define an abstract view with all the possible information in the configuration and deployment processes. A meta-model which represents a number of possible configurations was also produced.

MDA-2. The authors of [23] have adopted a MDE approach to define a DSML that was used to model SoS security architectures.

MDA-3. The authors of [24] have defined a SoS profile that extends on the SysML reference meta-model with specific language constructs. They have also introduced an extension of this work in [25].

MDA-4. In our previous paper [26] we have provided a MDA method that simplifies SoSs complexity by increasing their abstraction level.

B. (Model-Driven) MD-based approaches

MD-1. Authors of [27] have proposed a formalism for relating basics SoS concepts by means of a UML class diagram. They have identified a set of basic concepts to describe a modelling approach for distributed collaborative SoSs.

MD-2. The goal of [28] was to show how SysML models can be used to support a set of needs that are essential for a SoS.

MD-3. In [29], the authors have investigated through a case study in the construction domain the interplay between SoS and CS architectures.

MD-4. The paper [30] has provided an approach to support design activities in the SoS development process.

C. (Services-Oriented Architecture) SOA-based approaches

SOA-1. The authors of [31] have proposed an approach to assist the SE community during the integration among CSs of a SoS and to use as a basis for the composition of Directed SoS.

SOA-2. The authors of [32] , have proposed a service-based architecture, which they named MV-SoSA, that serves as a basis when composing new Mixed-type SoSs.

SOA-3. The authors of [33] have realized a modular reconfigurable SoS based on a platform of reusable distributed CSs integrated within a SOA.

D. Ontology-based approaches

Ont-1. The authors of [34], 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.

Ont-2. The aim of this paper [35], 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.

Ont-3. The authors [36] have proposed an approach to build a SoS conceptual model and a foundational ontology adapted from DOLCE to depict SoS interoperability context [37].

Ont-4. The authors of [38] have proposed a SoS cyber effects ontology that outlines the requirements for a series of ontologies necessary to model the SoS effects of cyber-attacks.

E. (Architecture Description Language) ADL-based approaches

ADL-1. This approach [39] suggests a Maude-based formal and executable model where communications and relationships architecture are well defined.

ADL-2a. The authors of [40] have presented SosADL, an ADL based on a π-Calculus with Concurrent Constraints specially designed for describing SoS architectures.

ADL-2b. The extended work [41] enables the description of evolutionary architectures, which maintain emergent behaviour supporting dynamic reconfigurations.

ADL-2c. And in [42], they have focused on the description of SoS architecture to support automated verification.

F. Bigraph-based approaches

Big-1. In [43], the authors have proposed a novel methodology based on the formal technique of BRS with an inspiring vision from multi-scale modelling.

Big-2. The authors of [44] and [45] have demonstrated how bigraph-based approaches can engage with SoS through abstract relationships that allow for dynamic interaction.

Big-3. In [46], 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 modelling and simulation of SoS.

G. Hybrid approaches

Hyb-1. In [47] where the authors have exploited different models and in particular executable models from SysML specifications.

Hyb-2. This work [48] has focused on developing a conceptual meta-model called M2SoS that represents SoS ontologies.

Hyb-3. The authors of [49] have presented a hybrid modelling method based on service-oriented and ontology-based modelling.

Hyb-4. The authors of [50], have presented a MDA for service-oriented SoS architecting, modelling and simulation.

Hyb-5. The authors of [51], have used a hybrid approach with both Colored Petri Nets and Object Process Method modelling languages to create executable architecture models for SoSs.

H. Comparison

It is important in this regard to review and compare the previously mentioned studies and see to what extent they can encourage better consideration of the SoSs top-down development and conduct their different engineering processes (see Table III). Therefore, various processes are involved in the development lifecycle of SoSs. The processes refer to activities that can guide SoSs development from the system requirements level down to the software implementation level, and naturally, by coordinating the various processes for the development of a project.

In fact, SoSE adopts a structured, main three-process engineering to develop projects from Analysis through Implementation that permits releasing an efficiently finished SoS that

TABLE III
CHARACTERISTIC TABLE OF THE STUDIED APPROACHES.

Ref/Pub	Formalism / language	Static modelling			Dynamic modelling				Integ / deploy	Application domain
		Enti-ties	Media-tors	Organi-zation	Evol-ution	Intera-ctions	Beha-viour	Reconf iguration		
MDA-1: 2012	MDA	CS,S	ports	-	-	1	-/+	-	-/+	
MDA-2: 2016	MDA	CS,G	ports	-	-	-	-	-	-	Smart Campus
MDA-3: 2016	SysML Profile	CS,G	interface	-	-/+	1	-/+	-/+	-	Smart Grid
MDA-4: 2020	MDA	CS,R,G,C	roles	hierarchy	-	1 to 5	-	-	-	Aircraft Emergency
MD-1: 2012	UML	CS,R,S,G	ports	composition	-/+	1,2,3	-/+	-/+	-/+	Fire Fighting
MD-2: 2012	SySML	C,S	Interface	-	+	1	-/+	-	-	Crisis Response
MD-3: 2019	UML	C,M	-/+	hierarchy	-	1	-	-		industry 4.0
MD-4: 2018	SysML	CS ,R,M	-	-	-	3,4	-	-	-	Crowd Management
SOA-1: 2018	SOA	CS,S,G	interface	-	-/+	1	-/+	-	-/+	
SOA-2: 2016	SOA	CS,R,S	interface	-	-	1	-/+	-		
SOA-3: 2013	PDE		services	-	-	-/+	-/+	-	-	Residential sector
Ont-1: 2006	UML	CS,M,SC	-/+	-	-	1	-	-	-/+	Maritime Protection
Ont-2: 2019	OWL	CS,C	-	-/+	-	1	-	-	-	Search and Rescue
Ont-3: 2014	ODPs	CS	-	-	-	1	-	-	-	Traffic Management
Ont-4: 2015		CS	-	-/+	-	1	-	-	-	Defense
ADL-1: 2018	Maude	CS,R	-/+	hierarchy	-/+	1,3	-/+			Maritime Transport
ADL-2a: 2016	π-calculus	CS	-/+	-	-/+	1	-/+	-/+	-	Flood Monitoring and Emergency Response
ADL-2b: 2016	π-calculus	CS	-/+	-	-/+	1	+	-/+	-	
ADL-2c: 2016	π-calculus	CS	-/+	-	+	1	-/+	-/+	-	
Big-1: 2017		CS	-		-	1	-	+	-	Smart Buildings
Big-2: 2014		CS	ports	-	-	1	+	-	-	e-learning manag
Big-3: 2017	BRS	CS	ports	-/+	-	1	+	-	-	Cargo Information
Hyb-1: 2017	SysML+CPN	-/+	-/+	-/+	-/+	1	-/+	-/+		Observation
Hyb-2: 2018	MDA+Onto	CS,G,C,S	-/+	-/+	-	1,2	-/+	-	-/+	Mass Casualty Incid
Hyb-3: 2012	SOA+Onto	CS, C, S	-	-/+	-	1	-	-	-	Military
Hyb-4: 2014	MDA+SOA+DEVS	CS,C,S,M	Services	composition	-	1	-	-	-	Airport
Hyb-5: 2015	CPN+OPN	CS,C	-	-	-	1	+	-	-/+	FILA

+ : aspect taken into account; +/- : relatively considered aspect; - : aspect not taken into account; CS: Constituent-System; R: Role; C: Capability; S: Service; G: Goal; M: Mission; 1:SoS-CS relationship; 2:globalGoal-subGoal relationship; 3:Role-Role relationship; 4:Capability-Role relationship; 5:Role-Goal relationship.

satisfies stakeholders and performs as required. In this section, we present a comparative study according to the main three SoSE processes as follow:

- **Conceptual Analysis:** addresses high-Level SoS Requirements, distinguishes relationships and analyses mission capability assessment.
- **Architectural Design:** refers to both static (Entities, Mediators and Organization) and dynamic (Evolution, Interactions, Behaviour and Reconfiguration) architectural aspects.
- **Integration and Implementation:** investigate the Integration, configuration and deployment of SoS.

I. Synthesis

Several attempts to deal with the SoSE process and design the SoS architectural description have been published over the years, in order to establish a generic and reusable solution for elaborating SoSs architectures and verify their properties. The majority of the approaches have some advantages and disadvantages .i.e. all of them are only limited to dealing with some SoS concepts, not to mention a complete SoSE process or the overall SoSE processes. To the best of our knowledge, almost all the cited approaches do not offer support to deal with the Conceptual Analysis process of SoS system. Namely, by taking into account the high-Level SoS requirements, Understanding the CSs and their relationships and interdependencies, effective mission capability, etc.

About the last two remaining operations, the comparison study examines different techniques that can be applied to SoSs design and implementation approaches. These techniques come from a wide range of backgrounds, ranging from SoSs aspects and principles to formal methods, conceptual models, hybrid methods, etc. These techniques face a number of different challenges:

• Some focus on describing the SoS as a whole, addressing structural organizations, ignoring how the CSs interoperate.

• Some handle complexity with numerous heterogeneous CSs and interactions in terms of interfaces or services.

• Others express the SoS at different levels of abstraction which is broad enough to cover the different aspects of SoS.

• And in the same context, other techniques express the SoS at different levels of abstraction and provide links between these levels.

• More still target the reasoning focusing less on detecting CSs behaviour and more on how the goals and requirements change at runtime.

• Clarify how viewpoints can express different parts of SoS characteristics and map these viewpoints to each other.

• Others determine how the combination of some techniques can be used to improve the specification and analysis of the static and dynamic aspects of SoS.

V. ENHANCED APPROACH FOR MODEL-BASED SOS ENGINEERING

SoS development does not follow the normal system development process. SoS capability is based on the contributions of the individual CSs. This interdependence between the SoS and the CSs makes a document-centric development impractical as an exorbitant effort is required to maintain [52]. In such a complex scenario, three important activities are involved in the SoS development process (see Fig. 2):

- Conceptual design and CSs selection.
- Architectural design.
- Integration and deployment.

In this section, we take a first step towards achieving a unified MBSE methodology, allowing decision-makers to design informed architectural solutions for the SoSs challenges. In a SoS, the systems contributing to the SoS objectives are typically in place when the SoS is established, and the SoS systems engineer needs to consider the current state and plans of the individual systems as important factors in developing an architecture for the SoS.

We aim to solve some of the aforementioned issues and challenges by giving a set of models to document and communicate from the system requirements level down to the software implementation level. At each stage of the SoSE process (see Fig. 2), we propose a set of specific models that are connected

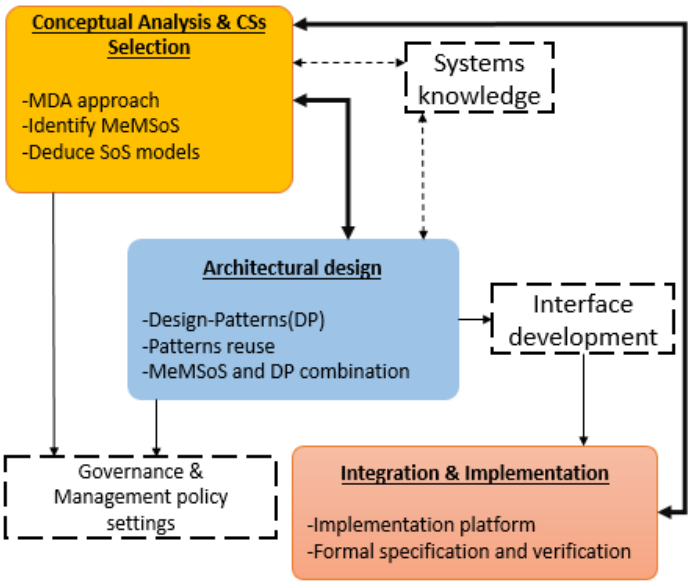

Fig. 2. SoS Engineering process.

and dependent on each other so that changes in one model automatically require the update of the set of models. This interdependence among the models provides the extra level of verification.

A. Conceptual Analysis

In the context of our ongoing work, we have envisaged that the conceptual analysis requires an advanced methodology that solves the analysis-related problems and at the same time bridges the gap between SoS level requirements engineering and SoS architectural design.

In [26], we have introduced the first step forward toward MBSE methodology via an MDA technique. We have only targeted the SoS architectural analysis phase where a method to obtain a high-level designing and reasoning of SoS analysis has been defined. We have presented in this part the MeMSoS(Fig. 3), an abstract meta-model to deduce models that allow the graphical and ambiguous specification.

This abstract and generic meta-model defines a logical structure of all elements involved in any SoS (for example: CSs, Roles, Capabilities, Goals, etc.), and permits to describe all the SoS features at a same high-level of abstraction (for example: components' Hierarchical and Goals organizations, CSs, relationships, Roles interactions, CSs interdependencies, etc.).

Thereby, in the SoSE context, we have found that the MDA methodology can be leveraged to distinguish SoSs elements, analyzing the conceptions they refer to and abstract their conceptual analysis-related problems, and with the end of this work, the adoption of Meta-Modelling techniques to design SoSs architectures was an intrinsic step in the MDA approach providing a common, unambiguous, structured and accurate SoS architectural analysis that will help different SoSs

stakeholders to understand a certain degree of SoS challenges and its potential solutions at the architectural design activity in both further research and prototype development.

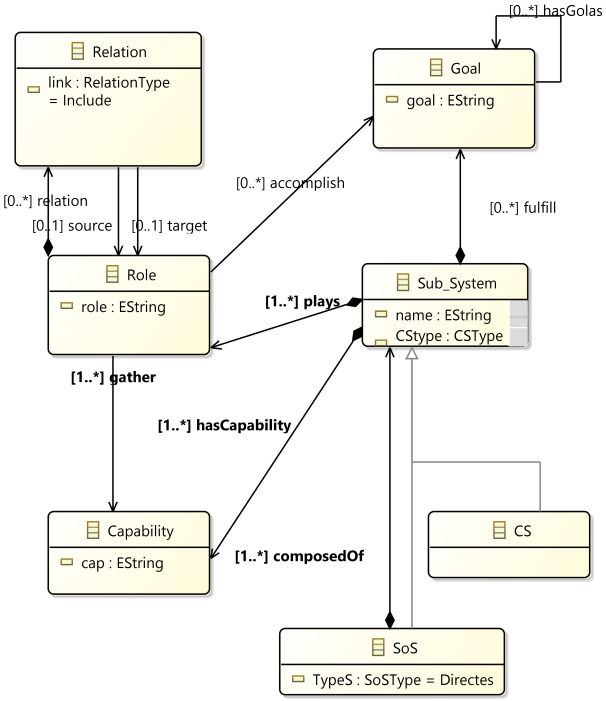

Fig. 3. An extract of MeMSoS.

B. Architectural Design

According to the study of various previous works, the concept of a design pattern, as a set of proven solutions to a recurring problem within a context, has not yet been addressed. Reusing patterns, combined with the deduced models of the first step, therefore, yields better-quality software within reduced time frames.

To this end, we propose to apply a design pattern-based methodology as an architectural framework for modelling SoSs in a modular way. Therefore, organizational patterns are developed to increase the value of specific organizational principles and structures for certain classes of software. In this context, a design pattern formalizes knowledge and experience in the SoSE area. Its primary purpose is to simplify SoS software architecture design and reuse, capturing and exploiting the knowledge used to design a SoS.

The design patterns-based approach seems the most suitable to meet SoSs different challenges and to model both structural and behavioral aspects of SoSs. Structurally, the goal is to design a general structural pattern that allows defining the architectural elements of a SoS, their CSs, spatial distribution and interactions between its different components. Dynamically, it will show via a modular architecture its ability to model the dynamic aspects of SoSs, in particular, the auto-adaptive behaviors of CSs and their evolutionary development.

C. Integration and Deployment

Integration, configuration and deployment of SoS is often a complex task because this system is always distributed on different geographic areas, composed of hundreds of components, running under multiple hardware constraints, on different resources, and subject to mission-critical requirements. In this step, we plan to offer a complete tool support implementation platform dedicated to design static and dynamic architectures.

This platform will allow presenting the result of integrating the hybrid modelling approaches and to validate them by presenting a complete formal environment for modelling SoSs and verifying their behavior.

As a result, it will open a window to several formal specification and verification formalisms in order to provide generic models that will be dedicated to automatically run and formally check the SoS models. This will offer more formal verification of various related properties and avoid ambiguities limiting their correct usage in application support tools.

VI. CONCLUSION

This paper aimed to provide a state of the art of work carried out on the modelling of different aspects in SoSs. We first discussed the definition of SoSs, their main characteristics, prevalent types, practical applications and some common SoSs challenges.

We conducted a literature review based on a set of research studies in the domain of SoSE and the contribution of each concerning the main SoSE processes. These studies agree on the fact that the development of such systems must allow joint engineering of SoS specifications and must take into account the most development-related challenges. However, and as it was expected to make an important impact on our future research works, there are still several key points to clarify and formalize which allow us to position our incoming path.

While keeping in mind that our work is generalist, modular and reusable. This study allowed us, on the one hand, to better understand the difficulties linked to the modelling of SoSs, and on the other hand, to choose the methodology of MBSE which seems to us to be the best suited to the description of such systems allowing decision-makers to design informed architectural solutions for the most SoSs challenges.

REFERENCES

[1] Leading Edge magazine, Naval Surface Warfare Center, Dahlgren Division, Dahlgren,Virginia, Feb.2013
[2] Kotov, V. (1997). Systems of systems as communicating structures (Vol. 119). HP Labs.
[3] 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.
[4] Dahmann, J. S., & Baldwin, K. J. (2008, April). Understanding the current state of US defense systems of systems and the implications for systems engineering. In 2008 2nd Annual IEEE Systems Conference (pp. 1-7). IEEE.
[5] Harvey, C., & Stanton, N. A. (2014). Safety in System-of-Systems: Ten key challenges. Safety science, 70, 358-366.
[6] Chin, K. S., Yau, P. E., Wah, S. K., & Khiang, P. C. (2013). Framework for managing System-of-systems ilities. DSTA HORIZONS, 14.

[7] Ncube, C., & Lim, S. L. (2018, August). On systems of systems engineering: A Requirements engineering perspective and research agenda. In 2018 IEEE 26th International Requirements Engineering Conference (RE) (pp. 112-123). IEEE.

[8] Assaad, M. A., Talj, R., & Charara, A. (2016, July). A view on Systems of Systems (SoS).

[9] Thesis: Support à la conception architecturale de systèmes-de- systèmes reconnus à logiciel prépondérant

[10] 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.

[11] DeLaurentis, D. (2005, January). Understanding transportation as a system-of-systems design problem. In 43rd AIAA Aerospace Sciences Meeting and Exhibit (p. 123).

[12] 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.

[13] Jamshidi, M. O. (2008). System of systems engineering-New challenges for the 21st century. IEEE Aerospace and Electronic Systems Magazine.

[14] Aljohani, T. M. (2018). Analysis of the Smart Grid as a System of Systems. arXiv preprint arXiv:1810.11453.

[15] 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.

[16] Lane, J. A., & Epstein, D. (2013). What is a System of Systems and why should I care?. University of Southern California.

[17] Dahmann, J. (2015). Systems of systems characterization and types. NATO Lecture Series on Systems of Systems Engineering, 1-14.

[18] Cavalcante, E., Cacho, N., Lopes, F., Batista, T., & Oquendo, F. (2016, December). Thinking smart cities as systems-of-systems: A perspective study. In Proceedings of the 2nd International Workshop on Smart (pp. 1-4).

[19] Dahmann, J. (2014, July). 1.4. 3 system of systems pain points. In INCOSE International Symposium (Vol. 24, No. 1, pp. 108-121).

[20] Ackoff, R. L. (1971). Towards a system of systems concepts. Management science, 17(11), 661-671.

[21] Dahmann, J., Lane, J. A., Rebovich, G., & Lowry, R. (2010, June). Systems of systems test and evaluation challenges. In 2010 5th International Conference on System of Systems Engineering (pp. 1-6). IEEE.

[22] Barbi, E., Cantone, G., Falessi, D., Morciano, F., Rizzuto, M., Sabbatino, V., & Scarrone, S. (2012). A model-driven approach for configuring and deploying systems of systems. 2012 7th International Conference on System of Systems Engineering (SoSE).

[23] El Hachem, J., Pang, Z. Y., Chiprianov, V., Babar, A., & Aniorte, P. (2016). Model driven software security architecture of systems-of-systems. 2016 23rd Asia-Pacific Software Engineering Conference (APSEC).

[24] Mori, M., Ceccarelli, A., Lollini, P., Bondavalli, A., & Fromel, B. (2016). A holistic viewpoint-based SysML profile to design systems-of-systems. 2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE). https://doi.org/10.1109/hase.2016.21

[25] Mori, M., Ceccarelli, A., Lollini, P., Frömel, B., Brancati, F., & Bondavalli, A. (2017). Systems-of-systems modelling using a comprehensive viewpoint-based SysML profile. Journal of Software: Evolution and Process, 30(3), e1878.

[26] 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). IEEE.

[27] Gezgin, T., Etzien, C., Henkler, S., & Rettberg, A. (2012). Towards a rigorous modelling formalism for systems of systems. 2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops.

[28] Lane, J. A., & Bohn, T. (2012). Using SysML modelling to understand and evolve systems of systems. Systems Engineering, 16(1), 87-98.

[29] 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.

[30] Cherfa, I., Sadou, S., Belloir, N., Fleurquin, R., & Bennouar, D. (2018). Involving the application domain expert in the construction of systems of systems. 2018 13th Annual Conference on System of Systems Engineering (SoSE).

[31] Vargas, I. G., Gottardi, T., & Braga, R. T. (2018). An approach to integrate systems towards a directed system-of-systems. Proceedings of the 12th European Conference on Software Architecture Companion Proceedings - ECSA '18.

[32] Braga, R. T. V., Vargas, I. G., & Gottardi, T. (2016). A service-based architecture for virtual and collaborative system of systems. In Anais. Porto Alegre, RS: SBC.

[33] Kaur, N., McLeod, C., Jain, A., Harrisson, R., Ahmad, B., Colombo, A., et al. (2013). Design andsimulation of a SOA-based system of systems for automation in the residential sector. IEEE.

[34] J. Osmundson, T. Huynh, and P. Shaw, "Developing Ontologies for Interoperability of Systems of Systems",SOSECE, 2006

[35] Knöös Franzén, L., Staack, I., Jouannet, C., and Krus, P., "An Ontological Approach to System of Systems Engineering inProduct Development,"in proceedings of the 10th Aerospace Technology Congress, Swedish Society of Aeronautics andAstronautics, Stockholm, 2019, pp. 35–44

[36] H. Benali, N. B. B. Saoud, and M. B. Ahmed, "Context-basedontology to describe system-of-systems interoperability," in2014 IEEE/ACS 11th International Conference on ComputerSystems and Applications (AICCSA). IEEE, 2014, pp. 64–71.

[37] Yang, L., Cormican, K., & Yu, M. (2019). Ontology-based systems engineering: A state-of-the-art review. Computers in Industry, 111, 148-171.

[38] 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.

[39] Seghiri, A., Belala, F., Benzadri, Z., & Hameurlain, N. (2018). A Maude based specification for sos architecture. 2018 13th Annual Conference on System of Systems Engineering (SoSE).

[40] Oquendo, F. (2016). Formally describing the software architecture of systems-of-systems with SosADL. 2016 11th System of Systems Engineering Conference (SoSE).

[41] Oquendo, F. (2016). Formally describing the architectural behavior of software-intensive systems-of-systems with SosADL. 2016 21st International Conference on Engineering of Complex Computer Systems (ICECCS).

[42] Oquendo, F. (2016). π-calculus for sos: A foundation for formally describing software-intensive systems-of-systems. 2016 11th System of Systems Engineering Conference (SoSE).

[43] Gassara, A., Bouassida Rodriguez, I., Jmaiel, M., & Drira, K. (2017). A bigraphical multi-scale modelling methodology for system of systems. Computers & Electrical Engineering, 58, 113-125.

[44] Stary, C., & Wachholder, D. (2016). System-of-systems support—A bigraph approach to interoperability and emergent behavior. Data & Knowledge Engineering, 105, 155-172.

[45] Wachholder, D., & Stary, C. (2014). Bigraph-ensured interoperability for system(-of-Systems) emergence. On the Move to Meaningful Internet Systems: OTM 2014 Workshops, 241-254.

[46] Gassara, A., Bouassida, I., & Jmaiel, M. (2017, April). A tool for modelling sos architectures using bigraphs. In Proceedings of the Symposium on Applied Computing (pp. 1787-1792).

[47] Rao, M., Ramakrishnan, S., & Dagli, C. (2008). modelling and simulation of net centric system of systems using systems modelling language and colored petri-nets: A demonstration using the global earth observation system of systems. Systems Engineering, 11(3), 203-220.

[48] Baek, Y., Song, J., Shin, Y., Park, S., & Bae, D. (2018). A metamodel for representing system-of-systems ontologies. Proceedings of the 6th International Workshop on Software Engineering for Systems-of-Systems - SESoS '18.

[49] 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.

[50] Hu, J., Huang, L., Chang, X., & Cao, B. (2014, March). A model driven service engineering approach to system of systems.In 2014 International Systems Conference Proceedings IEEE.

[51] 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.

[52] Dod, U. (2008). Systems engineering guide for systems of systems. Washington, DC, US Department of Defense, Office of the Deputy Under Secretary of Defense for Acquisition and Technology.