ORIGINAL ARTICLE

Journal Section

Systems of Systems: from Mission Definition to Architecture Description

Imane Cherfa[1,2] | Nicolas Belloir[2,3] | Salah Sadou[2] | Régis Fleurquin[2] | Djamal Bennouar[4]

[1]Université Blida1 - LRDSI, Faculté des Sciences, Algérie

[2]Université Bretagne Sud - IRISA, France

[3]CREC St-Cyr, Military Academy of St Cyr Coetquidan, France

[4]University of Bouira - LIMPAF, Algeria

Correspondence
Imane Cherfa, Université Blida1, LRDSI, Faculté des Sciences, BP 270, Route de Soumaa, Blida, Algérie
Email: imane.cherfa@univ-ubs.fr

Present address
[1]IRISA, Université de Bretagne-Sud, Campus de Tohannic, bâtiment ENSIBS, Rue Yves Mainguy, BP 573, 56017 Vannes cedex, France

Funding information

Systems of Systems (SoS) encompass a group of distributed and independent systems. This class of systems requires recurrent adaptation at runtime owing to the uncertainty and variability of the runtime environment. Thus, during their execution, SoS can deviate from the initial specification, which is often a consequence of successive evolutions. This problem occurs mainly due to (i) weak communication between the SoS analysis stage and architecture stage and (ii) the lack of links between the operational planning in the SoS analysis stage and systems that must be involved in the SoS architecture stage.

This paper proposes a model-based process that strengthens the links between the SoS analysis stage and the architecture stage in the wave life cycle. We ensure that the mission and role concepts for the SoS definition are sufficiently abstract to allow adaptation to the variability of the environment. This definition is translated into an abstract architecture that guides the choices of the system architect during the design and evolution stages. The proposed language is an adaptation of the Systems Modeling Language (SysML). Furthermore, we define a crowd management SoS to illustrate the process.

1 | INTRODUCTION

Model-based approaches [1] represent a promising path for the development and analysis of systems of systems (SoS). These approaches allow the control of the overall complexity of the SoS, clarification and documentation of its structure and behavior, and communication of these aspects to the stakeholders [2].Models can be developed at different stages during the SoS development process, expressing different points of views. Consequently, choices and decisions can be made at each stage of the development process using different models. For instance, the application domain expert makes decisions pertaining to business aspects, while the system architect makes decisions that lead to certain implementation choices. Thus, the first stakeholder is concerned with the need and the requirement stage, while the second one is concerned with the design stage. These two stages are crucial for system development as they form its base. Thus, the choices made during the design must be consistent with the decisions made during the definition of the requirements. However, the risk of loss of information is also the highest between these two stages [3], as the stakeholders concerned with these two stages often come from considerably different domains, and different domains involve risks of misunderstanding.

This problem concerns only the consistency of the information transmitted from the requirement stage to the design stage. However, in the case of SoS, whose nature inherently involves evolution [4], the consistency of the choices made for the evolution needs with the initial capability objectives must be verified. The evolutions of an SoS can occur far in time, which means that some or all of the initial stakeholders may no longer be present. In such a case, without strong links between the choices made during the design stage and the requirements defined in the previous stage, the risk that the SoS deviates from its original objectives is considerably high, which is undesirable.

To solve this problem, we propose the creation of a strong link between the SoS analysis stage and the architecture stage in the SoS wave life cycle, that is dedicated to the acknowledged SoS. The main concept is to make the SoS analysis stage closer to the architecture stage. This aspect is achieved by using a language sufficiently familiar to the application domain expert to clearly articulate her/his needs and requirements and sufficiently formal to serve as a guide and controller for the system architect during the design and evolution stages. We propose the use of our language in the mission paradigm [5, 6, 7, 8] for the definition of the SoS. The goal is that the mission must be sufficiently defined to assist the architect to determine systems that must be involved and the functions that these systems must perform. However, in the case of the definition of SoS, which operate in dynamic environments, the application domain expert cannot always predict the constituent systems that will actually exist when the SoS is launched. Reasonably, we can consider that the application domain expert knows the "types" of systems that her/his SoS needs. Therefore, during the definition of the mission by using our language, we refer to roles (abstract entities) instead of concrete systems.

Once the mission is completely defined by the application domain expert, it is translated by the architect to an abstract architecture holding invariants that can guide the choices among the possible solutions during the design and evolution of the SoS. In other words, if the architect tries to use a solution that is inconsistent with one of the application domain expert requirements, she/he will be notified of this aspect with information pertaining to the related consequences. Thus, the abstract architecture is the solid bridge between the SoS analysis stage and the architecture stage.

The remaining paper is organized as follows. In Section 2, we present the background and motivation. Section 3 discusses the state of the art and presents a conceptual model that serves as a basis for the proposed approach. Section 4 illustrates our general approach for SoS development. Section 5 describes the mission modeling, details the proposed profile for describing the capabilities and explains the architecture creation process. In Section 6, we illustrate the proposed approach through our case study, while Section 7 presents the concluding remarks and directions

for future work.

2 | BACKGROUND AND MOTIVATION

As might be envisaged in an emerging field, there does not exist unifying definition of SoS. The several definitions of SoS have their own merits, depending on their application domain [9, 10, 4]. To bind the field precisely without considering any application domain, Maier [11] characterized SoS in terms of five principal features: "operational independence, managerial independence, geographic distribution, evolutionary development and emergent behavior".

According to several authors [11, 12], the independence of the constituent systems is the main feature of SoS. Each constituent system assumes its own goals and operates independently [13]. The need to maintain autonomy while simultaneously operating within the SoS context considerably increases the complexity of an SoS and is at the heart of the SoS SE challenge [14]. As discussed by [15], when using traditional SE approaches such as Object-Oriented Systems Engineering Method (OOSEM) [16, 17] and Harmony-SE [18] systems engineers can trace system boundaries and define requirements clearly. Furthermore, the engineers can master the development environment to assure that the technical trade studies are the basis of the allocations of requirements to components. In the SoS environment, systems engineers must take into account considerations beyond the use of existing systems as the constituent systems of these SoS. They must allocate the realization details and functionalities, which may not be optimal from the point of view of the SoS. Furthermore, constituent systems must retain their independence. For these reasons, SE is highly challenging in the SoS context.

To guide the selection of System of Systems Systems Engineering (SoS SE) principles, the U.S. Department of Defense [19] categorized SoS based on their degree of centrality into "virtual, collaborative, acknowledged and directed SoS". Virtual SoS is characterized by the absence of central management and common goal. Collaborative SoS is characterized by the absence of a central authority, constituent systems of collaborative SoS interoperate more or less voluntarily to achieve the main common goals. The acknowledged SoS is under the responsibility of an organization, that sustains the SoS systems engineering while the constituent systems preserve their independent development and goals. Directed SoS are developed and supervised to meet special purposes. Constituent systems can operate independently but are controlled to fulfill the SoS goal. SoS SE is primarily concerned with acknowledged SoS.

To address SoS SE challenges, the U.S. Department of Defense published the SE guidance for SoS [19] and proposed the trapeze model. Seven core elements characterize the trapeze model and are described as follows [19, 15]: 1) "translating the SoS capability objectives into requirements" and 2) "assessing the performance pertaining to these capability objectives" as well as 3) "monitoring and assessing the external changes on the SoS". It is important for SE for SoS to 4) "understand systems that contribute to the realization of SoS objectives and their relationships" and 5) "to develop and evolve an SoS architecture". For the SE for SoS, it is crucial to 6) "address new requirements and solution options" and 7) "orchestrate upgrades to the SoS and implement the changes".

Dahmann [12] "built on the trapeze model and translated the SoS SE core elements, their interrelationships, and SoS decision making artifacts to a more familiar and intuitive wave model representation". Originally, the wave planning was introduced by Dombkins [20], and it was subsequently "applied to the SoS trapeze model to illustrate the incremental and iterative process that characterizes acknowledged SoS development" [12]. The SoS SE wave model is illustrated in Figure 1.

Six steps characterize the SoS SE wave model [12]. In the SoS SE wave model, systems engineers are actors in 1) initiating the SoS by understanding SoS goals and 2) conducting SoS analysis by taking into account several artifacts such as SoS performance measures and SoS risks and mitigations. Fundamental to SoS SE is the 3) development and

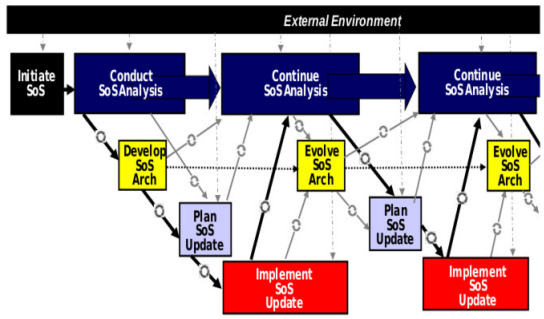

FIGURE 1 SoS SE Wave Model [12]

evaluation of the SoS architecture as well as the 4) planning of SoS updates and evaluation of the SoS priorities. The SoS SE team is involved in 5) monitoring the implementations at the constituent system level and finally 6) performing a continual SoS analysis to revisit key information.

The wave model helps improve the SoS SE design capabilities and approaches to manage the SoS problems. However, more recent research argues that in SoS SE, the design and the end to end process and management are required to be balanced [6, 5]. In fact, SoS are acquired to satisfy new capabilities in a mission context. The latter is a key element to assist SoS engineers to determine the systems that must be involved and the functions they must perform [6, 5]. It is important to consider the mission thread to bridge the dissociation between the SoS objectives and the individual functionalities undertaken by the systems that constitute the SoS to support the SoS mission. "The allocations of functions or activities to constituent systems can dynamically change over the course of a mission thread" [5]. To address these SoS SE challenges, we propose in this paper the maintenance of a mission focus throughout the SoS SE analysis and the architecture process included in the wave model.

3 | STATE OF THE ART

Our work concerns three domains of state of the art: i) SE, ii) SoS SE and iii) mission engineering (ME). In what follows, we discuss the existing work related to these three areas.

3.1 | SE approaches

A number of SE methodologies are currently used by the systems engineering community, offering guidance for analyzing, developing and documenting complex systems. "The Object-Oriented Systems Engineering Method (OOSEM) provides an integrated framework that combines object-oriented techniques, a model-based design approach and traditional top-down SE practices" [17]. OOSEM is now advocated as an example of a model-based systems engineering (MBSE) best practice since it was realigned with SysML (it was initially based on the unified modeling language (UML). the following activities are encompassed in the OOSEM "specification and design system process" [17]: (1) analysis of the stakeholder needs, and (2) analysis of the system requirements. (3) Logical architecture definition by highlighting how the logical components interact to fulfill the requirements. The (4) allocation of hardware, software, data, and procedures to the logical components. The (5) activity of optimizing and evaluating alternatives and finally,

the (6) activity of managing the traceability of requirements from the mission level requirements to the component requirements.

The Harmony [18, 21] SE process is characterized by three main activities: (1) "requirements analysis", (2) "system functional analysis", and (3) "architectural design". The Harmony SE is a model based process and uses SysML as the modeling language. In the Harmony "requirements analysis phase", requirements are grouped into use cases. The "system functional analysis" phase consists of translating functional requirements into set of system functions. A black box model is related to each use case. Incrementally, these black box models are aggregated into a black box system model. The "architectural design" activity is composed of the "system architectural design" and "subsystem architectural design" elements. In the subsequent system architectural design phase, the valid operational contracts is assigned, based on performance and safety requirements, to the physical architecture. The subsequent subsystem architectural design phase aims at deciding the operational contracts within a physical subsystem that should be implemented in the hardware and software (hardware/software tradeoff analysis).

MagicGrid [22] is a SysML-based framework for modeling complex systems. The MagicGrid framework consists of viewpoints (black box, white box and solution) and aspects (the four pillars of SysML: requirements, system structure, system behavior, and parameters) organized in a grid view. The cells of the grid represent different views of model-based systems engineering, which are described as follows [22]: (1) the requirement elicitation of stakeholders by using the SysML requirement diagram (RE); (2) a use case description of the refinements of functional stakeholder needs; (3) system context representation using the SysML internal block diagram (ibd); (4) measures of effectiveness (MoEs), which indicate the nonfunctional requirements, described in the SysML block definition diagram (BDD). The MoEs calculation procedures are specified with the SysML parametric diagrams. The (5) identification and specification of system requirements is performed by using the RE diagram, (6) and functional analysis elaboration is performed with multiple SysML activity diagrams, specifying internal system functions. The (7) logical subsystem communication identification is established using the control and resource flows defined in the functional analysis model. Both of the SysML BDD and SysML IBD are used to capture this view. The (8) measures of effectiveness (MoEs) as well as the measures of performance (MoPs) are captured for each logical subsystem, in the SysML BDD and parametric diagrams. The (9) component requirements are captured using the SysML requirement diagram. The (10) component behavior definition is performed using an association of SysML state machine, activity, and sequence diagrams; (11) component structure elaboration is performed by illustrating the physical connections between physical components, and this view is captured using both the SysML BDD and IBD. The (12) component parameter definition of each component is performed, in which each parameter captures the component characteristics and the links between them and describes how the MoEs and MoPs already specified are accomplished using these characteristics.

Other interesting SE approaches have also been reported in the literature, such as the IBM Rational Unified Process for Systems Engineering (RUP SE) [23], JPL State Analysis (SA) [24], and SYStem MODeling (SYSMOD). It is clear that the SE approaches are used to solve different tasks of the systems engineering process [17],and they have reached a level of maturity in the identification and gathering of artifacts and best practices for complex systems engineering. However, SE approaches can trace system boundaries and define requirements clearly [15], something that is not obvious in the SoS SE. Furthermore, trade analyses and measures of performance allow the optimal allocation of components to requirements while in SoS SE, SoS engineers need to take into account considerations beyond the use of existing systems as the constituent systems of these SoS [15]. They must allocate the realization details and functionalities that may not be optimal from the SoS point of view. Moreover, SoS engineers do not control the overall development environment of the SoS because the constituent systems must retain their independence.

3.2 | SoS SE contributions

In terms of the SoS SE, the Department of Defense (DoD) has published the SE Guidance for SoS [19], which provides a well-grounded practical guidance for systems of systems. According to the guide, seven main elements characterize the SoS SE (the trapeze model already described in Section 2), each element "can be mapped to the 16 technical management processes" defined in the Defense Acquisition Guidebook [25].

Dahmann [12] built on the trapeze model and proposed a translation of the trapeze model elements into the wave model rationale. The main elements of the wave model have been presented in Section 2 and Figure 1. The critical information to realize effective SoS SE, and the corresponding artifacts were identified by Dahmann [26]. As we are interested only in the analysis and architecture phases of the wave model, in the following section, we present only the artifacts related to these two stages. The artifacts are described as follows [26]: (1) The SoS capability objectives correspond to the main goals of the the SoS; (2) the SoS CONcept Of OPerationS (CONOPS) defines the use of the SoS constituent system functionality in an operational context; (3) the systems information corresponds to information pertaining to the systems that impacts the SoS capability objectives, and this information is collected to be used for replacements as the SoS evolves; (4) the SoS requirement space bounds the operational tasks and missions while considering the environment change that affect the execution of the required functions; (5) the SoS performance measures and methods capture the basis for assessing the overall performance of the SoS and for improving the SoS; (6) the effectiveness data of the SoS are collected from different environments to identify the areas needing more attention; (7) the SoS SE planning elements determine "the rhythm, technical reviews, and decision processes across the SoS evolution". These elements furnish also "the principal SE rules of engagement for the SoS and are utilized by all SoS actors"; (8) the SoS risks are captured and tracked.

The Department of Defense Architecture Framework (DoDAF) [27] and the Ministry of Defense Architecture Framework (MoDAF) [28] are respectively the architectural frameworks for the United States Department of Defense (DoD) and the UK Ministry of Defense (MOD). Both of DoDAF and MoDAF provide set of views, each of which is decomposed into products and data, for instance, operational view, capability view, and systems and services view. The operational view aims to describe the tasks and activities, operational elements, and resource flow exchanges required to conduct operations. The capability view aims to describe the mapping between the required capabilities and the activities that enable those capabilities. While the two frameworks have similar views, their respective metamodels are different. Despite this, the Object Management Group (OMG) proposed the unified profile for DoDAF and MODAF (UPDM) [29], a common modeling language for both frameworks that is based on the UML but can either be used with SysML. The UPDM prescribes more than 40 views. The viewpoints allow modeling in different levels of abstraction, and are rich in term of concepts, including all the concepts related to SE or SoS SE domains. However, the selection of a view point can be difficult, and it is difficult to take advantage of the interconnected views because none of these views provide a simplified perspective that addresses only the subsets of each view. The architectural frameworks constitute "in depth modeling approach, which requires significant resources" [30].

Previous work, such as the DANSE project [31] and the Compass project [32] have proposed a reduction in the architecture frameworks according to the target objectives. DANSE proposed that one should focus on the selected views of the UPDM instead of considering all the views. The SoS mission is described using the Operational View OV-1. Subsequently, Operational View OV-5 specifies the tasks to be involved to achieve the SoS mission. The Operational View OV-2 is used to define the data exchange within the system. The functionalities that can implement the capabilities are determined in the System View SV-5. Finally, System View SV-10A expresses the functional and nonfunctional constraints. Compass proposed to delimit SoS boudaries in early stage, while SoS modeling process must taking into account that SoS environment is open.

3.3 | Mission Engineering

Mission Engineering (ME) is an emerging field owing to the need for understanding and documenting the end to end mission execution in an SoS [5, 6]. Mission Engineering "*combines the structure of systems engineering and the tactical insights of operational planning to a system of systems to deliver a specific capability. The difficulty in ME revolves around the concept of a mission context, which manages the uncertainties, dynamics and stochastic behaviors of SoS*" [5].

In terms of software intensive SoS, Silva [33] proposed M2Arch, a model-based refinement process for SoS architectural modeling that uses missions as the basis. The mission model is defined using mKAOS [34], an SoS mission description language. In mKAOS, the mission is the specialization of a goal to the SoS domain. The mission is refined with and/or operators until sub-missions that can be handled by a constituent system are determined. The authors defined an SoS mission as encompassing five concepts: (i) priority, (ii) trigger, (iii) constraints, (iv) parameters, and (v) tasks, which are functional operations to be executed. The software architecture is generated automatically in SosADL [33], a formal language to describe SoS software architectures. However, this work did not consider the hardware and human constituent systems. Moreover, even if the mKAOS language allows the definition of an emergent behavior model, that includes "features that are produced from the interaction between constituent systems" [33], it does not consider several artifacts in SoS modeling such as effectiveness and performance measures, risks, and dynamic environment of SoS.

In the military application domain, the mission is a strong concept that is defined in a rigorous manner and generally expressed through a well-structured document. The missions and means framework (MMF) [7] is a framework for defining the DoD military mission and evaluating its utility quantitatively. The MMF consists of 11 elements used to define military operations. Seven levels specify the mission: (1) mission purpose that defines the why of the military evolution and indicates the reason and purpose of the mission, (2) context and environment that define under what circumstances a mission is to be accomplished, (3) index and location/time that define the where in terms of geographic location and the when in terms of time, (4) tasks and operations that define the "do what" of the mission, and describes the implied tasks for mission accomplishment. The purpose of this level is to analyze the task outputs and subsequently evaluate the mission effectiveness,(5) functions and capabilities define the capabilities which enable forces to conduct operations, (6) components and forces that defines the "by whom" specification, represented with the military actors (integrated units, personnel, equipment,.etc), and (7) interactions and effects that describe "how" the course of actions changes the state of components.

The military domain pays special attention to mission analysis in SoS SE. The mission context is unlike that in traditional SE approaches in which there is little flexibility because individual functions are mapped to only one element in the system [5]. According to [5], "mission context is a key element to assisting SoS engineers to determine the systems that must be involved and the functions that they must perform".

3.4 | Mission Conceptual Model

The analysis of the state of the art reveals that a novel approach, which incorporates the best practices of existing SE frameworks and approaches, taking into account the SoS artifacts and mission understanding, would be welcomed within the SoS SE community. The challenges addressed in this work are: the open SoS environment, the use of mission oriented approach and the definition of mission variants according the operational context.

To avoid any ambiguity, we introduce in the following section a mission conceptual model serving as the basis for our approach. The conceptual model was proposed based on the modeling experience using the SysML language of SE approaches, on SoS artifacts defined in [19, 26] and on the overall experience in rigorously defining a mission [5, 6, 7, 8].

Figure 2 highlights the involved concepts.

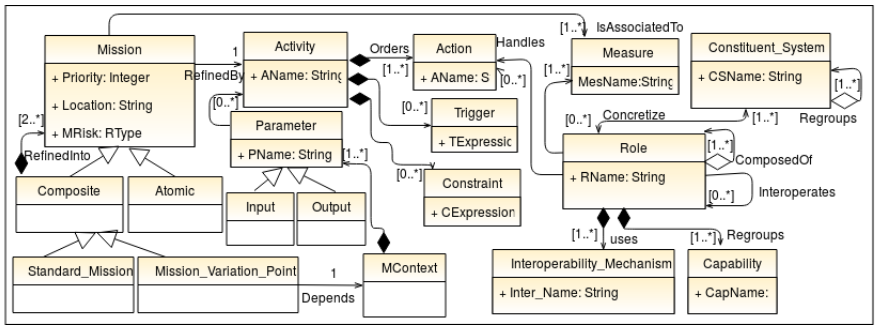

FIGURE 2 Mission Conceptual Model

In this paper, we define SoS as "a set of interacting systems that interact with each other and their environment to provide a common mission" [35]. The *mission* is the main concept on the conceptual model (capability objectives artifact in [26]). We define a mission as a finality that the SoS must achieve by collaborating constituent systems. We suggest decomposing the high level mission (generally abstract) into more concrete missions. In the model, this aspect is expressed by the existence of the two classes *atomic* and *composite* and the relationship *refinedInto* between *composite* and *mission*. The refinement is stopped when we can identify the activity that is associated with the mission (CONOPS artifact in [26]). An *activity* orders a set of *actions*; it can regroup *triggers* and *constraints*; and it can require *input parameters* and provide *output parameters* as in SE Approaches (SoS requirement space artifact in [26]). A *role* handles *action* and gathers the required competences (capability concept) to play the role needed to accomplish the action. The *capability* of a role is defined as "the ability to provide some expertise to the wider needs of an SoS" [30] (systems information artifact in [26]). For each mission, the effectiveness measures must be determined (SoS performance data artifact in [26]).

We define a role as "an abstraction of the characterization of the ideal behavior that will fulfill an action" [36]. Several types of the constituent systems could be used to concretize a role in the concrete architecture: humans, hardware and software existing systems, and institutions. A constituent system is chosen when its capabilities match those required by a role and by considering the trade study and performance measures. The constituent systems can be integrated to meet a role capability. Measures must be defined for each role to guide the choices of constituent systems(SoS performance measures and method artifact in [26]).

The nature of the collaboration between composite missions is basically described by the two variants of the mission composite: standard mission and mission with variation point. A standard mission is composed of sub-missions related with the AND, while a mission with a variation point is composed of mission variants (OR decomposition). The choice of a mission variant depends on the mission context (alternatives). The latter defines the circumstances under which a mission is to be accomplished [7]. We define the mission context as a set of contextual parameters that will determine the course of actions to be performed (performance data artifact in [26]). These parameters are always updated; for example, a location data point is always updated by a geolocation system. The attributes of the mission metaclass allow the specification of the characteristics of the mission as a location if it is important, risk, etc. (SoS risks and mitigation artifact in [26])

The roles interoperate with each other (communicate, exchange data, etc), and each role uses a set of interoperability mechanisms (Radio communication, ADSL connection, etc). Interoperability mechanisms are present in an

SoS; therefore, interoperability is possible by subtyping the communication media or by indirection via another CS. Interoperability thus does not require any adhoc glue design. For instance, to manage crowd, the control and command center can obtain data from cameras via an Internet connection and can communicate with local authorities via a GSM network. Example is given in the Section 6.

4 | GENERAL APPROACH

The proposed process is intended to support the analysis and architecture activities in the SoS wave life cycle. It offers a disciplined procedure for explicitly specifying the SoS end to end mission and generating the appropriate architecture. The process is applicable to acknowledged SoS in which the organization manages SoS and support the SoS SE while independent organizations and SE teams are responsible for the constituent systems [12]. Changes in the constituent systems are based on collaboration between the SoS and the system. The key ideas on which the process is based are as follows: (i) Relying on the design of and reuse of the SoS. (ii) Bridging the SoS analysis and architecture development stages by taking advantage of the expertise of the application domain expert(ADE). (iii) Automating the transition from the analysis stage to the architecture development stage as much as possible to avoid information loss between the application domain expert and the system architect. (iv) Elaborate the concrete architecture model from the abstract one. The latter serves as an invariant that guides the choices of concrete entities.

In the SoS SE development environment, "two levels of stakeholders exist with mixed, possibly competing interests: the SoS stakeholders and constituent system stakeholders" [19]. Since the constituent systems are independent and have their own objectives, stakeholders of individual systems may have little interest in the SoS, may assign SoS needs low priority, or may resist SoS demands pertaining to their system [19]. To manage the competing stakeholder interests, it is important for SoS SE engineer to focus on the operational view of the SoS and to balance the SoS objectives with the constituent system objectives [19]. We argue for the definition of two stakeholders to be involved in the development life cycle of an acknowledged SoS:

Application domain expert : The ADE masters the domain knowledge. Therefore, through her/his experience, this expert can anticipate the solution when refining the mission. The ADE focuses on the SoS operational environment (mission), and she/he does not control the constituent systems that impact the SoS but has the necessary knowledge to balance the SoS mission with constituent system goals. We propose the bridging of the analysis and architecture stages and automating the architecture synthesis as much as possible.

System architect : The system architect is responsible for the generation and realization of the architecture. Based on the conceptual models realized by the domain expert, she/he is responsible for deploying the required constituent systems to produce a concrete architecture.

Figure 3 introduces the main steps and the involved stakeholders in the process implementing the proposed approach. The latter is composed of top-down planning and decision making and bottom-up adjustment based on existing systems.

The goal is to refine the mission until the architecture is attained, while preserving the mission traceability. Therefore, the refinement steps are as follows:

1. Mission decomposition: This step is intended to provide a functional coarse grain view of the mission. This aspect is achieved through an analysis of the general mission objectives to recursively identify more precise sub-

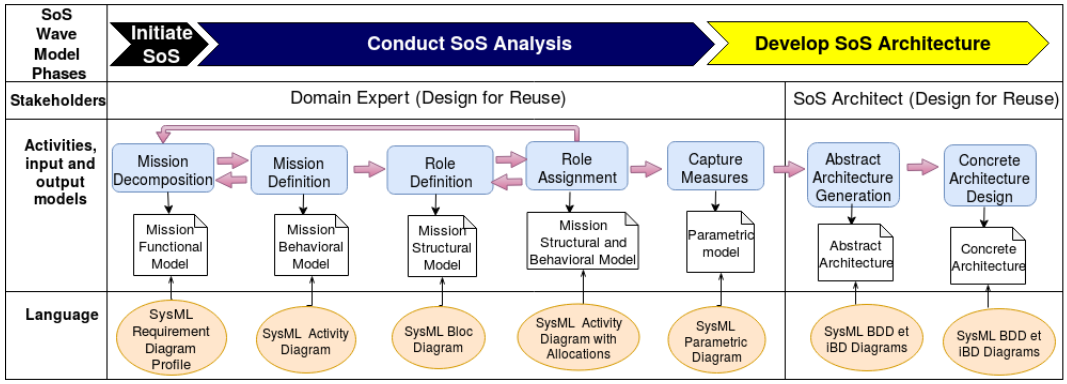

FIGURE 3 Actors and Responsibilities

mission objectives. The criterion for stopping the mission decomposition is the identification of a process that may realize a given sub-mission. We developed a profile extending the SysML requirement diagram to refine the main mission into sub-missions, create context dependent variation points, and capture mission risks. Therefore, this step results in a mission functional model of the SoS.

2. Mission definition: The aim of this step is the design of a fine grained behavioral view of sub-missions using *activities*.The view is elaborated using the SysML activity diagram. Each sub-mission in the mission functional model is associated with an activity using the refine relationship. Complex activities can be decomposed into subactivities, and the criterion for stopping activity decomposition is when a subactivity corresponds to a role capability that we call an action.
3. Role definition: The *role* is used to provide an abstract representation of hierarchy of entities having capabilities that enable the achievement of the mission. The capabilities could be provided or required by roles, thereby allowing the composition of roles. The produced model for role definition is based on a SysML profile extending the block definition diagram.
4. Role assignment: As mentioned previously, activities are composed of actions that correspond to role capabilities. The role is composed of several capabilities, and the same capability can appear in different roles. Therefore, this step is intended to designate the role that must be associated with each action of an activity. This association creates a link with the constituent system through the assigned role.
5. Capture measures: In this step, several effectiveness measures are required to be expressed using the SysML parametric diagram. On one hand, the mission effectiveness metrics are defined by the ADE for assessing the overall performance of the SoS mission. On the other hand, the metrics for choosing the best system among the existing ones for playing a role are defined for each role.
6. Abstract architecture generation and concrete architecture design: The architecture is a structural view that describes the constituent systems of the SoS and their connections. However, all the above-mentioned definitions refer only to roles instead of constituent systems. Therefore, the first generated architecture from the given definitions corresponds to the abstract architecture of the SoS. Thus, the abstract architecture is progressively refined during the architecture analysis to get the concrete architecture. For this step, both the SysML internal block diagram and SysML block definition diagram are employed in this phase.

5 | MISSION AND ROLE MODELING

In the proposed approach, the ADE is responsible for the structural and behavioral model. Constructing such a model is realized through an iterative process. The process is stopped when the expert finds a compromise between the capabilities needed by the mission and those offered by realistic constituent systems. The mission and role modeling is detailed in the following subsections.

5.1 | Mission Decomposition

This phase is intended to understand SoS top level missions and to plan a mission strategy. The essential elements considered in this phase are description of the main missions, variation points, mission location, mission risk and priority. We propose the gradual functional decomposition of the SoS mission and the splitting of complex missions into simple ones [37, 38]. This task is made possible by using the SysML requirement diagram (RE). The RE diagram "allows the specification of a function that a system must perform or a performance condition that a system must achieve" [39]. The SysML RE provides modeling constructs to represent text-based requirements and relate these requirements to other modeling elements. Different relationships are furnished to allow relating requirements to other requirements or to other model elements. These relationships include relationships for "defining a requirements hierarchy, deriving requirements, satisfying requirements, verifying requirements, and refining requirements" [39]. A standard requirement includes the unique identifier and text requirement. Users can add properties if needed [39].

The basic SysML RE is not sufficient to describe all the concepts cited above. For instance, it cannot represent the mission priority and mission risk. Furthermore, it does not allow the creation of variation points since the semantic of decomposition is the conjunction. Therefore, we propose the extension of the RE diagram to allow the ADE to add the desired properties and variation points. This extension is possible since SysML is a highly extensible modeling language [39]. A stereotype is one of the types of extensibility mechanisms in SysML; it is a profile class that allows designers to extend the vocabulary of SysML to create new model elements, which are derived from existing ones but have domain specific properties [39].

Figure 4 illustrates the extension of the SysML RE. The default properties *id* and *text* specify the unique identifier and text requirement, respectively. The *Requirement* is a stereotype that inherits from the metaclass *Class* of UML. The extension is performed by creating a stereotype called *Mission*, which contains the added properties. The stereotype Mission inherits the properties of its superstereotype Requirement, and the following properties are added: *location*, *risk*, *priority*, *version*, *and date*.

The default property *text* of the stereotype *Requirement* can be used to describe the mission goal. The mission location may represent an IP address, GPS coordinates, polar coordinates, region, etc. For this reason, the location is considered as a string parameter. The successive refinements of the main mission generate several sub-missions. The status of these sub-missions is not the same under the main mission, and the priority is the parameter that indicates the importance of each sub-mission. We assume that the priority can take an integer value that indicates the relevance degree of the mission. Given the context uncertainty, risks can affect missions. A risk must first be identified and later mitigated if possible. Iternative activities are defined to prevent the consequences from occurring. Based on the definitions of risk [40, 41], we propose the consideration of risk by attaching the triple R=C,P,Co to each mission (see Figure 4) in which C is a string value representing the future risk cause, and P is a numeric value representing the probability of risk occurrence. The suggested values for Co that represent the consequence are severe, high, moderate, low or very low.

The wave model is based on SoS upgrade cycles. Therefore the properties described above can change in each

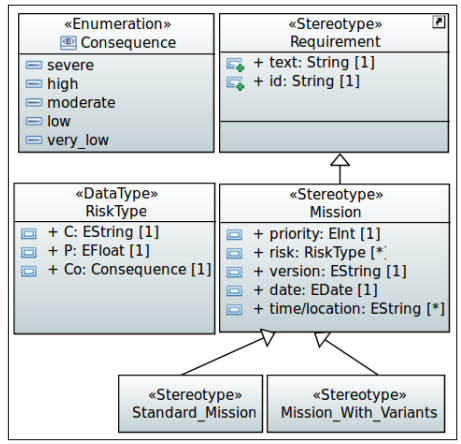

FIGURE 4 Mission Stereotype

upgrade cycle. For example, the priorities are reassessed in each cycle. Therefore, we use the mission stability level as a parameter to determine the stability of the mission definition. We referred to [42] to define this parameter, in which the authors propose to use the version and date of creation/change of/in properties to indicate if and when the mission was changed. To make our process applicable to a wide range of practical contexts, we propose supporting mission variants in the decomposition of the mission. To this end, we define two new stereotypes called the *Standard Mission* and *Mission with a Variation Point*, which inherit from the stereotype *Mission*. The *Standard Mission* is composed of a conjunction of sub-missions while the *Mission with a Variation Point* is composed of a disjunction of sub-missions. When defining missions, the *Mission with a Variation Point* must be defined with the contextual information that allows the resolution of alternatives (see Figure 5).

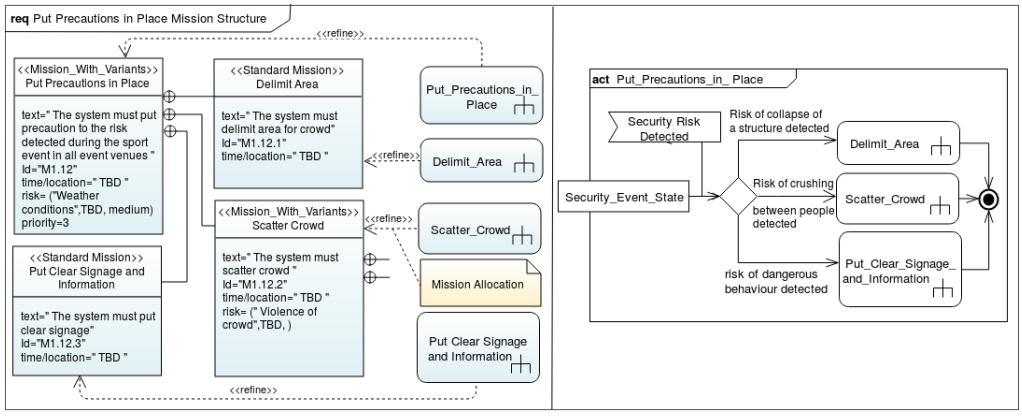

FIGURE 5 Mission Decomposition and Allocation Example

To match the activities/actions to each mission, we propose using the *refine* relationship in the RE diagram. The refine relationship is used to relate a mission to another model element. We use the *call behavior action* element that references an activity to refine a mission. A mission is refined by an activity, and the referenced activity is described

later using an activity diagram (see Figure 5).

5.2 | Mission Definition

This phase is intended to define the activities, course of actions, and capabilities required to handle actions considering the variability in the user environment that impacts the ways the capabilities are executed. Once the activities have been identified, they are described in this phase using the SysML activity diagram. Each *call behavior action* element identified in the mission functional model is refined using an activity diagram. The activity refinement process is stopped when the expert reaches a process composed only of actions that correspond to a capability. In the SysML activity diagram, partitions are used to group actions that have some common characteristics [17, 39]. Partitions are commonly used to regroup actions that are performed by the same system. We propose using partitions to group all the actions that are performed by the same role.

The mission strategy may change according to the context. We described the SoS context using activity *entry parameters*. In this manner, the context of a mission can be inferred from the parameter values, and all the alternatives can be defined at an early stage. *Signals* are used to express the mission triggers, while action scheduling can be expressed using *activities*, *actions*, *data and control flows*. *Constraints* can be set on actions to specify the business semantics. The mission definition phase results in the fine grained behavioral view of the sub-missions, called the mission definition model, which is defined using the SysML activity diagram.

5.3 | Role Definition and Assignment

The role definition phase focuses on constituent system level information that impacts the SoS mission. The roles, capabilities and the possible constituent systems are explored. Given the uncertainty of SoS boundaries, the challenge is to include the roles that can support the SoS mission, and to exclude the useless role ones. In our approach, the ADE considers roles that she/he judges to be relevant entities to the SoS mission. A constituent system enters the SoS boundary when it begins to affect the SoS behavior and leaves when its contribution is negated [13].

Role modeling can be performed using a block definition diagram, in which the structural and behavioral features of a role are described. Hierarchical relationships between the roles can be defined. The capabilities are modeled using operations. Coherent set of capabilities are grouped into interfaces. A *realization dependency* is added from the role to each provided interface, which means that the role will provide each capability in that interface. A role can assert that it requires a set of capabilities by adding a uses dependency to an interface. We created a stereotype Role that inherits the properties of its superstereotype *block* to use SoS vocabulary domain. The communication between roles is done by sub-typing their communication media. For example information exchange could be done by the use of Radio communication.

The purpose of the role assignment phase is to allow the expert to determine the constituent systems that will fulfill actions. Therefore, she/he provides a model that combines the functional and structural views of the mission. As shown in Figure 6, role assignment is based on the functional allocation of activity partition into a role. The role must have capabilities that allow it to achieve all actions contained in the partition. Actions are allocated to capabilities using a call operation action, as shown in Figure 6.

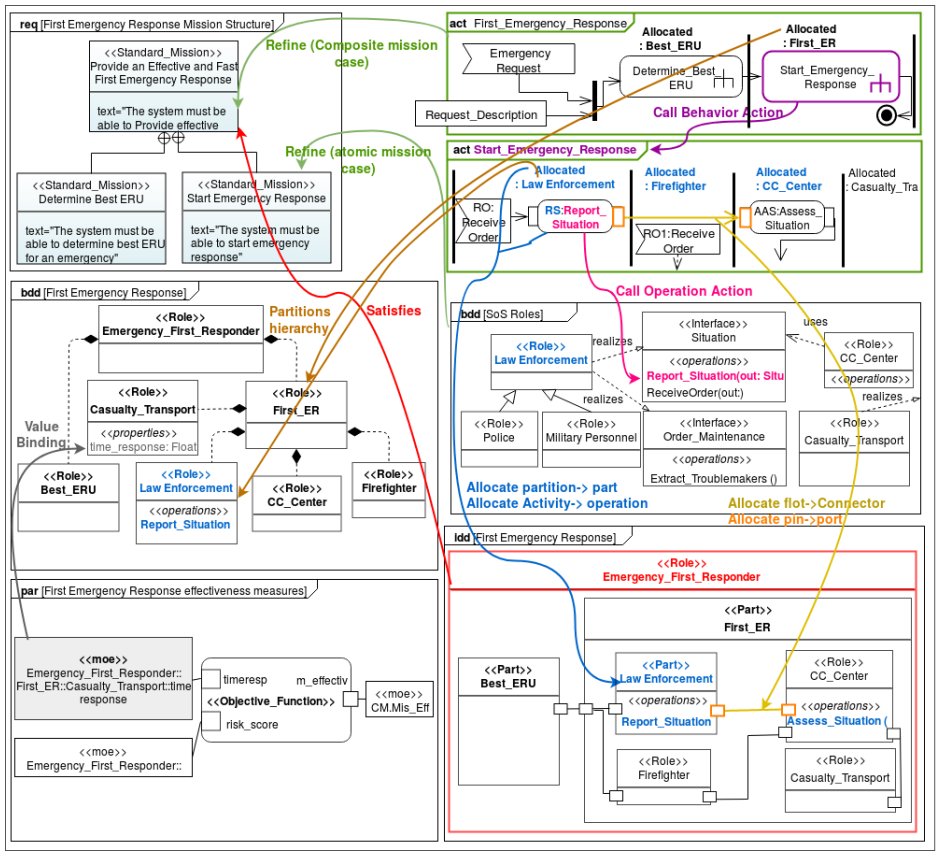

FIGURE 6 Role Assignment Principles

5.4 | Capture measures

This phase focuses on the performance of the SoS solution. The aim is to enhance the SoS performance as much as possible. As in OOSEM, we propose to capture MoEs in the SysML block definition diagram, and the methods and models for calculating the MoEs are described using the SysML parametric diagrams with the objective function. Two kinds of measures must be adopted in this phase:

- Mission measures of effectiveness: The mission MoEs represent the mission-level performance parameter whose value is critical for achieving the desired mission effectiveness.. [17] proposed a useful technique for deducting the MoEs, which consists of using a fishbone diagram to represent a tree of cause–effect dependencies, and then build the parametric diagram from the cause-effect tree.
- Roles measures of performance: Role MoPs are parameters captured to choose between several candidate constituent systems that can concertize a role. Several criteria can be considered, namely, the availability, cost, performance, etc.

5.5 | Abstract and Concrete Architectures

The next step in our approach is to generate the SoS abstract architecture. The focus of this phase is describing which abstract roles interact within a configuration and how. We propose automatizing the process of generating the abstract architecture from the activity diagram and the roles BDD. This abstract architecture is defined using the SysML block definition diagram (BDD) and internal block diagram (IBD). The BDD defines logical decomposition of the main block into roles, and the IBD defines the interactions among the roles such that they satisfy the mission. In the BDD, a block is created for each decomposition of the partition hierarchy. The block features are captured from the role BDD (see Figure 6).

Figure 6 shows also traceability among the different models. The *refine* relationship is used to trace the missions to the corresponding activities. Activities orders actions that are represented by *call behavior actions*. The later correspond to operations offered by roles. The IBD captures the internal structure of a block in terms of the parts, properties and connectors, and this structure is used to display different connections between the parts (roles) that compose the block. The main idea of the transformation here is to consider the activity diagram as a starting point. Considering that an activity refines a mission, our main goal is to build the block that can satisfy this mission (see Figure 6). To this end, we associate each activity model element to a block element that has the name of the model; this block is considered as the main block.

Since our activity diagram is composed mainly of partitions allocated to roles, the partitions are represented as parts in the main block. The parts have the same names and types as the partitions. An activity is characterized by input (or output) parameters. The parameters are provided (or required) by other blocks, which means that each parameter corresponds to an interaction point that is represented in the IBD by a block port. Each flow between the actions from different partitions indicates that data exchange occurs between two different roles. This aspect means that a flow between the two corresponding parts must be created using the corresponding ports. The elements of the resulting IBD are allocated since they are generated from the activity diagram. The resulting IBD is consistent with the activity diagram, as shown in Figure 6.

The abstract architecture is used to obtain one of the possible concrete architectures. To define the concrete architecture that will satisfy the mission, we used the synthesis candidate physical architectures OOSEM's activity [17] since it supports a geographical distribution of components, which is an important aspect of SoS. This OOSEM's activity defines the concrete architecture in terms of its physical components and relationships, and their distribution across system nodes. The physical components of the system are represented by hardware, software and persistent data. We propose that humans can also be used as physical components. The system nodes represent a partitioning of components based on partitioning strategies (physical location,.etc). A concrete architecture is defined by associating each role to combinaison of human, hardware and software constituent systems. The Measures of effectiveness are used to select the preferred architecture [17].

6 | CASE STUDY

In this section, we discuss and explain the steps of the proposed process using a case study focusing on an SoS dedicated to crowd management during a football event. This case study is a systems of systems which is defined in [43]. It is in the same time part of the disaster response system of systems, which is a widely used example of SoS [32, 4]. This SoS is aimed at developing an integrated crowd control system during temporary events of mass transit, such as sports events or political meetings. The case study objective is to refine the crowd management SoS capability objectives into an architecture description using the mission paradigm. The case study data are collected primarily

using document analysis based on documents concerning a French governmental field for crowd management and emergency response [44] and the FIFA regulations [45].

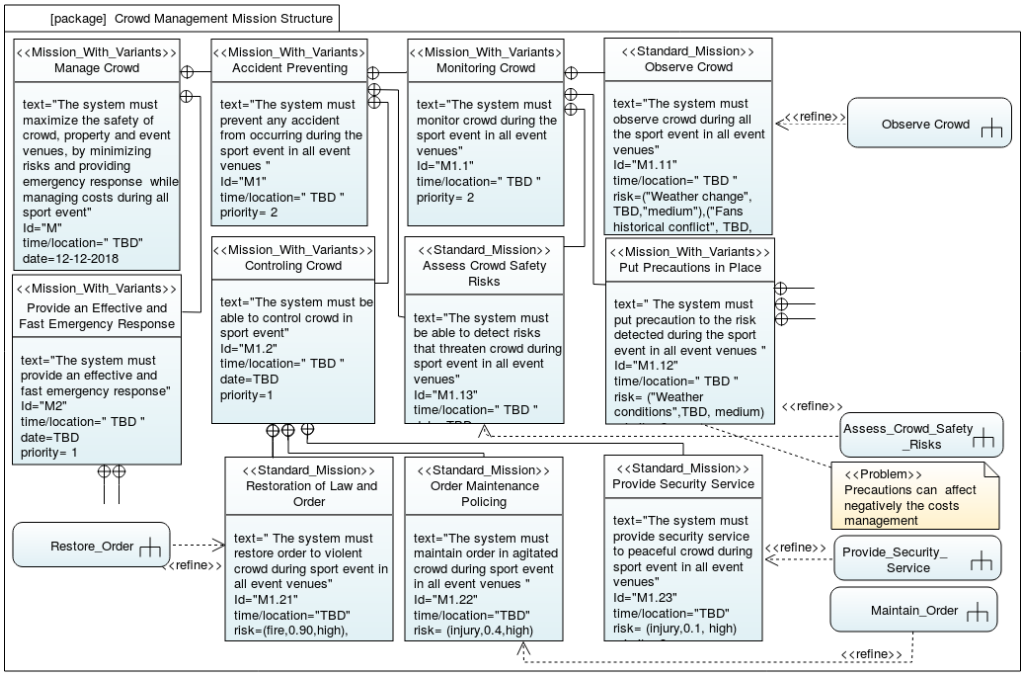

FIGURE 7 Mission Decomposition Diagram

6.1 | Mission decomposition

The initial top-level mission for the Crowd Management SoS is to *maximize the safety of crowd, property and event venues, by minimizing risks and providing emergency response while managing costs during all sport event*. It has to be decomposed into several sub-mission. Decomposition is contained in the Mission Structure package. Figure 7 shows part of the realized Mission Functional Model. Initial mission is represented by the *Manage Crowd* mission. It includes the text statement describing the mission and the "*M*" id. Two second-level sub-missions are designed: *Accident Preventing* and *Manage Costs* ones where the first one has higher priority than the second one. In order to achieve the *Management Crowd* mission, the two sub-missions must be realized. It is specified by the *AND* operator. The mission time/location are not determined yet and depends on the event location. Thus we used the *TBD* acronym to indicate that the values will be determined (To Be Determined). The decomposition process is iterative. For instance, the *Accident Preventing* sub-mission is decomposed into *Controlling Crowd* and *Monitoring Crowd* sub-mission. The later is next decomposed in several sub-missions. Decomposition is stopped when we can refine a sub-mission by an activity using a refine relationship. Using SysML *refine* relationship allows to reuse the SysML traceability mechanisms. The *Observe Crowd in Normal Conditions* Call Behavior action is used to refine the *Observe Crowd* sub-mission, by adding a set of activities description. Each one encompasses a set of activities/actions that refine the sub-mission and take into account the corresponding risks.

The initial top level mission for the crowd management SoS is to *maximize the safety of the crowd, property and event venues, by minimizing risks and providing emergency response while managing costs during all sports events*. This mission must be decomposed into several sub-missions. The decomposition is a part of the mission structure package. Figure 7 shows part of the realized mission functional model. The initial mission is represented by the *manage crowd* mission, which includes the text statement describing the mission and the "M" id. Two second level sub-missions are designed, namely, *accident prevention* and *provide an effective and fast emergency response*, and *accident prevention* has a lower priority than the second one.

The *crowd management* mission is a mission with variants. The semantic of decomposition is the OR operator since the *provide an effective and fast emergency response* mission is not executed if there is no emergency request. The mission time/location are not determined yet and depend on the event location. Thus, we use the TBD (to be determined) acronym to indicate that the values will be determined later. The decomposition process is iterative. For instance, the accident prevention sub-mission is decomposed into crowd control and crowd monitoring sub-missions, and the crowd monitoring sub-mission is further decomposed into several sub-missions. The decomposition is stopped when we can refine a sub-mission by an activity using the refine relationship. Using the SysML refine relationship allows the reuse of the SysML traceability mechanisms. The observe crowd call behavior action is used to refine the observe crowd sub-mission by adding a set of activity description. Each activity encompasses a set of activities/actions that refine the sub-mission and take into account the corresponding risks.

6.2 | Mission Definition

For each *call behavior action* refining a sub-mission in the Mission Decomposition Diagram, an activity is created with the same name. This aspect ensures that each mission is associated with a set of ordered actions. The activity *Observe Crowd* refines the *Observe Crowd* mission. The activity that realizes the *Observe Crowd* actions when a high risk of conflict is identified is based on observation actions inside and outside the stadium. Figure 8 shows the activity that realizes the *Observe Crowd in Stadium* actions. Each action is stored in a partition block, and the partitions are allocated to the corresponding roles. *Observe Entry Points*, *Observe Troublemakers* or *Observe Spectators* are examples of observation actions that provide observations as outputs. The *analyze observation* activity retrieves the observations and generates a state of safety and security in the stadium. The *analyze observation* activity is represented by *call behavior action* (see Figure 8), which means that it is associated with an activity diagram that describes it.

The observation actions should take into account several factors such as political tensions, historical rivalry between fans, and supporter profiles. Each contextual information is given as an input by the activity parameter (see Figure 8). The expected result from the observe crowd with historical conflict between fans activity is the secure event state that is provided as an output parameter.

Mission Decomposition Diagram is not supposed to contain partitions allocations. To avoid presenting two similar diagrams, one with the allocations and the second with no allocations, we have presented only Mission Decomposition Diagram with allocations.

6.3 | Role Definition and Assignment

Once actions/activities have been defined, they must be attributed to a role. A role can be abstract to varying degrees and can be specialized using the inheritance relationship. Figure 9 shows part of the crowd management Role BDD, which includes a set of role hierarchies from the most abstract role to concrete roles. For instance, according to the situation, cost, service availability, and observation target, the *observation* action could be performed using a *camera*,

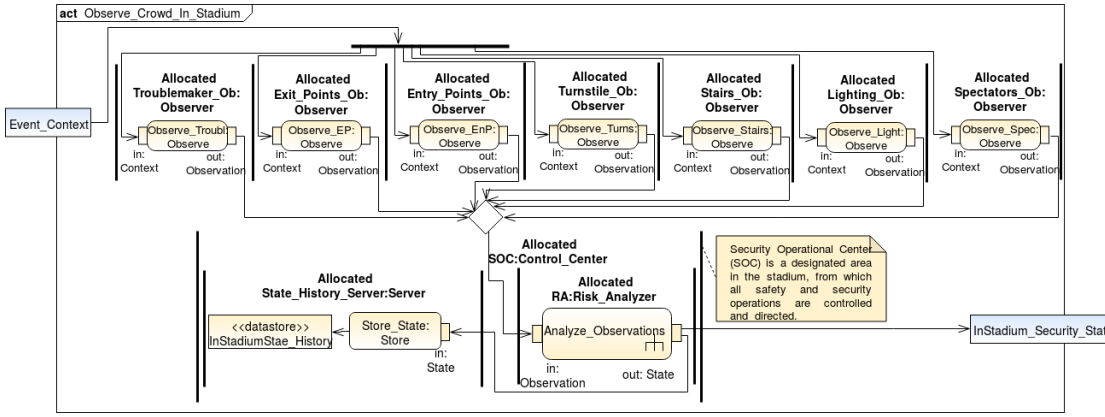

FIGURE 8 Mission Definition Diagram

steward, or *smoke detector*, etc.

As shown in Figure 9, the role observer provides a capability *Crowd Observation*. *Crowd observation* is required by the risk analyzer. The assignment of roles to actions is performed by typing partitions with roles. For instance, in Figure 8, the safety equipment observer partition is typed by the observer role, because it is not possible to determine at this stage if a camera will exist in this place or another physical entity will be used.

The observer role provides a capability *Crowd Observation*, which includes an operation called *observe*, as shown in Figure 9. A call operation action for *observe* is shown with pins corresponding to the entry and output parameters with all observation actions; for instance, *Observe Troublemakers* and *Observe spectators* call the operation observe, as shown in Figure 8. Allocating roles to partitions allows the maintenance of traceability between the role base and the activities/actions related to a sub-mission.

Figure 10 shows the mission-level performance and effectiveness measures that are based on mission outcomes. A part of the measures of effectiveness of the *crowd management in sport event mission* are: the supporters satisfaction, the risks detected and avoided, operational availability, and the global cost. The value of each moe is also calculated. For example, the risk score is calculated using objective function from the max density,cross flows emplacement and number, weather conditions, etc. The mission-level effectiveness are performed to support the evaluation of the design solution.

6.4 | Architecture Design

Figure 11 and Figure 12 show respectively, an aggregate of roles, where each role achieves a specific *Crowd Observer in Stadium* activity or action, and the interconnection among parts that participated in the *Crowd Observer in Stadium* activity. The two diagrams are generated automatically using the ATL rules from the Crowd Observer activity diagram and the role diagram.

In the BDD (see Figure 11), the *In Stadium Crowd Observer* role aggregates the *Control Center* role and the seven *Observer* roles (troublemakers observer, entry points observer, etc). The seven observer roles are responsible of the *observe* mission at a particular location.

In the IBD (see Figure 12), the parts represent how the roles are used in the observation context and have the same role names as shown in the activity diagram. The flow ports are consistent with their definition in the activity

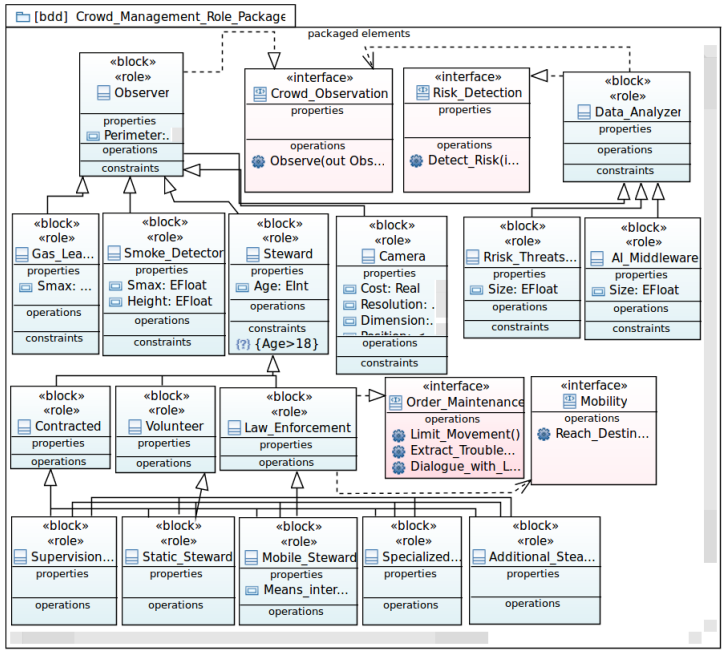

FIGURE 9 BDD Roles Diagram

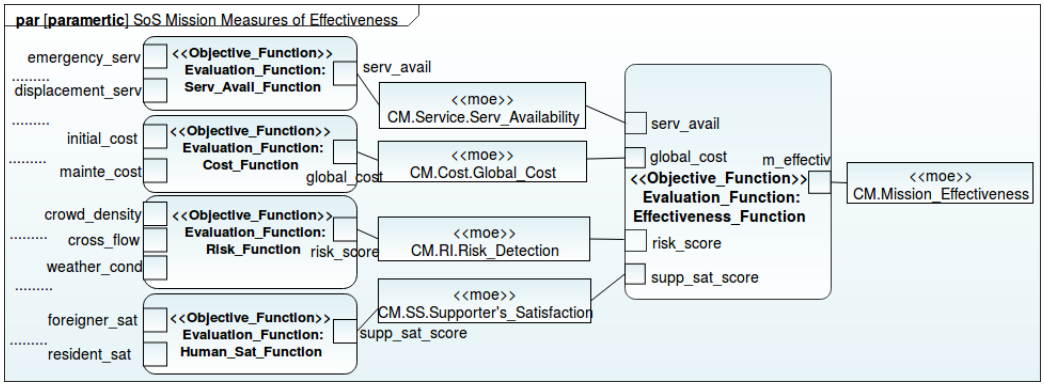

FIGURE 10 Mission Effectiveness Measures

diagram. The IBD for *In Stadium Crowd Observer* role shows the interconnection among the roles that are involved in the *In Stadium Crowd Observer* Activity Diagram. However, there is additional activity diagram that corresponds to the *Analyze Observations* activity. This activity diagram includes different sets of interacting roles. Indeed, all the parts (roles) from all the activity diagrams are represented in the Figure 12. Likewise, the hierarchy of all roles is represented in the Figure 11. The *Control Center* aggregates the *Server* and *Risk Analyzer* roles. The *Risk Analyzer* role aggregates the *Receptor*, *Recorder*, *Data Analyzer* and the *CCTV Operator*.

The abstract architecture is used to obtain a possible concrete architecture by replacing the abstract items with

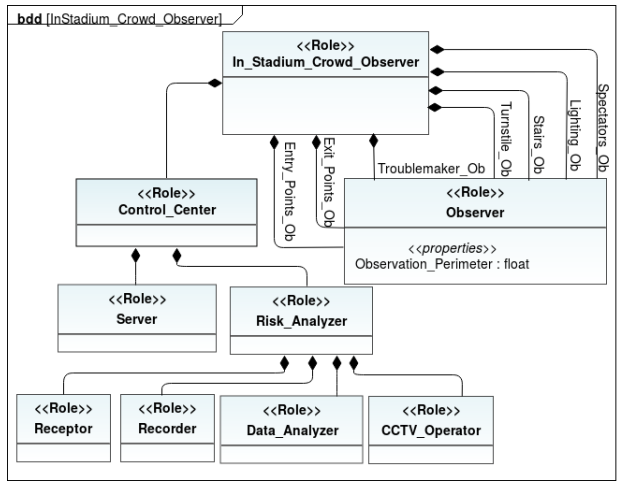

FIGURE 11 Crowd Observer in Stadium Abstract BDD

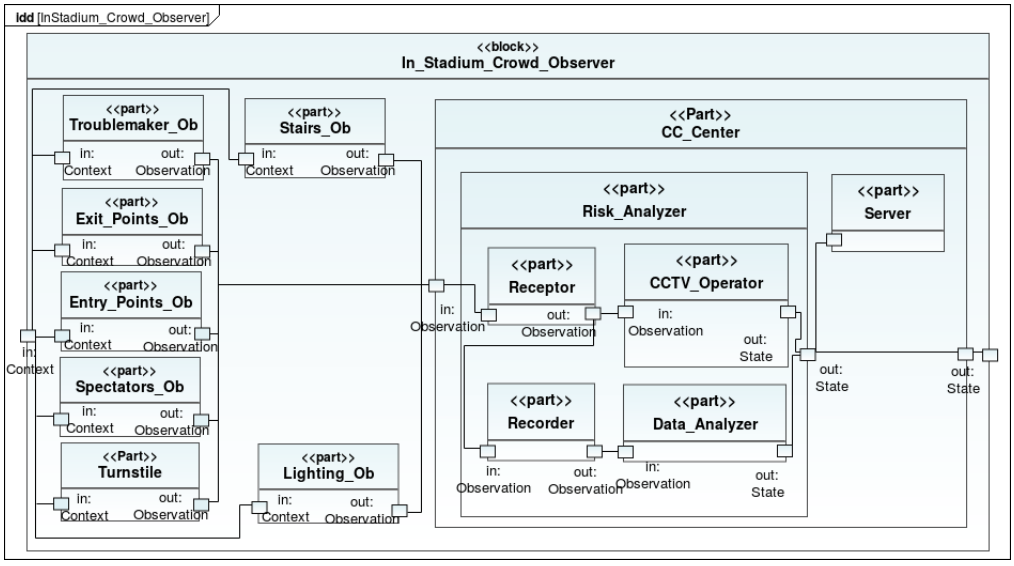

FIGURE 12 Crowd Observer in Stadium Abstract IBD

concrete ones. Each solution is characterized by a set of attributes that have a value distribution. The attributes for a given solution are then evaluated using an objective function, and the results for each alternative are compared to select the preferred solution. Figure 13 shows two variants of the *Observer* role serving as solution to perform the observer role in the *observe exit points* mission. The operational availability, cost, and security effectiveness are the measure of effectiveness (MoE) for the observe mission. The overall effectiveness is calculated for each alternative using a weighted equation of their MoE values.

According to the effectiveness of each alternative. The architect selects, evaluates and chooses the preferred

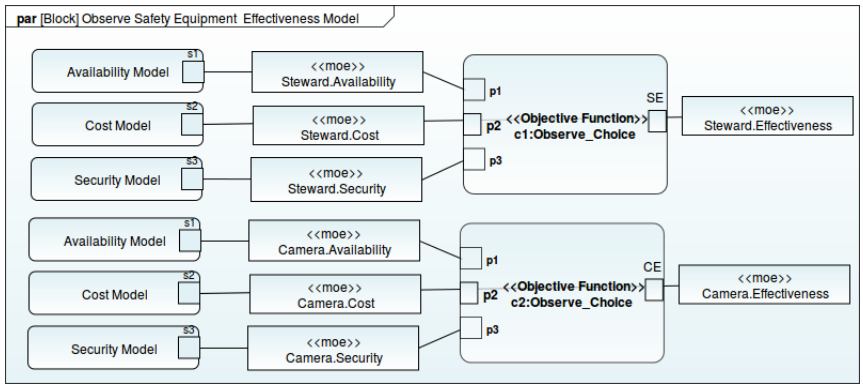

FIGURE 13 Effectiveness Evaluation Results between the two Observers Variants

concrete constituent systems. Like OOSEM, we used the concept of "physical node that represents an aggregation of physical components at a particular location". Figure 14, represents an example of concrete roles allocated to the abstract roles. The *Troublemakers Observer* role was assigned to *mobile steward* human concrete role, who are engaged to penetrate the crowd and observe troublemakers. The concrete architecture constrains the solution space with preselected concrete systems that are available and are able to be assembled in the SoS. When a role is allocated to software, the later must also be allocated to a corresponding hardware role to execute it. Likewise, when a role is allocated to human, the later must also be allocated to a corresponding hardware role to allow him communicate. As already mentioned and as shown in the Figure 14, the steward can communicate using using the Radio headsets.

The concrete architecture evaluation is performed using the overall mission effectiveness (see Figure 10). Simulation remains the best way to analyze the impact of an architecture solution on mission mission effectiveness.

6.5 | Discussion

The main goal of this paper is to consider mission thread to bridge the dissociation between SoS objectives, and the individual functionalities undertaken by constituent systems, to support the SoS mission. To address this SoS SE challenge, we proposed in this paper to maintain a mission focus throughout the SoS SE analysis and the architecting process included in the wave model.

6.5.1 | Back to SoS SE Challenges

This section returns to a subset of challenge problems that were introduced in Section 2, and considers how each challenge is addressed based on the models that have been developed in the case study.

- Operational independence: Which means that any constituent system is independent and can operate serviceably if the SoS is disassembled. In the proposed process, we considered that a constituent system is an independent entity that is able to provide to the SoS a subset of its functionalities, which are called capabilities. The mission actions are allocated to roles and the roles are replaced by available constituent systems based on performance measures. Thus, when a constituent system is disassembled from the SoS, it continues to operate independently.
- Dynamic environment: Constituent systems supporting each role in a mission will vary over the course of the

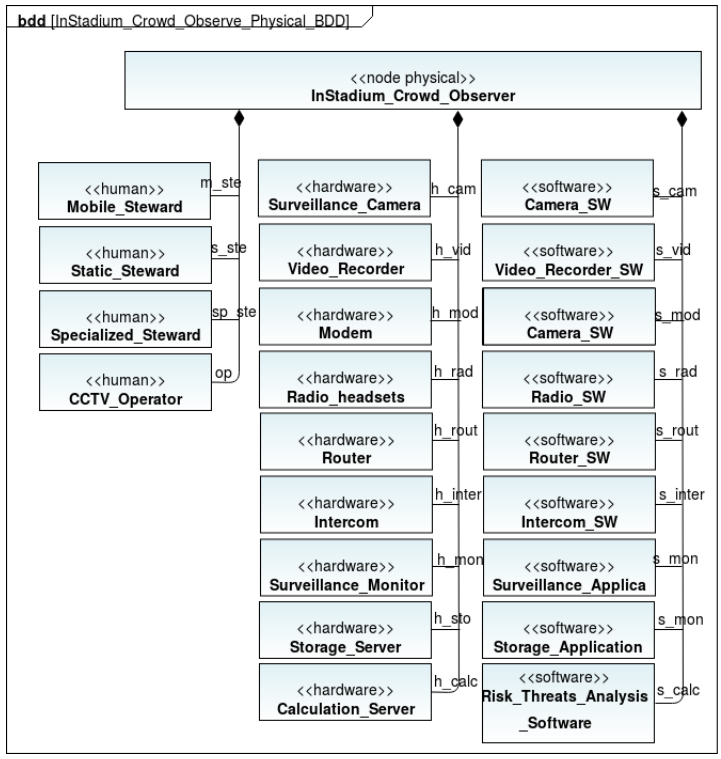

FIGURE 14 Physical Block Definition Diagram of Crowd Observer in Stadium Node

actions. The mission context is key player to determine the constituent systems that are involved in mission accomplishment. We use set of parameters to determine the mission context, when parameters change, the course of actions in the activity diagram changes. The activity diagram in Figure 5 shows a decision node that depends on the security event state. The activity that will be executed depends on the value of risk. Indeed, different architectures are generated for different risk values.

- SoS evolution: SoS evolves over time, but evolves slowly. The evolution could consist of the addition of a new constituent system or a change in the behavior of a constituent system. In the case of addition, we add the new constituent system in the role diagram and add the generalization relationships towards the appropriate roles, or create a new role if the new constituent system holds a new behavior. In the case of a change in the behavior of a constituent system, the role diagram is also updated according to the new capabilities. In the two cases (addition or modification of constituent system), the measures of performance are made and compared to the MoP of the constituent systems holding the same capabilities. The challenge here for the architect is to propagate changes across the concrete architecture.

6.5.2 | Back to the State of the Art

SE approaches are mature and guide engineers in the analysis, development and documentation of complex systems. The majority of them are based on SysML to take advantage of its syntactic richness. We took advantage of their

maturity from the methodological point of view: We captured performance and effectiveness measures using the parametric diagram as in OOSEM. For the role definition phase, we used the service-based interactions provided and required as Harmony SE does to describe the system components. We have refined the RE diagram directly by an activity diagram as in MagicGrid. However, while these approaches are based on the concept of *requirement* that must be met, our approach is based on the *mission* paradigm. Through this paradigm we want to offer a more operational view that supports all the tactical information that can change the order of execution of the actions, the involved systems and different operational environments. Thus, we considered in the decomposition through the RE diagram the different points of variation of a mission, something that is not considered in the SE approaches. These points of variation are solved by activity diagrams taking contextual information as input parameters to show the different alternatives and decisions. So refinement does not just consider leaf missions as in other approaches.

SoS SE work is most often influenced by DoDAF and MoDAF frameworks. For example, the Compass and Danse projects have tried to reduce the number of views, proposed by these frameworks, to make them more manageable. An example of these influences concerns the fact of considering concrete systems during the specification stage. So, SoS boundaries are dictated by existing systems. In our approach, the SoS specification is more open thanks to the use of a more abstract definition of constituents (Role concept).

Due to the diversity that can exist between the constituent systems, interoperability becomes an important aspect when describing a SoS. We consider that this aspect is the responsibility of the system architect and must be treated in a downstream step with respect to the specification. This out of scope of this paper and will be dealt with in a specific work.

7 | CONCLUSION

This paper supports the maintenance of a mission focus throughout the SoS SE analysis and the architecture process included in the wave model. In fact, the SoS are acquired to satisfy new capabilities in a mission context. The later is a key element to assist SoS engineers to determine the systems that must be involved and the functions they must perform.

The first contribution of this paper is the proposition of a mission conceptual model. The later shows the main concepts characterizing SoS mission. The mission conceptual model was proposed based on the modeling experience using the SysML language of SE approaches, on SoS artifacts defined in [19, 26] and on the overall experience in rigorously defining a mission [5, 6, 7, 8]. The second and the main contribution of this paper is the proposition of a mission based process that strengthens the links between the SoS analysis stage and the architecture stage in the SoS wave life cycle. The process concerns an acknowledged SoS.

The SoS analysis stage is conducted by the ADE, and the architecture stage is directed by the system architect. The ADE defines the SoS mission taking into consideration the mission context. The mission context is represented by global parameters, their corresponding values determine the mission threads. The ADE defines also the roles, abstract entities that encapsulate the ideal behavior that will fulfill an action. The concept of role is used to deal with the uncertainty of the availability of SoS constituent systems. We used model transformation mechanisms to generate the corresponding abstract architecture, from which the system architect can deploy concrete constituent systems. Different abstract architectures are generated for different mission contexts, and different possible concrete architecture could be obtained by replacing the roles with human, software and hardware constituent systems. Measures of performance where used to choose the best constituent system and the available one.

The use of this approach to model several SoS in the same domain can help identify recurrent concepts in the

form of parts of sub-missions, roles and capabilities. Recording these concepts in knowledge bases allows their reuse, which can help the domain expert enhance the efficiency in SoS design and decrease the cost of using the model-based approach. We consider the SysML models as the basis for the assessment of the SoS architecture. Such models allow to assess gaps in mission performance and to improve the analysis of SoS behavior earlier in the development cycle.

Dahmann [46] argues that the SysML model "represents an unambiguous, structured and executable representation of the SoS architecture that can be exploited and simulated". The simulation is useful to enhance the effectiveness of SoS mission or to observe the SoS behavior in order to detect undesired emergent behavior [47]. This implies to make an efficient link with simulation tools. An interesting perspective will be the development of automated interfaces between the architecture models and a simulation environment. Another interesting perspective is the integration of the security aspect into the SoS mission process to identify vulnerabilities at an early stage as proposed in [48].

references

[1] Estefan JA. Survey of Model-Based Systems Engineering (MBSE) methodologies, INCOSE; 2008.

[2] Woodcock J, Larsen PG, Bicarregui J, Fitzgerald J. Formal Methods: Practice and Experience. ACM Computing Survey 2009 october;41(4):1–36.

[3] Council on Systems Engineering I. INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities; 2015.

[4] Nielsen CB, Peter Larsen G, Fitzgerald J, Woodcock J, Peleska J. Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions. ACM Computing Survey 2015 sep;48(2):1–18.

[5] Vesonder G, Verma D. RT-171: Mission Engineering Competencies Technical Report; 2018.

[6] Sousa-Poza A. Mission Engineering. International Journal of System of Systems Engineering 2015 01;6:161.

[7] Sheehan JH, Deitz PH, Bray BE, Harris BA, Wong ABH. The Military Missions and Means Framework. U.S. Army Materiel Systems Analysis Activity; 2004.

[8] Military Agency For Standardization. STANAG: Formats for ordres and designation of timings, locations and boundaries. NATO, 9 ed.; 2014.

[9] Jamshidi M. System of systems engineering - New challenges for the 21st century. IEEE Aerospace and Electronic Systems Magazine 2008 May;23(5):4–19.

[10] Sage AP, Cuppan CD. On the Systems Engineering and Management of Systems of Systems and Federations of Systems. Inf Knowl Syst Manag 2001 dec;2(4):325–345.

[11] Maier MW. Architecting principles for systems-of-systems. Systems Engineering 1998;1(4):267–284.

[12] Dahmann J, Rebovich G, Lane J, Lowry R, Baldwin K. An implementers' view of systems engineering for systems of systems. In: 2011 IEEE International Systems Conference Montreal, Canada; 2011. p. 212–217.

[13] Lowe PN, Chen MW. System of Systems Complexity: Modeling and Simulation Issues. In: Proceedings of the 2008 Summer Computer Simulation Conference SCSC '08, Vista, CA: Society for Modeling & Simulation International; 2008. p. 36:1–36:10. http://dl.acm.org/citation.cfm?id=2367656.2367692.

[14] Cole R. The changing role of requirements and architecture in systems engineering. In: 2006 IEEE/SMC International Conference on System of Systems Engineering IEEE; 2006. p. 6–10.

[15] Lane JA, Dahmann JS. Process Evolution to Support System of Systems Engineering. In: Proceedings of the 2Nd International Workshop on Ultra-large-scale Software-intensive Systems ULSSIS '08, New York, NY, USA: ACM; 2008. p. 11–14.

[16] Lykins H, Friedenthal S, Meilich A. Adapting UML for an object oriented systems engineering method (OOSEM). In: INCOSE International Symposium Minneapolis; 2000. .

[17] Friedenthal S, Moore A, Steiner R. A Practical Guide to SysML: The Systems Modeling Language. 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 2014.

[18] Douglass BP. White paper: The Harmony Process; 2005.

[19] Department of Defense. Systems Engineering Guide for Systems of Systems; 2008.

[20] Dombkins D. Complex project management : seminal essays / by David H. Dombkins. BookSurge Publishing North Charleston, S.C; 2007.

[21] Hans-Peter H. White paper: SysML-based systems engineering using a model-driven development approach; 2008.

[22] Morkevicius A, Aleksandraviciene A, Mazeika D, Bisikirskiene L, Strolia Z. MBSE Grid: A Simplified SysML-Based Approach for Modeling Complex Systems. INCOSE International Symposium 2017;27(1):136–150.

[23] Murray C. White paper: Rational Unified Process for Systems Engineering, RUP SE, Version 2.0; 2003.

[24] Ingham MD, Rasmussen RD, Bennett MB, Moncada AC. Engineering complex embedded systems with State Analysis and the Mission Data System. In: AIAA Journal of Aerospace Computing, Information and Communication; 2004. p. 507–536.

[25] Department of Defense. Defense acquisition guidebook. Washington, D.C.: U.S. Dept.of Defense, Pentagon; 2010.

[26] Dahmann J, Rebovich G, Lane JA, Lowry R. System engineering artifacts for SoS. In: 2010 IEEE International Systems Conference San Diego, CA, USA; 2010. p. 13–17.

[27] U S Department of Defense, DoDAF Architecture Framework Version 2.02; 2010. https://dodcio.defense.gov/library/dod-architecture-framework/.

[28] UK Ministry of Defence, MOD Architecture Framework (MODAF); 2004. https://www.gov.uk/guidance/mod-architecture-framework.

[29] Information technology - Object Management Group. Unified Profile for DoDAF and MODAF (UPDM), 2.1.1; 2017.

[30] Lock R, Sommerville I. Modelling and Analysis of Socio-Technical System of Systems. In: Proceedings of the 15th IEEE International Conference on Engineering of Complex Computer Systems ICECCS '10, IEEE Computer Society; 2010. p. 224–232.

[31] Lochow T, Sanduka I, Bullinga R, Arnold A, Kalawsy R, Cristau G, et al. Concept Alignment Example description; 2013.

[32] COMPASS Consorsium, The Compass Project; 2014. http://www.compass-research.eu/.

[33] Silva E, Cavalcante E, Batista T. Refining Missions to Architectures in Software-Intensive Systems-of-Systems. In: 2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (JSOS) Buenos Aires, Argentina; 2017. p. 2–8.

[34] Silva E, Batista T, Oquendo F. A mission-oriented approach for designing system-of-systems. In: 10th System of Systems Engineering Conference SoSE 2015; 2015. p. 346–351.

[35] Luzeaux D, Ruault JR. Systems of Systems. ISTE Ltd; 2010.

[36] Cherfa I, Sadou S, Belloir N, Fleurquin R, Bennouar D. Involving the Application Domain Expert in the Construction of Systems of Systems. In: 2018 13th Annual Conference on System of Systems Engineering (SoSE) Paris, France; 2018. p. 335–342.

[37] Bresciani P, Perini A, Giorgini P, Giunchiglia F, Mylopoulos J. Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems 2004 may;8(3):203–236.

[38] Van Lamsweerde A. Requirements Engineering: From Craft to Discipline. In: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering New York, NY, USA: ACM; 2008. p. 238–249.

[39] Object Management Group. Systems Modeling Language V1.5. http://www.omg.org/spec/SysML/1.5/: Object Management Group; 2017.

[40] Office Of The Under Secretary Of Defense For Acquisition Technology And Logistics. Risk Management Guide for DOD Acquisition, 6th Edition (Version 1.0). WASHINGTON DC.: Defense Technical Information Center; 2006. https://books.google.fr/books?id=-pdTAQAACAAJ.

[41] International Organization for Standardization. ISO 31000 Risk management — Guidelines. 2 ed.; 2018. http://www.iso.org/obp/ui/#iso:std:iso:31000:ed-2:v1:en.

[42] Dos Santos Soares M, Vrancken J, Verbraeck A. User requirements modeling and analysis of software-intensive systems. Journal of Systems and Software 2011;84(2):328 – 339. http://www.sciencedirect.com/science/article/pii/S0164121210002876.

[43] Gorod A, E White B, Ireland V, Gandhi SJ, Sauser B. Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering. Complex and Enterprise Systems Engineering, CRC Press; 1 edition (July 1, 2014);.

[44] Groupement des Industries de Défense et de Sécurité terrestres et aéroterrestres. Gestion des foules. GICAT; 2018.

[45] FIFA. Stadium Safety and Security Regulations. FIFA; 2018.

[46] Dahmann J, Markina-Khusid A, Doren A, Wheeler T, Cotter M, Kelley M. SysML executable systems of system architecture definition: A working example. In: 2017 Annual IEEE International Systems Conference (SysCon) Montreal, Quebec, Canada; 2017. p. 1–6.

[47] Benabidallah R, Sadou S, Ahmed-Nacer M. Using System of Systems' States for Identifying Emergent Misbehaviors. In: 27th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2018, Paris, France, June 27-29, 2018 IEEE Computer Society; 2018. p. 66–71.

[48] Messe N, Belloir N, Chiprianov V, Cherfa I, Fleurquin R, Sadou S. Development of Secure System of Systems Needing a Rapid Deployment. In: 14th Annual Conference System of Systems Engineering, SoSE 2019, Anchorage, AK, USA, May 19-22, 2019;. p. 152–157.