# Involving the Application Domain Expert in the Construction of Systems of Systems

Imane Cherfa*§, Salah Sadou*, Nicolas Belloir†, Régis Fleurquin* and Djamal Bennouar‡
*Université Bretagne Sud - IRISA, France
†Military School of St Cyr Coetquidan - IRISA, France
§University of Blida - LRDSI, Algeria
‡University of Bouira - LIMPAF, Algeria
Email: *firstname.lastname@irisa.fr*

***Abstract*—The system of systems (SoS) is a system whose definition is based on pre-existing independent systems in the runtime environment. The latter is in perpetual evolution thus forcing a recurrent adaptation of the SoS. Thus, during their life cycle the SoS are very exposed to the problem related to the evolution mentioned above. This problem is mainly due to a poor communication between the requirement definition stage and the design stage.**

**In this paper we propose a method for addressing SE for SoS using the concepts Mission and Role. The first one allows the definition of the SoS's behavior, while the second allows to abstract this definition with respect to the constituent systems that may actually exist in the environment. This definition will be translated into an abstract architecture. The later will serve as a guide and controller of the choices proposed by the system architect during the design and evolution stages. With our approach we have correctly defined an SoS concerning crowd management during a sport event.**

## I. Introduction

During system development process, choices and decisions are made at each stage. For instance, the application domain expert makes decisions on business aspects, while the system architect is responsible of implementation choices. These two stockholders cover the stages concerning the requirement definition and the design. These two stages are crucial to the 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. This is necessary to facilitate the evolution of the system [1]. If the definition of the requirements is not formal, the risk of deviation of the design from the initial objectives is real with each evolution of the system. The system of systems (SoS) is a system whose definition is based on pre-existing independent systems in the runtime environment [2]. The latter is in perpetual evolution thus forcing a recurrent adaptation of the SoS. Thus, during their life cycle the SoS are very exposed to the problem related to the evolution mentioned above. To solve this problem, we propose to create a strong link between the requirement definition stage and the SoS design stage. The idea is to allow the application domain expert to define her/his objectives and requirements in a formal way that will serve as a guide and a controller of the choices proposed by the system architect during the design and evolution stages. We suggest to use the mission paradigm [3] for the language intended for the application domain expert. This definition must include a maximum of knowledge from the domain expert. This definition will be translated to an abstract architecture holding invariants that will guide the choice among the possible solutions during the development and adaptation of the SoS.

This approach requires a good understanding of all details about capabilities of the constituent systems. However, it is not possible at the requirement definition stage to know which constituent systems will really contribute to the mission accomplishment at run time. The application domain expert has only prior knowledge of the systems that may be used by the SoS. That's why we propose to manipulate roles, more abstract entities, during the mission definition instead of concrete systems.

So, the purpose of the work presented in this paper, is to provide a process-based model to specify the mission and choose constituent systems that have the necessary capabilities to its accomplishment.

We illustrate the global process in all its phases with the Crowd Management case study, which is part of the Disaster Response System of Systems case study.

The remainder of this paper is organized as follows: next section presents our general approach for the SoS construction. Section III describes mission modeling, goes into details of the proposed language for describing capabilities and explains the abstract architecture modeling and its realization. In Section IV we illustrate the proposed approach through a case study. Section V discusses some related work while Section VI presents concluding remarks and directions for future work.

## II. General Approach

The proposed approach is based on the idea that a model can serve as a mean of transmitting knowledge from the application domain expert to the system architect. However, it is not always easy to find a form of modeling that suits two actors from different domains. In our approach, we propose a sufficiently generic SoS modeling to suit experts from the non IT domain. Then, we generate a specific architecture for the system architect. Figure 1 introduces the main steps, and the involved stockholders, of the process implementing our approach. In the following, we describe these steps.



978-1-5386-4876-6/2018/$31.00 ©2018 IEEE

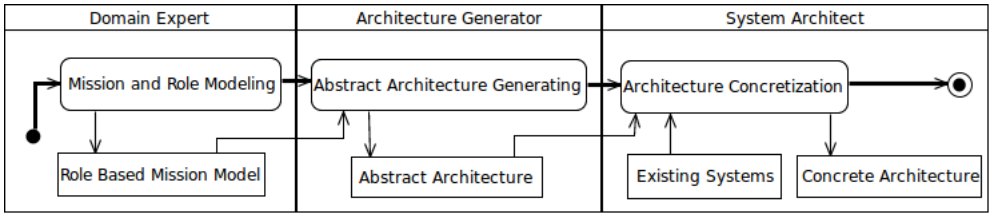


Fig. 1. Actors and responsibilities

### A. Mission and Role Modeling

In this step, we interlace the requirements analysis and design phases of system development process. As pointed by [4], SoS is designed to enhance the overall robustness, lower the cost of operation and increase reliability of the overall complex (SoS) system. Thus, we believe that the application domain expert is the most suited person to state the what and the how about the business aspect of the SoS. Preserving this information helps to achieve the properties planned for the SoS. We use the mission paradigm for the definition of SoS by the application domain expert. Not knowing what systems will actually exist when running the SoS, the application domain expert will describe the mission in the form of a collaborative roles. The later correspond to an abstract representation of what she/he hopes to find in the runtime environment. The collaborative aspect will be described through SysML activity diagrams, while the roles will be described with capability diagrams that we propose (further described in next section). The result is the SoS mission description model based on the involved roles and their collaborations.

### B. Abstract Architecture Modeling

The mission definition contains all the knowledge of the application domain related to the mission. The mission description form is not adequate for a system architect. Indeed, the purpose of the mission description language is to allow non-IT experts to better express their needs. This step consists of transforming the mission description into an abstract architecture. The architecture is called abstract because it uses roles instead of concrete systems. For the description of the abstract architecture we use the SysML block diagram. To this diagram, we associate OCL constraints in order to express all the semantics drawn from the definition of the mission. At this level, the architect holds an architecture representing a first solution for the targeted SoS, but which does not take into account the state of its execution environment.

### C. Concrete Architecture Modeling

At this level, the system architect uses her/his knowledge about the concrete systems in order to map them to the roles from the abstract architecture. So, the system architect must know the semantics of the capabilities, as described in the roles, to identify their equivalents in the behavior of existing systems. This is the only additional knowledge required for the system architect to be able to manipulate abstract architecture. Thus, the architect can use all his knowledge on architectural aspects and on the environment of execution to transform the abstract architecture into a concrete one. However, the architect is limited in his choice of solutions by the constraints defined in the abstract architecture and which will apply on the concrete architecture. This is the guarantee of the good respect of the requirements defined by the application domain expert.

## III. Mission and Role Modeling

To define SoS missions, we need to describe all important concepts related to SoS functionalities. However, considering mission as a main concept is not limited to SoS engineering. Indeed, in the military domain, mission is a strong concept that is defined in a rigorous manner, and generally expressed throw a well structured document. The most famous one is called *Operation Order* [5], which specifies how a mission must be defined in case of the NATO organization. we make the assumption that the military mission is based on a large and long experience, we propose to transpose the military vision of the mission to the SoS context. Thus, a SoS is built to accomplish a given mission. So, we can specify a SoS through the specification of its mission.

According to [5], a mission specification must express five important elements defining the mission perimeter:

1) *Situation*: describes any items of information which affect the mission planning.
2) *Mission Statement*: sets a concise statement of the mission to be accomplished. This is the main objective of the mission.
3) *Execution*: specifies ordered and coordinated operations that contain roles/tasks to be carried out.
4) *Administration and Logistics*: give the plan for administrative and service support of operations.
5) *Signal and Command*: define command and signal instructions used to communicate.

Transposing this five concepts to describe a SoS mission may be done as followed. Situation concept can be used to describe the environment in which the mission of the SoS will evolve. Mission Statement allows to describe the main objective of the SoS (for instance, rescuing as many people as possible). Execution concept is used to organize a sequence of actions that the SoS must perform. Signal and command describe the constituent systems communication under the SoS. As we focus on the operational aspect of the mission, Administration and Logistics will not be taken into consideration. This aspect will be the subject of a future work.

One of the differences between a military mission and a SoS mission is that when specifying a SoS mission we have no guaranty that the constituent systems remain the same throughout the SoS life-cycle. While in the military case, often the constituent systems are stable during the operation. As explained in the introduction, we think that the best way to deal with this uncertainty, about the state of the runtime environment of the SoS, is to design the SoS without considering the real existing constituent systems. We propose to specify an abstraction called *Role* to designate constituent systems. Role are characterized by what they are able to do (called *capability*). To concretize the definition of the SoS capabilities will be used to identify constituent systems corresponding to abstract roles. Thus, our approach is based on two complementary models: The first one is used to specify the mission of the SoS, while the second one allows to specify the roles involved in the mission and their capabilities. These models are described in the following.

### A. Mission Modeling

To specify a SoS mission we propose to use the SysML Activity diagram. Indeed, SysML language [6] is the reference ADL for system architects. It was defined in order to specify complex systems. Activity Diagram is very suited to express data and control flows between actions. In the SoS context, data and flows are used to express collaboration between roles (constituent systems). An action is something realized by a constituent system. With *activity entry parameters*, we can express the SoS situation. *Activity name* sets the mission statement. Using *signals* and *events* we model Signal and Command elements. Finally, the action scheduling can be expressed using *activities*, *actions*, *data* and *control flows*. *Constraints* can be set on *actions* to specify the business semantics.

To summarize, a SoS mission is considered as a set of processes, each one composed of an ordered set of activities or actions representing a graph. This graphical representation may be very helpful for other aims than SoS specification (costs estimation, critical path identification...).

Modeling a SoS is basically realized by an application domain expert throw a refinement process. The refinement is stopped when the expert finds a needed role that may match a realistic constituent system.

### B. Capability Modeling

To be able to identify a matching between a role used in a diagram, representing a mission, and a possible constituent system, the system architect relies on the capabilities required to play the role. Thus, the latter must be correctly specified by the application domain expert in order to avoid any error in the choice of the constituent systems by the system architect.

To this aim, we propose a new kind of diagram which allows to model role capabilities. Figure 2 shows the metamodel (called CMM) of the Capability Diagram highlighting the involved concepts.

Based on her/his knowledge of the application domain, the expert must maintain some level of abstraction so that it allows to conceive the mission without prejudging on the constituent systems that will actually exist at the launch of the SoS. This is done through the concept of *Role* defined by the metaclass *Role* in Figure 2, which is the main concept in the CMM metamodel. It gathers the needed competences (metaclass *Capabilities*) to play a role needed to accomplish the mission. A capability of a role is defined as the ability to provide some expertise to the wider needs in the SoS context [7].

We define a role as an abstraction of the characterization of the ideal behavior that will fulfill an action. In the final concrete architecture, a role would be realized by various kind forms of constituent systems: existing systems, organizations or humans. A constituent system is chosen when its capabilities match with those required by a role.

The metaclass *Setting* defines the information that we need to know about a capability. It may encapsulate the specificities of the application domain for a given capability. For instance, the capability *search and rescue at sea* which is associated to the role *maritime rescue*, needs information about *wind speed* and *wave height*. This kind of information is defined by a name (metaattribute *Setting_Name*) and a value (metaattribute *Setting_Value*).

Capabilities in the role may not be feasible in certain circumstances. For instance, if the wind speed is greater then 103 Km/h (Beaufort 11), the capability *search and rescue at sea* can not be done. All constraints related to a role are expressed through ACL (Architecture Constraints Language) [8] profile. The later, is composed of an OCL-like language, and a MOF Metamodel(CMM in our case). The metaattribute *Expression* in the metaclass *Constraints* represents the constraint associated to the role.

### C. From Mission to Abstract Architecture

The architecture modeling phase defines the SoS global architecture in terms of systems interconnected through data and control flows. In our approach, architecture is defined in two stages: first through a transformation of the mission definition into an abstract architecture. Then, by the concretization of the latter in an architecture containing the existing constituent systems in the SoS's environment.

The application domain expert is at the origin of the definition of SoS. She/he uses the definition of the mission and the role diagram to express the objectives of the SoS while making explicit her/his know-how. The aim of our approach is to preserve this definition and make it useful to the system architect. Thus, the construction and evolution of the SoS will be conform with the decisions made by the application domain expert. To this end, we generate an abstract architecture that becomes the work base for the system architect. This abstract architecture is defined using the SysML bloc diagram. So, the main objective of our approach is the generation of this abstract architecture.

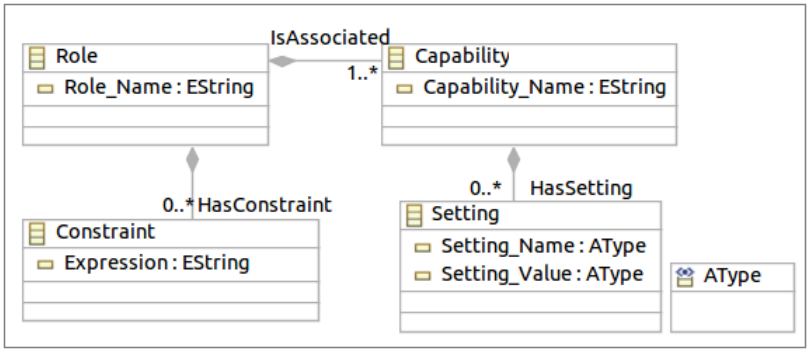


Fig. 2. Capability Modeling Metamodel (CMM)

We transform the mission definition into an abstract architecture using the ATL[1] model transformation language. The transformation model is too long to be presented in this paper, however, here are its principles:

**Role transformation**: Each role affected to an action is translated into a bloc. Role capabilities are translated into operations part of the bloc. Capability settings are defined as properties of the bloc. To distinguish settings of each capability, we used a special notation to the properties: capability.setting. Constraints related to roles are expressed in the constraint part of the bloc.

**Process transformation**: To Each process in the activity diagram corresponds a collaboration between several roles. The role that handle an activity is represented as a bloc composed of an already described collaboration of blocs. Data and control flows exchanged between roles are expressed through ports and flows, and are deducted from the activity diagram.

## IV. Case Study

In this section, we go through and discuss the three stages of the proposed process using the Crowd Management case study. This case study is a part of the Disaster Response System of Systems which is a widely used example of SoS [2], [9], [10]. The Crowd Management case study was defined in [10]. It aims at developing an integrated crowd control system during temporary events of mass transit, such as sport events or political meetings. The considered case study concerns the development of a Crowd Management SoS (CMSoS) related to sport events.

The challenge is to describe the different activities and their associated roles to achieve *Managing Crowd* mission of the SoS.

### A. Mission and Role Modeling

The high level process related to the main mission (*Managing Crowd*) is composed of the seven activities proposed in [10] (see Figure 3) . They are needed to control any emergency situation, to which we add the *Event Planning* activity that includes all processes related to the event planning and the collect of venues informations (Understanding Visitors and Stakeholders, Capacity Planning,...). This activity is essential since we are interested by a determined event (sport event).

The *Managing Pre-event Stage* activity serves to control crowd behavior and avoid any incident. It starts when the event starts. It is a repeating activity and interruptible at the same time. It can be interrupted by the *End of Event* or when there is an *Accident Alert*.

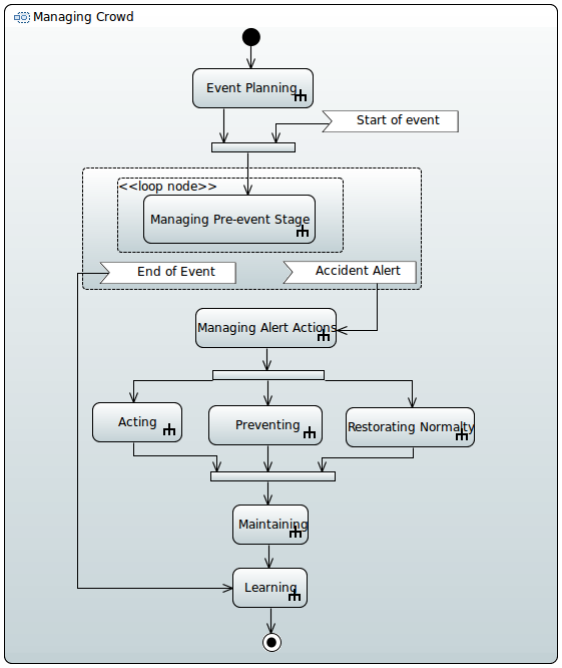


Fig. 3. Crowd Managing SysML Activity Diagram

Once the *Accident Alert* signal received, the *Managing Alert Actions* activity, which consists in applying the best emergency protocol, is triggered. The three activities *Acting, Preventing and Restorating Normality* aim respectively at consolidating the first aid, avoiding disaster(minimizing damages) and working to restore normality. When the three activities are finished, the *Maintenance* activity for repairing damages is launched and followed by *Learning* activity that serves to improve future actions and decisions. This activity can cause SoS model

[1]Atlas Transformation Language: http://www.eclipse.org/atl

updates.

Let us focus more on the processes related to *Managing Pre-event Stage*. To avoid any accident, *Training Staff* and *Assessing Risks* activities remain important activities (see Figure 4): The *Training Staff* activity aims to test the impact of decisions at different level of the crowd aggressiveness by simulation and test, while *Assessing Risks* activity serves to detect continuously any risk in event venues (Stadiums, nearby parks, nearby streets, nearby parkings, nearby restaurants, nearby hotels,...etc ). In case of risk detection the *Put Precaution* action will be trigged.

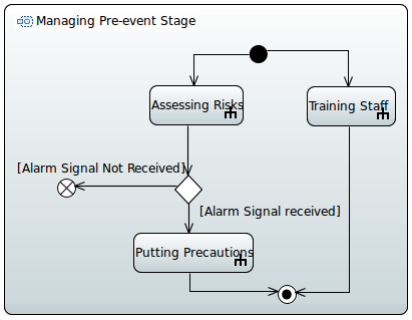


Fig. 4. Managing Pre-event Stage SysML Activity Diagram

A simplified process related to the *Assessing Risks* activity is shown in Figure 5. For *Assess Risks* activity, all event and venues information collected by the *Event Planning* activity constitute the entry parameters. For instance, the entry parameter *Venue Characteristics* includes *Weather characteristics of venue, geographical characteristics of venue, Floor characteristics, Venue capacity, ....etc*. The *Assessing Risks* activity collects continuously useful information about event venues. For instance, *Cross Flow Information* includes *cross flow location, causes, density, date and time* elements and *Hazard Item Information* includes *item type, utility, location, date and time* elements. Date and time are necessary elements since they allow to distinguish new information from the old one. Information is putted in *Data Stores* to be stored permanently, and used to detect risks. For the case of *Detect Risk of Crushing Between People*, *venue characteristics, entrance and exit characteristics, crowd characteristics, crowd density, cross flows informations and crowd behavior* give complementary information that allow making decision about presence or not of risks. In case of presence of risk, alarm signal is sent.

Sub-activities in the *Assessing Risks* activity are considered as actions, because we can directly match roles that handle them. So, the refinement of this activity is completed. The attribution of roles is done through simultaneous modeling of roles and processes where a role represents an abstraction of characterization of ideal behavior, which has realistic capabilities. Figure 6 shows an example of role modeling where the role *Observer*, which is responsible of *Sending Continuously Informations About Event Venues*, has the following capabilities: *Locating*, *Estimating Measures and Quantities*, *Detecting Anomalies*, *Predicting Changes*, *Tracking People*, *Communicating*, and *Observing Crowd*. Capabilities may be associated to settings. For instance, the capability *Communicating* is associated to the setting `communication_type`, which is an information about the type of the used communication medium. The capability *Observing* is associated to the setting *property_type* that informs if the venue is private or public, and to the setting *permission* which is a boolean that informs if there is a permission to observe. Constraints are expressed at the role level(see Figure 6).

The constraint related to the role *Observer* is a conjunction of three constraints. The first one expresses that it should be at list one common type of communication medium between an observer and a risk detector to communicate (another role having communicating capability). The second one expresses that observer must have permission to track people (protection of private property). And the last one requires that if the property is private, observer must have permission to observe. These constraints must be persistent in the SoS architecture in order to prevent the system architect from making choices that are incompatible with the decisions made by the application domain expert.

### *B. Architecture Modeling*

The abstract architecture blocks represents the roles responsible of actions. The communicational aspects are deducted from the activity diagram. Figure 7 represents the abstract architecture related to *Assessing Risks* activity where the *Observer* role collects information from event venues, send them to the *Risk Detector* role in charge of analyzing them and deciding if there is a risk or no. A signal alarm is sent to the *Coordinator* role if risk is detected. The constrained part of the blocks is taken from the definition of the corresponding roles.

The abstract architecture is used to get one of the possible concrete ones by replacing abstract items with concrete ones. Possible solutions are limited by the constraints defined on the blocks. For instance, Figure 8 shows a concrete architecture based on the abstract one presented in Figure 7.

Let's take example of assessing risks in a small park nearby a stadium where spectators wait for bus. *Monitoring Agent*, which is a human, can play the role of *Observer* since she/he has all needed capabilities and permissions to observe people. So She/he satisfies all the constraints needed by the played role. A *Qualified Staff* can play a role of *Risk Detector* since she/he has the competence requested and satisfy the communicating constraint with *Monitoring Agent*. In fact, There is several types of communication medium possible between them. Finally, The *Control and Command Coordinator* is the most appropriate person to play the role of *Coordinator*. Indeed, she/he also satisfies the communicating constraint with other constituent systems (emergency services, police,...) .

### *C. Discussion*

The intended purpose for Crowd Management SoS is to minimize the risk of accident in sport event, and in case of accident to minimize the victims number. In such a system

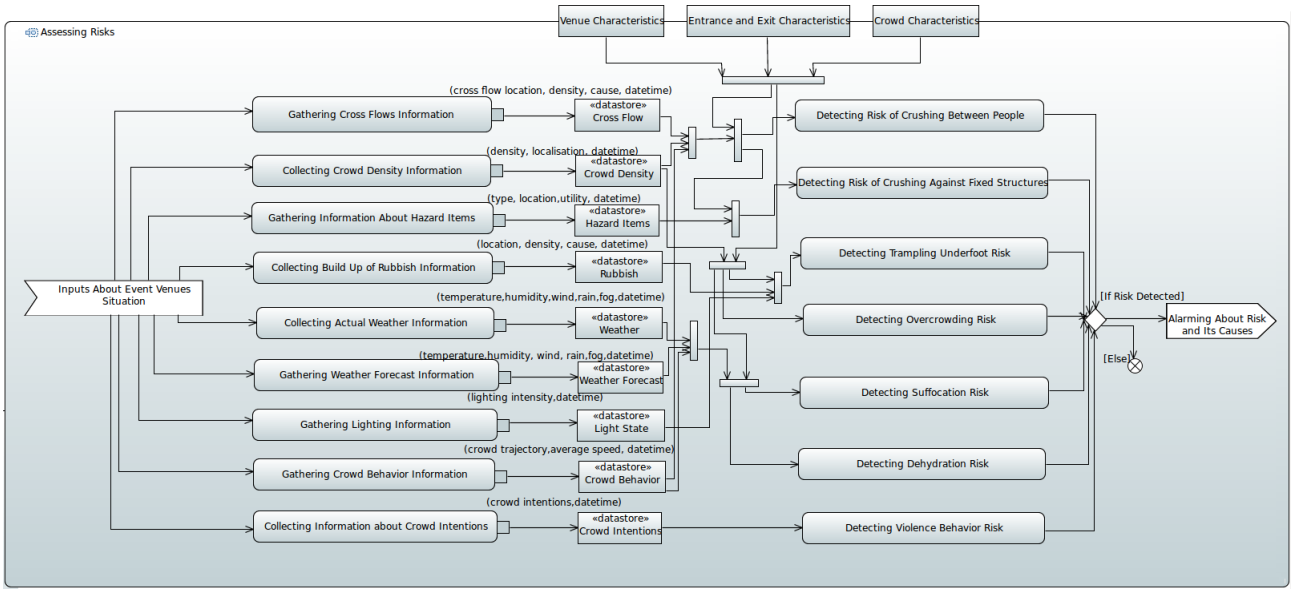


Fig. 5. Assessing Risk SysML Activity Diagram

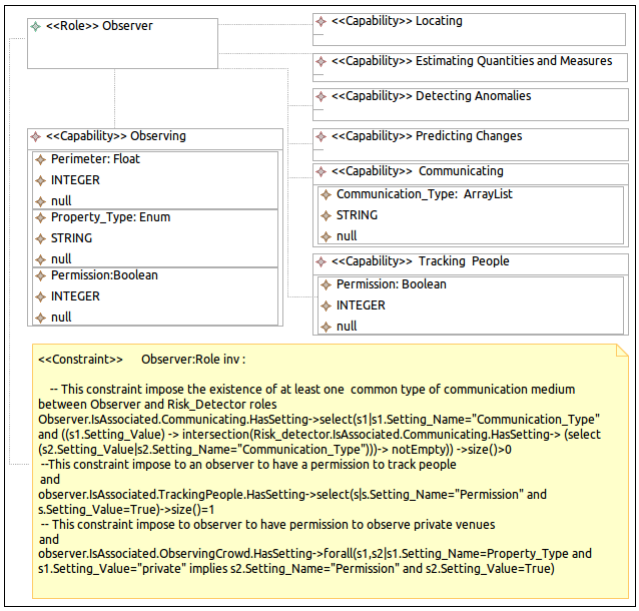


Fig. 6. Observer Role Modeling Diagram

it will be necessary to be able to respond to any eventuality. Knowing that the constituent systems often have managerial independence, it is possible that a system which the application domain expert would have thought, when defining the mission, may fail at a time in the life of the SoS. In addition, the system architect can respond to this failure by combining available systems to meet the expected role. For instance, which has been attributed to a human agent may also played by collaboration between a *camera* and a *CCTV Agent* systems. Indeed, this collaboration covers all capabilities expected by the *Observer* role. However, checking the constraint imposed by the role becomes more difficult to achieve.

Despite these drawbacks, the approach can be used as it is currently to provide a clear description of the SoS mission, the needed capabilities and produce the abstract architecture. We think that the global activity diagram may be used as a graph to open new possible investigations as the optimization of the redundant actions, the calculus of costs and the identification of critical paths .

## V. Related Work

Our work concerns mission and capability modeling in SoS. In this context, authors in [3] proposed mKAOS, an SoS mission description language. mKAOS extends KAOS [11] by specializing the KAOS elements and models to the SoS domain. In MKAOS, mission is a specialization of goal to SoS domain. The mission is refined with and/or operators until finding submissions that can be handled by a constituent system. In our approach, mission is an activity encapsulating the way to achieve the goal of the SoS. In mKaos, capabilities are classified into two types: operational capabilities and communicational capabilities. An operational capability represents an operation that a constituent system is able to execute. A communicational capability defines the connectivity intrinsic to the constituent systems, which enables them to cooperate to achieve global missions. In our approach, communication is a capability as well as an operational capability. Further, constraints can be used to impose a communication medium if it is necessary.

Authors of [7] propose an approach to model SoS throw capability specification of subsystems. The authors propose a taxonomy for capability concept. Thus, they broken down capabilities by type (Technical, Socio-technical Resources, Manual, Information Resources, Personnel Resources) and

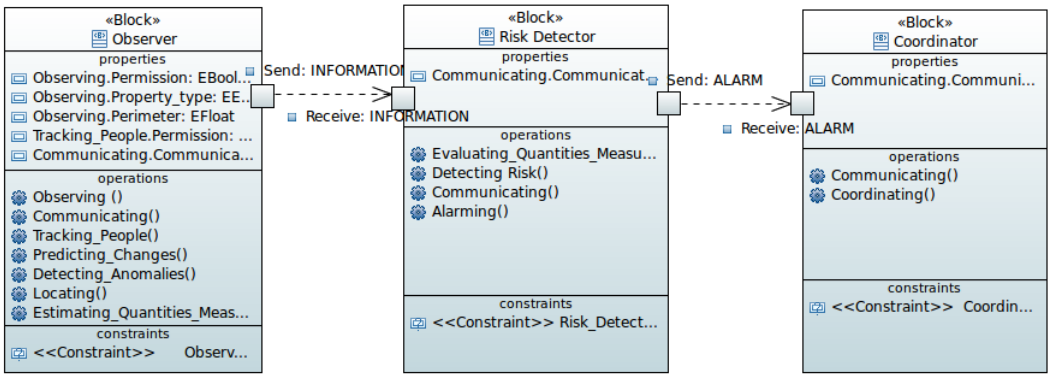


Fig. 7. Assessing Risk Abstract Architecture Modeling

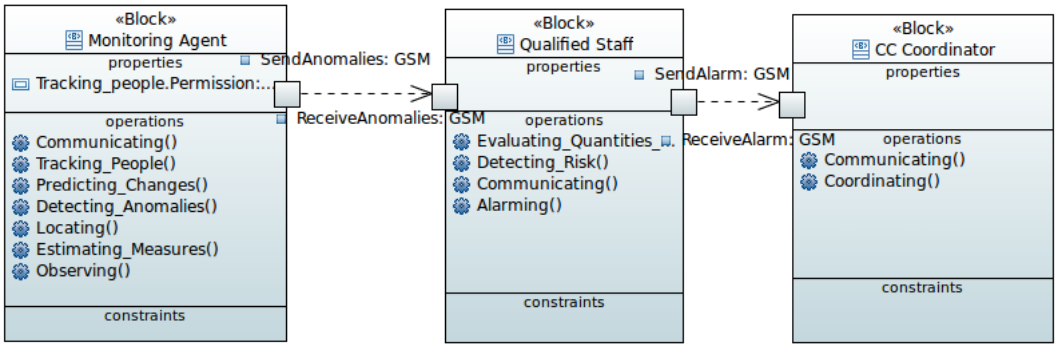


Fig. 8. Assessing Risks Concrete Architecture Modeling

maturity (Current, Legacy, Development). The authors propose a graphical notation to illustrate dependencies between subsystems throw capability dependencies. This work gives some interesting features about capabilities (type and maturity). However, this information is directly related to the constituent systems that will be used at run-time, whereas the SoS designer does not have a clear idea about them. This information is more useful during the creation of the concrete architecture of the SoS. In our approach, we consider the design stage and use the concept of Role, which is an abstract concept, to specify capabilities independently of the constituent systems that may exist during the execution stage.

The Department of Defense Architecture Framework (DoDAF) [12] is an architecture framework for the United States Department of Defense (DoD). In the DoDAF framework, there is several views, each of which is broken down into products and data: operational view, capability view, systems and services view, etc. The operational view aims to describe the tasks and activities, operational elements, and resource flow exchanges required to conduct operations while the capability view aims to describe the mapping between the required capabilities and the activities that enable those capabilities. The Ministry of Defense Architecture Framework(MoDAF) [13] is an architecture framework for the UK Ministry of Defense(MOD). Similarly to DoDAF, MoDAF provides a set of views that provide a standard notation to capture information about a business in order to identify ways to improve it. With our approach, we consider that the expert of the application domain is the most appropriate person to model activities and capabilities. As the domain expert is not necessarily an ICT expert, thus the proposed approach does not impose significant IT competences. Despite of being presented as a potential SoS modeling technique, DoDAF and MoDAF frameworks constitutes an in depth modeling approach which requires significant resources [7].

In [14], authors propose a conceptual model that can be used for SoS modeling. The main concepts used in this model are: roles, assumptions, capabilities, constituent systems. The authors consider that the capabilities of the SoS are totally or partially determined by the capabilities of its constituent systems. The authors do not address a way to model capabilities of constituent systems. They only addressed a list of relations on the constituent systems (communication, coordination, etc.). What is proposed corresponds to a bottom up approach for defining a mission while we propose a top down approach.

In [15], authors propose to model an SoS with MAS and to differentiate between the concepts of goal and mission.The SoS's goal is decomposed into subgoals, each of them is associated to a mission. The later is considered as a set of actions. In our approach we consider a mission as a set of activities that can be refined. The authors' approach focuses on the way of organizing the subsystems to satisfy the goal.

Our proposed meta-model aims to exploit the concept of role by focusing on the needed capabilities to achieve a mission. We do not focus on the existing systems. In this way, the roles can be attributed easily to actions. Thus, we promote a more abstract view of the mission.

In addition to the differences noted above, with existing work, we support that capability specification languages must also allow to express some constraints. The constraints may represent a condition of use of a resource, a condition of capability realization or a need of other capabilities. These concern the SoS business knowledge that must be respected during its design. We notice that no one of the cited work deals with that.

## VI. Conclusion

This paper provides an approach to support design activities in the SoS development process. We highlighted the importance of the mission concept for SoS design. The explicit and precise definition of a mission is a prerequisite to generate, control and adapt the architecture of an SoS. Thus, the idea supported by our approach is to allow the application domain expert to define her/his objectives and requirements in a formal way that will serve as a guide and controller of the choices proposed by the system architect during the design and evolution stages. To rigorously design the mission, we referred to a military standard which defines how a mission must be specified in the case of NATO organization. The retained concepts where described thanks to the SysML activity diagram.

With this approach we propose an intermediary between the requirements definition stage and the design stage. The goal is to create a safe transfer of information between these two stages. One can think that our approach resolve a problem that really exists only when the application domain is far from the IT domain. However, the separation between the definition specific to the application domain and that specific to the choices made by the system architect makes it possible to preserve the business decisions. Indeed, future evolutions of the SoS may be performed by another system architect, who needs to know the limits for the possible choices.

As a first extension to our approach, we plan to provide the system architect with a means to build a constituent system corresponding to a collaboration of existing constituent systems. Thus, we will respond to the problem raised by the discussion at the end of the case study section. The challenge is that this representation should facilitate the evaluation of the constraint imposed by the role in question.

In the approach presented in this paper, we have limited the transfer of information from the requirements definition stage to the design stage. In future work we want to allow the transfer of information also to the simulation stage. Simulation plays an important role in SoS engineering, mainly for managing emergent behaviors [16]–[18]. We believe that the rules that come from the definition of the mission can help to identify, if not avoid, emerging behaviors harmful to the good functioning of the SoS.

## References


[1] C. Tibermacine, R. Fleurquin, and S. Sadou, "Preserving architectural choices throughout the component-based software development process," in *Fifth Working IEEE / IFIP Conference on Software Architecture*, November.

[2] C. B. Nielsen, G. Peter Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "Systems of systems engineering: Basic concepts, model-based techniques, and research directions," *ACM Computing Survey*, vol. 48, no. 2, pp. 18:1–18:41, sep 2015. [Online]. Available: http://doi.acm.org/10.1145/2794381

[3] E. Silva, T. Batista, and F. Oquendo, "A mission-oriented approach for designing system-of-systems," in *10th System of Systems Engineering Conference*, ser. SoSE 2015, May 2015, pp. 346–351.

[4] M. Jamshidi, "System of systems engineering - new challenges for the 21st century," *IEEE Aerospace and Electronic Systems Magazine*, vol. 23, no. 5, pp. 4–19, May 2008.

[5] M. A. F. Standardization, *STANAG: Formats for ordres and designation of timings, locations and boundaries*, 9th ed., NATO, 2014.

[6] O. M. Group, "Systems modeling language v1.5," Object Management Group, http://www.omg.org/spec/SysML/1.5/, Tech. Rep. formal/2017-05-01, 2017.

[7] R. Lock and I. Sommerville, "Modelling and analysis of socio-technical system of systems," in *Proceedings of the 2010 15th IEEE International Conference on Engineering of Complex Computer Systems*, ser. ICECCS '10. IEEE Computer Society, March 2010, pp. 224–232. [Online]. Available: http://dx.doi.org/10.1109/ICECCS.2010.40

[8] C. Tibermacine, R. Fleurquin, and S. Sadou, "A family of languages for architecture constraint specification," *J. Syst. Softw.*, vol. 83, no. 5, pp. 815–831, may 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2009.11.736

[9] C. Group, "The compass project," http://www.compass-research.eu/, 2014.

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

[11] A. Van Lamsweerde, "Requirements engineering: From craft to discipline," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. SIGSOFT '08/FSE-16. New York, NY, USA: ACM, November 2008, pp. 238–249. [Online]. Available: http://doi.acm.org/10.1145/1453101.1453133

[12] U. D. of Defense, "Dodaf," dod-cio.defense.gov/Portals/0/Documents/DODAF.

[13] U. M. of Defence, "Modaf," gov.uk/guidance/mod-architecture-framework.

[14] D. Werner and A. S. Vincentelli, "A conceptual model of system of systems," in *Proceedings of the Second International Workshop on the Swarm at the Edge of the Cloud*, ser. SWEC '15. New York, NY, USA: ACM, April 2015, pp. 19–27.

[15] S. Jean Baptiste, "Conception and modeling of systems of systems : a multi-level mult-agent approach," Theses, Université de Lille 1, Dec 2013. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01099382

[16] R. Benabidallah, I. Cherfa, S. Sadou, and M. A. Nacer, "Situation/reaction paradigm for sos simulation," in *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, June 2017, pp. 48–53.

[17] E. Bonabeau, "Agent-based modeling: Methods and techniques for simulating human systems," *Proceedings of the National Academy of Sciences*, vol. 99, no. suppl 3, pp. 7280–7287, 2002.

[18] C. Parisi, F. Sahin, and M. Jamshidi, "A discrete event xml based system of systems simulation for robust threat detection and integration," in *2008 IEEE International Conference on System of Systems Engineering*, June 2008, pp. 1–8.