

System of Systems Engineering: Meta-Modelling Perspective

Charaf Eddine Dridi, Zakaria Benzadri, Faiza Belala

▶ **To cite this version:**

Charaf Eddine Dridi, Zakaria Benzadri, Faiza Belala. System of Systems Engineering: Meta-Modelling Perspective. Budapest, Hungary, Oct 2020, 15th SoSE 2020: Budapest, Hungary, France. hal-04227309

HAL Id: hal-04227309
https://hal.science/hal-04227309

Submitted on 3 Oct 2023

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

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

System of Systems Engineering: Meta-Modelling Perspective

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

***Abstract*—Systems-of-Systems (SoS) have emerged as a special type of large-scale complex systems and have been widely used in various fields. An SoS is a result of the integration of a number of pre-existing, independent complex software systems called Constituent Systems (CS). During the SoS Engineering process, the specific characteristics of CSs such as the operational and managerial independence, distribution and the emergent behaviour make the modelling of their structure, relationships and interactions a complex task.**

On the other hand, Model-Driven-Architecture (MDA) is still one of the most promising software architecture approaches, it provides a method which simplifies SoSs complexity by increasing their abstraction level. The aim of this study is to introduce a software description meta-model that can be used for modelling SoSs. The meta-model highlights the importance of Goals, Roles, Capabilities and CSs concepts for SoS design. These concepts will be translated into an abstract architecture where their composition and relationships are well-defined. The main contributions of this paper are: (1) a Meta-Model for System-of-systems called MeMSoS, (2) its supporting tool and (3) an illustrative case study of an Aircraft-Emergency-Response-SoS (AERSoS).

***Index Terms*—Modelling, SoS, CS, SoSE, MDE, MDA, Meta-model.**

I. Introduction

In recent years, the technological development of large-scale systems has experienced significant growth thanks to the latest innovative advancements in various fields. Large-scale systems have become more and more complex, they are built by integrating independent sub-systems from different providers to create a particular class of systems called System-of-Systems (SoS). Thus, an SoS is composed of a set of systems that interact to provide a unique capability that none of the Constituent-Systems (CSs) can accomplish on its own. These dispersed, autonomous and heterogeneous CSs have also a degree of managerial and operational independence, behave in a cooperative manner and interact with other CSs which in turn can be complex SoSs in order to accomplish their local goals as well as the SoS global goal.

SoS Engineering is the process of integrating existing systems to create new functionalities and capabilities, the architectural design is the most crucial activity in the SoS Engineering (SoSE), it involves two main activities:

- **Systems selection:** SoS application domain experts choose the required systems that have the necessary capabilities to integrate them within the SoS.
- **Conceptual design:** design the mechanisms to integrate them and facilitate their interactions.

Thereby, in the SoSE context, we argue that the MDA approach can be leveraged to model SoSs architectures and abstract their design-related problems. With this aim, the adoption of Meta-Modelling techniques to design SoSs architectures is an intrinsic step in MDA approach. SoSs Meta-Modelling (model of models) provides a common, unambiguous, structured and accurate SoS architectural model.

A. Context and problematic

SoSE differs from the ancient Software Engineering (SE) in that CSs, goals, roles, capabilities and interactions are considered as first-class entities. The complexity of SoSs arises mainly from the interaction of large-scale complex systems and their internal complexity (operational and managerial independence, evolutionary, emergent behaviours, and dispersion).

To reduce the complexity of such types of systems, we adopted an MDA approach for modelling and describing SoSs using a meta-model that includes these entities and their relations. This solution is based on the strong relationships between the main components of a CSs in SoS, namely:

- **SoSs and their CSs:** to present the SoS hierarchical organization.
- **Different roles:** to specify different interactions between Roles.
- **Roles and Capabilities:** to identify the required Capabilities of a Role.
- **Local and global Goals:** to specify the SoS global goal and its sub-goals.

The proposed meta-model presents a comprehensive architecture for SoSs and helps different SoSs stakeholders to understand certain degree of SoS challenges and its potential solutions at the architectural design activity.

B. Related works

The SoS research domain has experienced a fast growth as shown by the increasing number of conferences, workshops and scientific papers [1]. Despite all these researches, a number of studies and literature reviews [2] emphasize that the research in the SoS engineering process is still at early

stages and there is still a need for more contributions for SoS architecture and modelling.

In [3] the authors propose a process-based model to support design activities in the SoS development process by introducing some concepts like mission and roles. The proposed approach serves as a guide and controller of the choices proposed by the system architect during the design and evolution stage. Authors of [4] presented a viewpoint-driven approach to design SoS by adopting a SysML profile. This study was an extensive discussion to [5], the authors focused on the relevant concepts of the conceptual model. Moreover, they showed how these concepts are coded into a SysML profile. Authors in [6] identified a model of SE which provides a framework for supporting the systems engineer in this SoS environment. This study reviews the core elements of SoS which provide the context for the application of basic SE processes adapted for the challenges of SoS.

Whilst different MDA based architectural design methods have been adopted to manage SoSs problems, little attention has been devoted to SoSs main entities (CSs, Roles, Capabilities and Goals) organization and their possible interactions.

C. Paper organization

This paper is organized as follows. Section 2 describes the state of the art in SoS and MDA. Then, a motivating example for SoS, the aircraft emergency response system, is presented in section 3. In section 4, we give the principle of our model-driven approach for designing SoS; we define the meta model "MeMSoS" for SoS and then we describe its essential functionalities. Sections 5 and 6 are devoted to the validation of our approach through an illustrative case study and a supporting tool. Finally, Section 7 concludes the document with some perspectives.

II. Background

The starting point in our work is investigating and understand the characteristics of SoS, the CS including their relationships and SoS architectural principles. Using an MDA approach, we aim to raise the level of abstraction in the modelling of SoSs and increase the automation of their development to produce as a result, a practical meta-model for representing and describing SoS architectures.

A. System-of-Systems

We should note that there is no formal definition for what SoS is. Thus, many researchers have written papers giving various attempts to define and characterize SoS [7]. Most of the provided definitions refer to an SoS as a set of useful and independent sub-systems that interact to create a large-scale system and perform unique capability that cannot be provided by any of the constituent-systems [8] [9] [10] [11].

According to [12], a successful system of systems relies on the following fundamental parts: (1) characteristics which distinguish SoS from large but monolithic systems, (2) architectural principles for the construction of SoS, (3) systems built from communications.

Maier [10] [12] [13] has defined five key characteristics to distinguish very large and complex systems from real systems of systems:

- **Operational independence of elements:** every component-system works independently to achieve its own individual goals and collaborating with the other CSs to accomplish the SoS global goal.
- **Managerial independence of elements:** every CS belongs to a specific company and it can be managed independently by the company to which it belongs.
- **Roles and Capabilities:** to identify the required Capabilities of a Role.
- **Geographic distribution of elements:** The CSs of the SoS are dispersed, it means that the components can exchange only information with one another and not substantial quantities of mass or energy.
- **Emergent behaviour:** SoS global functionalities and purposes do not remain in any single CS, these behaviours are emergent properties of the entire SoS and cannot be localized to any component system.
- **Evolutionary development:** The system-of-systems does not appear fully formed, their structures, functions and purposes are subject to several requirements, functionalities and evolutions (insertion, modification or suppression) of its CSs.

Beyond this, SoS can take four different types. These types are primarily based on the relationships among the CS in the SoS [14], the levels of responsibility and authority overseeing the evolution of the SoS [11]. Maier has identified three types "Directed, Virtual and Collaborative" and the fourth type "Acknowledged" was found in a number of SoSs in DoD projects [14] [15] [16]:

- Directed: an SoS with a central managed purpose and central ownership of all CSs;
- Virtual: lack of central purpose and central alignment;
- Collaborative: voluntary interactions of independent CSs to achieve a goal that is beneficial to the individual CS;
- Acknowledged: independent ownership of the CSs, but cooperative agreements among the owners to an aligned purpose.

B. MDE, MDA and Meta-model

Following the object approach of the 80s and its principle of "everything is object", software engineering is now moving towards Model Driven Engineering (MDE) and the principle of "everything is model" [17], MDE has made several significant improvements in the development of complex systems. The authors of [18] consider that the implementation process has become a classic concern for software developers and that MDE has given them a new concept that changes their focus from code to model.

Therefore, the MDE aims more radically, to provide a large number of models to separately capture, describe and organize each of the concerns of different stakeholders.

The central concept of the MDE is the concept of model for which there is no universal definition to date. Despite that,

many studies [19] [20] [21] agree on a relative consensus that a "model" is an abstraction of a system representing some aspect of it based on a specific set of concerns, constructed with an intended goal in mind and must be able to be used to answer questions in place of the actual system. The model is related to the system by an explicit or implicit mapping.

The Object Management Group's MDA approach is the specialization of MDE, it aims to provide a new way of designing systems by separating the neutral business of the company from the aspects linked to the platform.

The concepts of system, model and meta-model as well as the relations which link them, represent the basic principles on which the MDE and MDA are based. OMG standardization has distinguished four abstract layers that may be instantiated to any application domain. As illustrated in Fig. 1, these layers in the context of SoS may indicate:

- **M3:** the Meta Object Facility (MOF) represents the most abstract layer.
- **M2:** the meta-model we propose complies with MOF and represents a meta-model for SoSs.
- **M1:** the Model conforms to the proposed meta-model and represents this various SoS models.
- **M0:** the SoS implementation (executed code) represents the lowest layer.

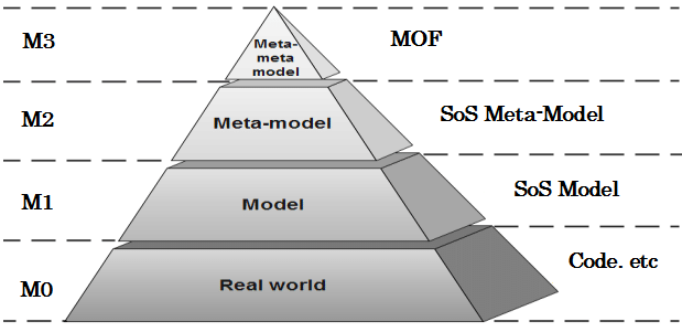


Fig. 1. Four-layer meta-modelling infrastructure of OMG.

The two main artefacts of MDE are models and meta-models. This later may be in different types (Computation-Independent Model (CIM), Platform-Independent Model (PIM), Platform-Specific Model (PSM), and Platform-Description Model (PDM)) [22] [23].

A CIM model represents the needs of the client and the requirements of the system, its aim is to help better understand the problem by describing the services that an application can offer in its environment. The components are not included in a CIM. For instance, a CIM for SoSs may not have any information about CSs (it may just include goals of an SoS and roles to be played).

A PIM is a model of analysis and design, it is an abstract model independent of implementation technology and any execution platform [22]. A PIM for an SoS can include CSs, roles, goals, capabilities and associations among each other.

A PSM is a model that takes into account the technological platform at an abstraction level. the exact model of an SoS can be considered as one of the PSMs of the related SoS. For instance, The model presents the implementation details of the SoS and includes the entities of the SoS, which are given as instances of the meta-classes defined in the Platform-Specific Meta-Model.

A PDM models the platform on which the system will run. It describes the different functionalities of the platform (file, memory, BDD systems...etc.).

From a general point of view, we call model transformation any program whose inputs and outputs are models. A transformation expresses structural correspondences between the source and target models. These structural correspondences are based on the meta-models of source and target models [22].

III. Motivating Example: Aircraft Emergency Response System-of-Systems (AERSoS)

Air traffic has become widespread since the 1970s and it continues to grow; the International Air Transport Association (IATA) predicts 8.2 billion air travellers in 2037 [24]. Aviation accidents involve not only passengers and flight crews, but also a vast territory, around the airport with a large number of residents who, in some cases, have paid a heavy price in terms of their lives. As an example, the painful aviation accident on April 11, 2018 in Algeria causes 257 dead [25]; where an army aircraft of the Ilyushin-76 type crashed just after a short take-off from the Boufarik Air Force Base.

From an IT point of view, frequent problems causing aviation accidents are mainly due to a misunderstanding of the interaction and communication between aviation constituent systems. Researchers must pay particular attention to the idea that the aviation is a "System of Systems" [26] and the impact of such approach at conceptual design stage [27]; since aviation system is bound by a set of distinct subsystems (aircraft subsystems, air traffic management/control subsystems, etc.) and each of these subsystems has managerial and operational independence (provided by different companies).

On this basis, we propose in this paper an MDA-based approach for System-of-Systems to illustrate through this case study: the "Boufarik Air Force Ilyushin Il-76 Crash" scenario [24]. We assume that the problem in this air crash occurred at the engine level and due to poor communication (interaction) between the constituent aviation-systems. Thus, we identify the "Aircraft Emergency Response System-of-Systems" (AERSoS) which maximizes aircraft safety by minimizing critical situations; this by providing the required emergency response and reacting to the aircraft critical problems. The AERSoS interacts with many independent CSs which are deployed in the aircraft or even those which are dispersed. AERSoS constituent systems are composed through communication links as described in Fig. 2.

At aircraft level, approximately hundreds of sensors installed on all parts (turbines, oil, lubricant ... etc.) of the aircraft subsystems that communicate with each other and collect data on the aircraft's performance and flight conditions

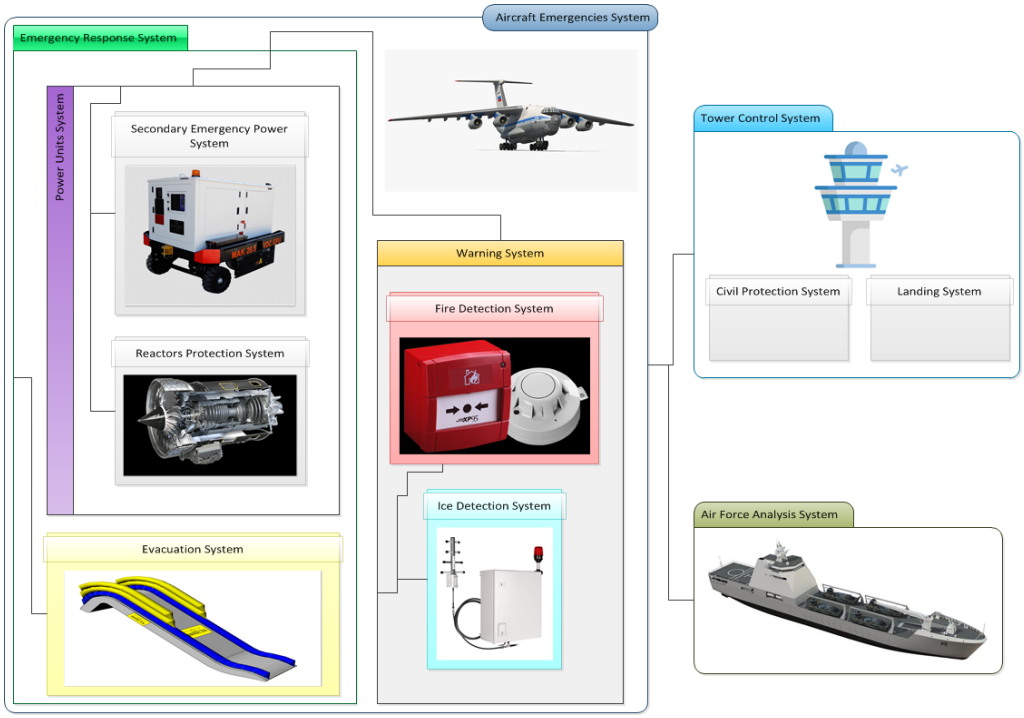


Fig. 2. Aircraft Emergency Response System-of-Systems (AERSoS).

-according to Cisco [28]. We classify these sensors into: (1) those responsible for collecting data on high and unusual temperature in turbines, oil, lubricant...etc., and (2) those which detect the gases emitted by potential fires.

A warning SoS signals a problem that occurred with the aircraft at a given position. This system constituted of sub-systems that interact directly with the installed sensors; In the case that these sensors detect something wrong (fire, flame, gases...etc.), another autonomous system (which takes on the task of verifying the validity of the indications and evaluating the risks) must:

- Signals a state of danger on the plane and informs the flight crew.
- Provokes the initial response systems which perform functions that are intended to mitigate the emergency: (1) systems designed to isolate damaged engines and suppress fires (2), systems that generate secondary power (ex: hydraulic and electrical power, liquid oxygen...etc.) and (3) Passenger rescue system (provide oxygen, life jackets..etc).
- Interacts with the control tower to try to clear the runway and airway for an emergency landing at the nearest airport using GPS, as well as airport fire-fighters to ensure the first aid.
- Interacts with the Air Force Analysis System (AFAS) for the analysis of the technical report containing the details of the missing parts to be sent to maintenance for a quick repair upon landing.

As we explained, this SoS like others requires special attention to describe its complex behaviour through the arrangement of its subcomponents, from several points of view. In this paper, we will focus on its architectural abstract specification. We start by showing the use of the MeMSoS meta model in order to identify the different architectural entities of this system, in particular the interactions that may exist between them. Then, we evaluate this architectural model using a disaster scenario. The proposed approach targets the complex and distributed system architectural specification phase, where a way to obtain a high-level designing and reasoning of such systems behaviour analysis after their deployment.

IV. A Model-Driven Approach for SoS Design

A plethora of approaches are actively investigated by the research community to support design and analysis of SoSs. However, there is a deficiency concerning the definition of suitable modelling and specification notations as well as supporting formal verification. In this paper, we argue that MDE can have a performed impact in engineering of SoSs. In the following, we present the first steps of our approach and the technical framework which will be developed to support the engineering of such complex and heterogeneous systems.

A. Principle

an SoS is a hierarchically defined composition where the super-element (the SoS) is made of a set of CSs. Each CS may be itself an SoS. The bottom-elements of the hierarchy are atomic CSs that have no CSs. CSs are distributed autonomous entities composed of roles and capabilities that cooperate and coordinate with other CSs to accomplish their local goal as well as the SoS global goal.

The global goal of an SoS can be met only if CSs exchange enough information. Since CSs are independent, distributed and are not able to exchange mass or energy, communications are the principle that allows the exchange of information. Thus, modelling an SoS which is functionally adequate (which accomplishes the global goal) is achieved by designing its CSs with a relationship-driven interaction.

Our findings led us to propose a new structural definition of SoS, primarily based on Links concept:

Definition 1: SoS = {CS, G, R, C, L} where:

CS: is set of Constituent-Systems, a CS is any complex autonomous system that acts following its own list of Capabilities and plays certain Roles.

R: are CS Roles describing the CS ideal behaviour through the gathering of the required Capabilities to play the Role needed to accomplish the SoS global Goal.

G: is a Goal, an instance of the SoS Roles, that represents local objectives of a CS and Global objectives of SoS. A Goal can be simple or complex.

C: represents the functions provided by a CS in a specific Role to the wider needs of an SoS. A CS Role can execute several Capabilities (C) concurrently.

L: are dynamic Links, forming possible interactions and communications among different CS, R, G and C.

CSs are developed as reusable entities, they must be sufficiently specific to be easily identified, employed and reused in an SoS. The originality of the proposed structural definition of SoS **(definition 1)** lies in the various types attached to the Link main constituent of an SoS:

1) CSs Links: describe the association of the CSs, These Links are mostly devoted to represent an SoS as a strong organized composition.
2) Roles Links: express the interactions between different Roles defined on CSs.
3) Capability-Role Links: binding the Roles with the their required Capabilities.
4) Goals Links: describe the relations between a Goal and its corresponding sub-goals.

This abstract and generic definition will be associated to a meta-model within the MDA approach. Thus, this meta-model defines a logical structure of elements involved in any SoS models. While the notation describes the diagrammatic notations used to represent visually the models. The use of this approach in designing SoS architectures permits to describe all the SoS features at a same high-level of abstraction.

B. MeMSoS: Meta-Model for System-of-Systems

We have chosen the class diagram notation to represent graphically the proposed meta-model MeMSoS (Meta-Model for System-of-Systems). UML Classes, aggregations, inheritance...etc. are used to represent elements and entities of this meta-model (see Fig. 3).

The proposed MeMSoS is defined based on the key consideration that an SoS is more than simply a collection of connected CSs. Therefore, the main purpose is that it has been designed in order to explicitly model and represent all SoS architectural elements. MeMSoS gives a particular attention to the hierarchical composition of CSs and highlights the importance of CS, R, C, G and L in the whole SoS architecture. This is quite evident from Fig. 3.

As shown, MeMSoS meta-model defines SoS as a combination of Sub-Systems which are composed of G (Goals), C (Capabilities), R (Roles) and their L (Links). The class SoS is the core of the proposed meta-model, it represents an SoS as an aggregation with the cardinality of two-to-many of Sub-Systems where each Sub-System class represents a generalization of two different systems classes (SoS and CS).

In addition, the Class Sub-System has three one-to-many aggregations expressing Capabilities, Roles and Goals. It is associated with these classes with multiplicity one-to-many since any Sub-System could have Capabilities, Roles and Goals.

The class Roles in turn is associated with one or more Relation class and has two dependencies (gather) and (accomplish) to Capability and Goal classes respectively.

Finally, the class Relation is used to model the possible interactions among different Roles; the source and target associations represent respectively both initial and final Roles.

C. MeMSoS Features

MeMSoS groups four main SoS modelling characteristics according to two system viewpoints called structure and interaction.

1) **Hierarchical organization:** the MeMSoS classifies systems by their position in a three-level hierarchy from simple atomic CSs, via a collection of interacting CSs forming a single SoS which in his turn may be a Sub-System to another super-SoS product up to an array composed of several geographically dispersed systems of varying types (CSType and SoSType classes) interacting to achieve a common Goal.
   The SoS hierarchical organization occurs in SoS, Sub-System, CS classes and their associations. In other words, the aggregation defines both the highest level SoS (considered as the summit of the hierarchy) and its constituent Sub-Systems. The generalization indicates whether the components constituting the whole SoS are themselves very-complex (SoS) or atomic CSs.
2) **Goals organization:** Goals of SoSs in MeMSoS can be organized as a tree structure in which high level Goals may be realized through the combination of lower level Goals; i.e. The global Goal of an SoS is a complex Goal that can be decomposed into sub-Goals, every sub-goal corresponds to a Sub-System and the criterion for stopping Goal decomposition is when a sub-goal corresponds to a local Goal of an atomic CS.
3) **Roles interactions:** a Role can participate in the achievement of several local Goals at the same time, accordingly it can interact with several other Roles and fulfill other Goals in parallel. Reuse of an existing Role by using different types of relationships reduces the

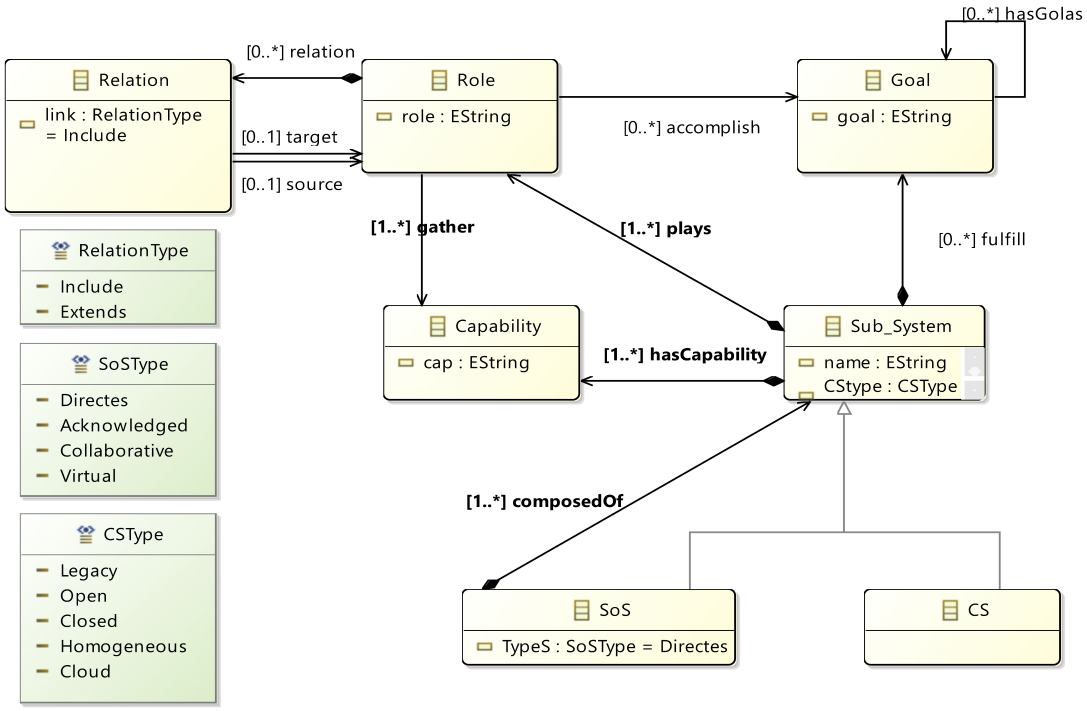


Fig. 3. Meta-Model for System-of-Systems (MeMSoS).

overall effort required to achieve the global Goal of an SoS. The Role Relation class specifies the possible associations between multiple Roles :

- Include: if R1 include R2, then R1 live and die with R2, i.e. R1 cannot stand by itself. Include relationship shows that the Capabilities of the included Role is part of the including Role.
- Extends: if R1 extends R2, then R2 is independent of R1, i.e. As the name implies it extends the extended Role and optionally adds more Capabilities to it.

4) **Roles, Capabilities and Goals:** MeMSoS provides the properties of interoperability and heterogeneity that allow the unrestricted sharing of Roles and Goals between different Sub-Systems in the SoS level and Capabilities between Roles in a CSs level, this can reduce the cost of creating new Roles, Goals and Capabilities by allowing existing Sub-Systems to be reused in multiple ways for multiple purposes:
The specification of Roles requires identifying the Capabilities for which the Role is responsible to achieve the Goal and the same Capability can appear in different Roles in the system level.

Obviously, our contribution here is related to the levels of M1 and M2 of the meta-modelling infrastructure of OMG (Fig. 1). The resulted MeMSoS may serve to define several mapping or transformation. We limit our selves in this article to deduce models for SoSs respecting this meta-model definition.

V. AERSoS Example Application

We show in the following section, how we apply our approach to deal with the main limits of the AERSoS architectural design.

A. Instantiation of MeMSoS

In this section, we illustrate the AERSoS architectural model to show the MeMSoS's meta-modelling ability to design the software architecture of such an intensive and complex system. An AERSoS exhibits most of the dimensions of our MeMSoS. The challenge is to model the whole AERSoS and show its associated entities (CSs, R, G, C and L).
In order to instantiate MeMSoS and create an SoS model, we provide a graphical editor which can be used to design AERSoS model. To do this, we have created representations for each entity (classes, relations, aggregations, etc.) of MeMSoS, this is shown in Fig. 4. In this model we find a set of instances of the MeMSoS entities (see Table1), for example:

- Instances of the class SoS (represented as rectangles with purple and dashed bordures); e.g. Aircraft Emergencies SoS, Tower Control SoS, Warning SoS…etc.

- Instances of the class CS (rectangles with solid green bordures); e.g. Fire Detection System, Evacuation System, Fire Force Analysis System…etc.

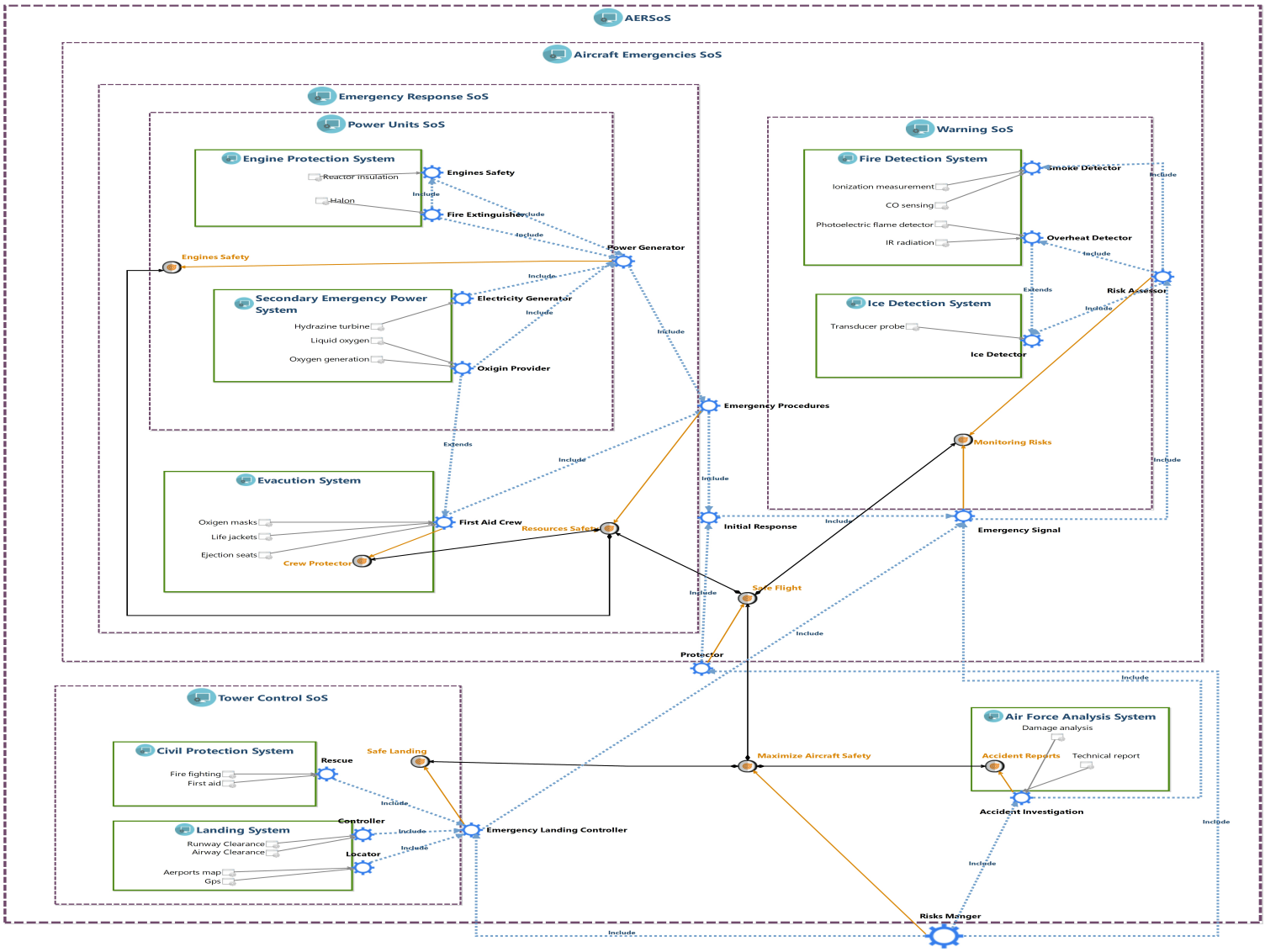


Fig. 4. Aircraft Emergency Response System-of-Systems model.

- Instances of the classes Role (border nodes in the shape of blue gear), Goal (orange nodes) and Capability (small nodes inside systems).

- Dynamic links between these entities are instantiated using various colored arrows.

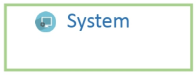
B. *AERSoS modelling*

1) **Hierarchical organization in AERSoS:** MeMSoS allows specifying the hierarchy of the *AERSoS* and its constituents, which are fully independent, autonomous and complex; This feature is presented in *AERSoS* model as a set of overlapping rectangles. For instance, in Fig. 4, the *AERSoS* includes *Tower Control SoS*, *Force Analysis System* and *Aircraft Emergencies SoS* which in turn contain *Warning SoS* and *Emergencies Response SoS*...etc.

2) **Goals in AERSoS:** the SoS's constituents also fulfil local Goals which can be sub-goals of another global Goal. Their combination form a tree structure where the more global Goals near the SoS and the sub-goals near the CSs. Thus, relations (orange links) between Goals denote sharing of the same common Goals (the orange diamond shape refers to global Goals). In Fig. 4 for example, the global Goal *Maximize Aircraft Safety* has the sub-goals *Safe Landing, Accident Reports* and *Safe Flight*. The Goal *Safe Flight* in turn has two sub-goals called *Resources Safety*, and *Monitoring Risks*...etc.

3) **Roles interactions:** The Roles of a CS define how it interacts with other CSs, in addition, they ensure the concepts of heterogeneity and interoperability of CSs by abstracting the type of their relationships. A Role link expresses its possible needs to the Capabilities of

TABLE I
ENTITIES INSTANTIATION.

Entity	Instance
SoS	
CS	
Role	
Goal	
Capability	
accomplish	
hasCapability	
hasGoal	
Relation	

another particular Role. The latter can even be a Role for other systems.

In AERSoS model, the Roles are presented as border nodes (blue gear icons) on the rectangles of systems. Roles links (dotted blue arrows) are also modelled as stereotyped links to specify the sense and the type (see section IV.B) of interaction.

In what follows, we briefly introduce examples of Roles interactions in the *AERSoS* model:

Risk Assessor include *Smoke Detector*, *Overheat Detector* and *Ice Detector*; means that the Role *Risk Assessor* can not stand alone and will not be played until Roles *Smoke Detector*, *Overheat Detector* or *Ice Detector* are played. In the same way, the *Warning SoS* can not play the Role *Emergency Signal* until finishing the Role *Risk Assessor*.

According to these interactions, *Warning SoS* will prevent any false indications or fake signals may occur at the level of its CSs or their Capabilities. On the other hand the constituents *Fire and Ice Detection Systems* will collaborate to fulfill the Goal *Monitoring Risks*.

Emergency Procedure includes *Initial Response*; means that the first Role depends on the second which, in turn, was included by the previous Role *Emergency Signal*. When something goes wrong, *Warning SoS* declares a state of emergency. The *Emergency Procedures*, *Emergency Landing* and *Accident Investigation* Roles must be informed so that they can independently take the necessary reactions.

When an engine is lost, power on the Aircraft will drop quickly. In order to provide the energy to both the aircraft and the crew, the Role *Emergency Procedure* interacts with *Power Generator* and *First Aid Crew* Roles which are played respectively by Power *Units SoS* and *Evacuation Systems*. (power supplying is ensured by the two Roles *Electricity and Oxygen Generators*.

A stereotyped 'extends' type relationship is shown in the model which relates between the two CSs *Secondary Emergency Power System* and *Evacuation System* via their own Roles: *Oxygen Provider* extends *First Aid Crew*; means that the first Role is not always necessary, but that it may be in certain situations when the crew needs more oxygen.

In response to a critical situation, an emergency landing is made by *Tower Control SoS* which play the Role *Emergency Landing Controller* to clear the runway and the airway providing a *safe landing*.

4) **Goals, Capabilities and Links:** The capabilities in *AERSoS* can be any entity (sensors, operations, devices, services) needed by a Role to fulfill a certain goal. In our model, Capabilities are represented in forms of small nodes inside their systems, they are directly associated (with a gray arrow) to their corresponding Roles so that they can fulfill their local Goals.

The *AERSoS* global and sub Goals are also represented as small orange nodes. Each Goal can be fulfilled by one or multiple interacted Roles. The links between Roles and Goals are modeled using orange arrows. Table 2 shows examples of interactions among some Goals Capabilities and Roles:

VI. IMPLEMENTATION AND VALIDATION

To provide tool support and proof of implementation concept of our approach, we intend to realize a complete prototypical tool chain to edit and analyze dynamic behaviour of SoSs. In this section, we are interested by an SoS graphical editor.

A. Graphical Editor

Our approach is part of the MDA context. Therefore, we wanted our framework to be based on standards in this domain and a chain of tools. These tools should be able to describe the meta-model's abstract syntax and capable of semi-automatically generating its corresponding graphical editor (concrete syntax).

This chain essentially includes two tools running in an Eclipse IDE: (1) EMF (Eclipse modelling Framework): uses Ecore modelling Language to specify the abstract syntax of the meta-model. (2) Sirius uses the transformation of models for the automatic creation of graphical editor. Both were developed primarily as Eclipse modelling Projects by Obeo.

The graphical editor corresponding to our MeMSoS could now be used to create new SoS model instances taking into account its specific characteristics and interactions aspects. Fig. 5, shows the tool palette of tools facilitating the creation of the SoSs components.

TABLE II
SUMMARY OF AERSOS COMPONENTS.

SoS	Sub-Systems(CSs)			Roles	Capabilities	Goals
Aircraft Emergencies SoS	Emergency Response SoS	Power Units SoS	Engines Protection System	Engines Safety, Fire Extinguisher	Reactor insulation, Halon	Engine Safety
			Secondary Emergency Power System	Electricity Generator, Oxigen Provider	Hydrazine turbine, Liquid Oxygen, Oxygen generator	Energy Provider
		Evacuation System		First Aid Crew	Oxygen masks, Life jackets, Ejection seats	Crew Protection
	Warning SoS	Fire Detection System		Smoke Detector, Overheat Detector	Ionization measurment, CO sensing, Photoelectric flame detector, IR radiation	Fire Sensing
		Ice Detection System		Ice detector	Transducer probe	Ice Sensing
Tower Control SoS	Civil Protection System			rescue	Fire fighting, First aid	Crew Safety
	Landing System			Controller, Locator	Runway clearance, Airway clearance, Airports map, Gps	Safe Landing
	Air Force Analysis System			Accident Investigator	Accident report, Damage Analysis, Technical report	Accident Report

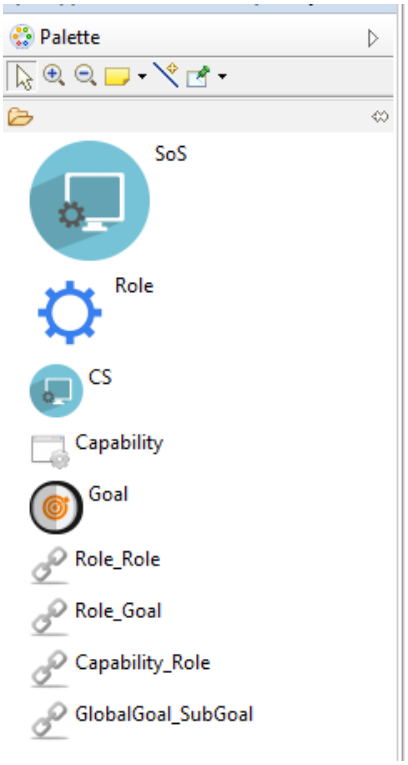


Fig. 5. SoSs graphical editor.

B. Consistency checking

Consistency check in a MeMSoS models refers to the conditions of use of the palette tools(Fig. 5), i.e. SoS, Cs, Goals, Capabilities and Links that a set of interacted CSs must satisfy in order to form a meaningful SoS model. Some of MeMSoS models related consistency checks performed are:

- Ensure whether interacted CSs match in terms of composition, e.g. each CS appearing in the MeMSoS-model belongs to a specific SoS and it cannot in any case have sub-systems.
- Automate the task of linking different nodes: there are four different arrows that connect different kinds of nodes, each arrow connects a specific pair of nodes.

Fig. 6 shows a screen-shot of the AERSoS model instance. The model shows a root object representing the whole AERSoS. This SoS has constituents, which represent AERSoS sub-systems (sub SoSs or atomic CSs), whose components represent the Roles, Goals, Capabilities and Links of these sub-systems.

VII. CONCLUSION

According to MDA and MDE principles, models and meta-models may play first-class citizens in SoS engineering. In this paper, we have given a model-driven approach to deal with the definition of independent and heterogeneous constituent systems of an SoS and their possible interactions in terms of architecture, roles, goals and capabilities.

The resulted meta-model has been used to derive SoS models that allow graphical and ambiguous specification of such systems. Besides, we have illustrated through the AERSoS case study, how we can model a disaster scenario of this critical SoS.

Since MDE components need to be investigated into software development environment, we have thought about developing such a tool and we have started with a graphical editor directed by the syntax of SoSs which is expressed by MeMSoS.

Several lines of work are planned to be addressed in the near future. The authors are working on the definition of ”Domain Specific modelling Language” to represent the various facets of SoSs, including their behavioural aspects. Furthermore the authors are working on how they use model transformation languages (ATL for instance) to transform models derived from MeMSoS to other models, particularly formal ones (Maude models), to achieve formal verification of certain properties, and how to integrate other tools in order to provide a complete tool support implementation of their approach for modelling and reasoning about SoSs.

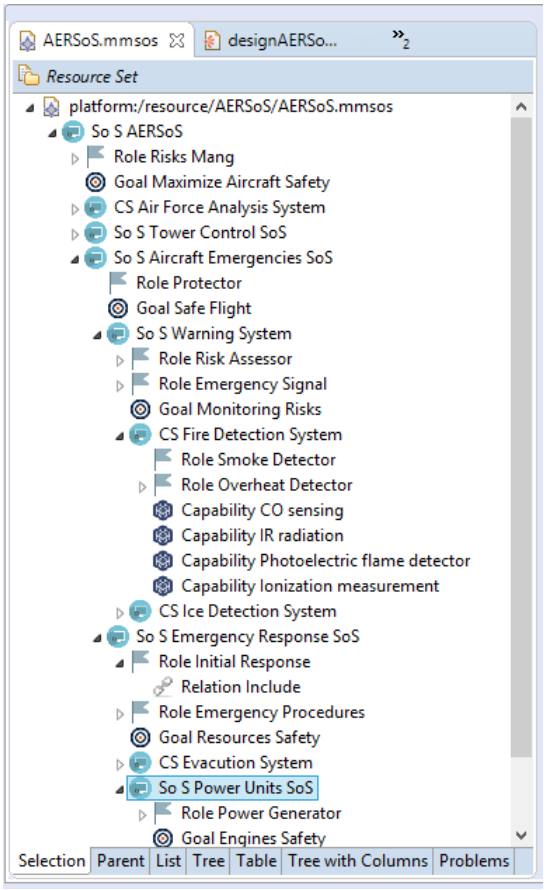


Fig. 6. AERSoS EMF model instance.

REFERENCES

[1] Guang-rong You, Xiao Sun, Min Sun, Ji-min Wang and Ying-wu Chen (2014). Bibliometric and social network analysis of the SoS field. 2014 9th International Conference on System of Systems Engineering (SOSE).

[2] Kotov, V. (1999). Systems of Systems as Communicating Structures. Object-Oriented Technology and Computing Systems Re-engineering, 141-154.

[3] Cherfa, I., Sadou, S., Belloir, N., Fleurquin, R., & Bennouar, D. (2018). Involving the Application Domain Expert in the Construction of Systems of Systems. 2018 13th Annual Conference on System of Systems Engineering (SoSE).

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

[5] Mori, M., Ceccarelli, A., Lollini, P., Bondavalli, A., & Fromel, B. (2016). A Holistic Viewpoint-Based SysML Profile to Design Systems-of-Systems. 2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE).

[6] J. Dahmann, J. Lane, G. Rebovich, and K. Baldwin, "A Model of Systems Engineering in a System of Systems Context," Proceedings of 2008 Annual Conference on Systems Engineering Research, Redondo Beach, CA, 4-5 April 2008.

[7] Nielsen, C. B., Larsen, P. G., Fitzgerald, J., Woodcock, J., & Peleska, J. (2015). Systems of Systems Engineering. ACM Computing Surveys, 48(2), 1-41.

[8] ISO/IEC/IEEE 15288, Systems and software engineering - System life cycle processes

[9] Department of Defense (DoD), 2004, Defense Acquisition Guidebook Ch. 4.2.6. "System of Systems Engineering," Washington, DC: Pentagon, October 14.

[10] Seghiri, A., Belala, F., Benzadri, Z., & Hameurlain, N. (2018). A Maude based Specification for SoS Architecture. 2018 13th Annual Conference on System of Systems Engineering (SoSE).

[11] Cocks, D. (2006). 3.3.2 How Should We Use the Term "System of Systems" and Why Should We Care?. INCOSE International Symposium, 16(1), pp.427-438.

[12] Maier, M. (1998). Architecting principles for systems-of-systems. Systems Engineering, 1(4), pp.267-284.

[13] Cocks, D. (2006). 3.3.2 How Should We Use the Term "System of Systems" and Why Should We Care?. INCOSE International Symposium, 16(1), pp.427-438.

[14] Dahmann, J. and Baldwin, K. (2008). Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering. 2008 2nd Annual IEEE Systems Conference.

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

[16] Madni, A. and Sievers, M. (2013). System of Systems Integration: Key Considerations and Challenges. Systems Engineering, 17(3), pp.330-347.

[17] Bézévin, J. (2004). Sur les principes de base de l'ingénierie des modèles. L'objet, 10(4), pp.145-157.

[18] Selic, B. (2003). The pragmatics of model-driven development. IEEE Software, 20(5), pp.19-25.

[19] Benoît Combemale. Ingénierie Dirigée par les Modèles (IDM) – État de l'art. 2008. ffhal-00371565f.

[20] Omg.org. (2020). Model Driven Architecture (MDA) — Object Management Group. [online] Available at: https://www.omg.org/mda/ [Accessed 1 Mar. 2020].

[21] Bezivin, J. and Gerbe, O. (n.d.). Towards a precise definition of the OMG/MDA framework. Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001).

[22] Diaw, S., Lbath, R. and coulette, B. (2010). État de l'art sur le développement logiciel basé sur les transformations de modèles. Techniques et sciences informatiques, 29(4-5), pp.505-536.

[23] Kardas, G. (2013). Model-driven development of multiagent systems: a survey and evaluation. The Knowledge Engineering Review, 28(4), pp.479-503.

[24] IATA Forecast Predicts 8.2 Billion Air Travelers in 2037. (n.d.). IATA - Home. https://www.iata.org/en/pressroom/pr/2018-10-24-02/

[25] New York times. (2018, April 11). Military Plane Crashes in Algeria, Killing at Least 257.

[26] Liu, H., Tian, Y., Gao, Y., Bai, J., & Zheng, J. (2015). System of systems oriented flight vehicle conceptual design: Perspectives and progresses. Chinese Journal of Aeronautics, 28(3), 617-635.

[27] Harris, D., & Stanton, N. A. (2010). Aviation as a system of systems: Preface to the special issue of human factors in aviation. Ergonomics, 53(2), 145-148.

[28] IOX : Une Nouvelle Architecture Cisco Pour Le Fog Computing. (n.d.). Cisco Blogs. https://gblogs.cisco.com/fr/datacenter/iox-une-nouvelle-architecture-cisco-pour-le-fog-computing/