Architecture Description Method for Open Systems-of-Systems to Reduce Misunderstanding the Scopes of Managed Objects

Nobuyuki KOBAYASHI*, Hirotaka FUJINO*, Seiko SHIRASAKA*

*The System Design and Management Research Institute of Graduate School of System Design and Management, Keio University, 4-1-1 Hiyoshi, Kohoku-ku, Yokohama-shi, Kanagawa 223-8526 Japan
**Graduate School of System Design and Management, Keio University, 4-1-1 Hiyoshi, Kohoku-ku, Yokohama-shi, Kanagawa 223-8526 Japan

Abstract

The previous study argued that companies which do not consider or clarify managed objects as either a monolithic system or a system-of-systems (SoS). They are likely to confront problems in their analysis of cost and benefit relationships. In addition, managed objects are Systems-of-Systems that need to assume changing whether they are closed systems or open systems. Specifically, open Systems-of-Systems dynamically connect to each other in order to collectively provide a superordinate functionality, which could not be provided by a single system alone. In addition, the components and the structure of an open system-of-systems cannot be completely predicted at design stage, so that a typical top-down approach based on a concrete system would not work. Therefore, the aim of this study is to propose an architecture description method that clarifies the objects to reduce misunderstanding the scopes of managed objects at the designing open System-of-Systems. In other words, the proposed description method in this study is an open SoS design approach which is emergence-aware, in order to predict both components and the structure of an open SoS at design stage. The description methods dealt with in this study are life cycle, context diagram, use case specification (use case description), function flow block diagram, and allocation. The evaluation method is to confirm from the two viewpoints of whether the subjects can describe the managed objects as open SoS (Descriptability) and whether the subject can explain the managed objects as open SoS (Explainability). The results of this study suggested that the proposed architecture description method is effective for the subjects to recognize open SoS from the viewpoints of Descriptability and Explainability.

Keywords: Open System-of-Systems, Context Diagram, Use Case Specification, Function Flow Block Diagram (FFBD), Allocation, Architecture description method.

1. Introduction

It must distinguish between groups of systems that differ in the characteristics of good practices in development [1]. Maier [1] argued that companies which do not consider or clarify managed objects as either a monolithic system or a system-of-systems (SoS) are likely to face problems in their analyses of cost and benefit relationships. While vast literature on emergence in SoSs exists, and despite the fact that emergence is an intrinsic characteristic of SoSs, there are no SoS design approaches which are emergence-oriented or emergence-aware [2]. In addition, managed objects are Systems-of-Systems that need to assume changing by whether they are closed systems or open systems [3]. We decided to consider that managed objects in this study are Systems-of-Systems that assume to change (open Systems-of-Systems) [4]. The reason is that preventing open systems from becoming vague requires particular effort [5]. Additionally, open systems dependability is the ability to accommodate changes in purpose, objectives, environment and actual performance and to achieve accountability continually, so as to provide expected services as and when required [5]. Specifically, Trapp states that open Systems-of-Systems dynamically connect to each other in order to collectively provide a superordinate functionality, which could not be provided by a single system alone [4]. In addition, he states that the components and the structure of an open system-of-systems cannot be completely predicted at design stage, so that a typical top-down approach based on a concrete system would not work [4]. Therefore, the aim of this study is to propose an architecture description method that clarifies the objects to reduce misunderstanding the scopes of managed objects at the designing open System-of-Systems. We consider that clarifying the scopes of managed objects to reduce misunderstanding of them at the designing open SoS assists to reduce a part of scope that the components and the structure of an open SoS cannot be completely predicted at design stage. In other words, the proposed description method in this study is an open SoS design approach which is emergence-aware, in order to predict both components and the structure of an open SoS at design stage. The description methods dealt with in this study are life cycle [6], context diagram [7], use case specification (use case description) [8], function flow block diagram [9], and allocation [10]. The reason why those description methods are used in this study is that systems engineers and systems designers generally utilize those description methods for describing monolithic system architecture in systems engineering. On the other hands, in this study, we consider that we can explain an architecture description method for a system-of-systems. The architecture description method we would propose

*Corresponding author: n-kobayashi5@a6.keio.jp

Received: May 2, 2020
Accepted: September 10,2020
would be a guideline: a way of writing for describing architecture description easier. A system-of-systems consisting of two or more monolithic systems is a combination of evolutional systems which does not have an administrator for accomplishing the purpose which the independent monolithic systems only cannot accomplish. Therefore, the proposed architecture description method for open SoS is a guideline: a way of writing to describe which the components and the structure of the open SoS change in the operation stage, in the design stage of the open SoS.

The evaluation method is to confirm from the two viewpoints of whether the subjects can describe the managed objects as open SoS and whether the subject can explain the managed objects as open SoS. The reason is that subjects can explain open SoS by understanding the difference between the description of closed SoS and the description of open SoS. If subjects didn't understand open SoS, subjects can't explain description results. Therefore, if subjects understood open SoS, subjects will aware of the difference in explaining the other subjects' description results. The number of subjects in the evaluation is 10 persons.

Next, we describe the novelty of this study in detail. Kobayashi et al. have proposed an assurance case description method to reduce misunderstanding caused by the difference of grasping the objects managed in various departments as a monolithic system or a SoS [11]. This study differs from the previous study [11] in that it does not use assurance cases and that managed objects are open SoSs. Fischinger et al. introduce the conception and development of an open systems architecture framework that enables the application of complex systems [12]. Therefore, the previous study [12] introduces a viewpoint for thinking about open systems, and the previous study is not a study that focuses on an architecture description method to reduce misunderstanding the scopes of managed objects at the designing open SoS. The previous study [13] shows how use case diagrams, use case specifications, sequence diagrams, interface class diagrams and I/O entity class diagrams can be used to model the SoS, SoS constituent systems, and the associated capabilities at both the SoS and constituent system level. The previous study [13] is aimed at describing SoS, and the previous study is not a study that focuses on an architecture description method to reduce misunderstanding the scopes of managed objects at the designing open SoS. The previous study [14] introduces to use model transformation mechanisms to generate the corresponding abstract architecture, from which the system architect can deploy concrete constituent systems. The previous study [14] proposed model transformation mechanisms for creating abstract architecture. There are two differences between this study and the previous study. One difference is to show that systems designers and systems engineers discuss assuming that they change which objects in the abstract architecture. The other difference is to show that systems designers and systems engineers describe the abstract architecture on the assumption that external objects will change to reduce misunderstanding the scopes of managed objects. From the above, the novelty of this study is to propose an architecture description method that clarifies the objects to reduce misunderstanding the scopes of managed objects at the designing open System-of-Systems. In other words, this study contributes to an architecture description guideline for an open SoS, in order for the open SoS to work on the operation stage in the constraint that the components and the structure of the open SoS cannot be completely predicted at design stage.

This paper is organized as follows. We describe the features of both the SoS and the open systems and the conventional notation to section 2. We show the proposed description method in section 3. We show the evaluation method in section 4, the evaluation results in section 5, and discussion in section 6. Finally, we show conclusions and future research topics in section 7.

2. Previous Studies

Maier [1] describes that a system of systems is an assemblage of components which individually may be regarded as systems. Five principal features of a “system of systems” [15] [16] is as follows:

- **Operational Independence.** Any system that is part of an SoS is independent and is able to operate serviceably if the SoS is disassembled.
- **Managerial Independence.** Despite collaborating with the other members of the SoS, the individual systems are self-governing and individually managed so that they “not only can operate independently, they do operate independently.”
- **Geographic Distribution.** The parties collaborating in an SoS are distributed over a large geographic extent. Although the geographic extent is defined vaguely, it is stressed that the collaborating systems can only exchange information and not considerable quantities of mass or energy.
- **Evolutionary Development.** An SoS’s existence and development are evolutionary in the sense that objectives and functionality can be under constant change, as they can be added, modified, or removed with experience. Thus, an SoS never appears completely formed.
- **Emergent Behaviour.** Through the collaboration between the systems in an SoS a synergism is reached in which the system behaviour fulfils a purpose that cannot be achieved by, or attributed to, any of the individual systems.

Open systems have the following characteristics [3] [5].

- They are large, complex and interconnected.
- They can include black box components.
- Their purpose, objectives, environment and actual performance are not determined and change through their lives. Unpredictable changes of user requirements, service objectives, services received via network, black box components, technological basis, etc., are commonplace.
- Their boundaries, functions and structure are ever-
evolving and perceived differently by different stakeholders. Preventing them from becoming vague requires particular effort.

- Accountability is vital in their system life cycle and for risk control, but it needs particular effort to establish because of lack of effective central control.
- Understanding of the systems and their risks by their stakeholders is neither complete nor certain at any given time.
- The possibility of failure due to an incomplete understanding of the systems, unanticipated events and changes cannot be eliminated or predicted.
- Systems need to be resilient, need to have risk controls including error proofing, need to be able to recover from failures and need to be to adapt to prevent recurrence.
- Achievement of dependability requires an interactive approach and depends on integration of the system operation and development. Performing dependability activities throughout the system life cycle and iterating them as often as needed is particularly important for open systems.
- A system necessarily exchanges services with a wide variety of other interconnected, independently managed systems. These surrounding systems are managed according to their own principles and stakeholders, and their interfaces are subject to change for various reasons. The system must serve diverse stakeholders. Each stakeholder has different objectives and there might be no single authority over the system; moreover, the objectives of the system and the surrounding systems change with time. The conditions for the system, such as requirements and constraints, change frequently and unpredictably. Thus, there are uncertainty and incompleteness about these conditions and they cannot be understood completely at any given time.

ISO 15288 [6] defines that every system has a life cycle. A life cycle can be described using an abstract functional model that represents the conceptualization of a need for the system, its realization, utilization, evolution and disposal [6].

The basis of the Context Diagram was introduced by DeMarco [7] as “Data Flow Diagram” for one of the Structured Analysis tools. It shows the situation of the services from the point of view of the data. Use Case was introduced by Jacobson [8] as one of the methods for Object-Oriented Software Engineering. Use Case captures the functional requirements by describing the behavior of the system. We use FFBD (Function Flow Block Diagram [9]) in this paper. The function flow diagram is a method of subdividing by function flow and hierarchy. This method is used as a functional design for architectural design. The allocation of the previous study [10] was that each function and sub-function shall be allocated a set of performance and design requirements. These requirements shall be derived concurrently with development of functions, time-line analyses, synthesis of system design, and evaluation performed through trade-off studies and system/cost effectiveness analysis [10].

3. The proposed architecture description method that clarifies the objects

The common procedure for all description methods is as follows. We show the procedure for using the System of Interest (SoI) [17] and the external object (X) connected to the SoI to reduce misunderstanding of managed objects’ scopes in this study. Also, the SoS is the managed object whose architecture can be described by this study. That SoS is described as the SoI connected the other SoI in the context diagram. Further, we describe the state that the external object (X) connected to the SoI in the context diagram.

Procedure 1: Describe using each architecture description method (life cycle, context diagram, use case description, function flow block diagram, allocation) without setting the SoI. Systems designers and systems engineers describe the above including the external environment of the open systems-of-systems in this procedure.

Procedure 2: Set two objects to SoI1 and SoI2 by systems designers and systems engineers. (In the case of three or more, increase the SoI.) Therefore, systems designers and systems engineers rewrite the two objects to SoI1 and SoI2 in all description results of procedure 1.

Procedure 3: Consider the prerequisites for establishing open SoS, using the two objects set in SoI1 and SoI2. (If systems designers and systems engineers need to change use case description or/and function, they describe use case description or/and function to the description results of procedure 2.)

Procedure 4: Set the object connected to the SoI of procedure 2 as the external object (X), (systems designers and systems engineers describe X to the description results of procedure 3.)

Procedure 5: Set the prerequisites for establishing the external object (X) connected to the SoI. (If systems designers and systems engineers need to change use case description or/and function, they describe use case description or/and function to the description results of procedure 4.)

We provide the example using context diagram for corresponding to each procedure described in Table 1. Concretely, 1) of Table 1 expresses the situation including the external environment using context diagram. 2) of Table 1 shows that Measurement server changed to SoI1, and Measurement target changed to SoI2. Additionally, 3) of Table 1 shows that Monitoring software changed to X. We state the reasons for each procedure. Systems designers and systems engineers can confirm the relationship among considered systems by visualizing the current state in the procedure 1. Therefore, systems designers and systems engineers can confirm the architecture description result on which necessary objects set in order to reduce misunderstanding the scopes of managed objects.
Next, the previous study shows as follows. A distinguishing feature between Systems Engineering and System of Systems Engineering is the difficulty of defining the System of Interest (SOI) \[17\]. In order to solve the above, systems designers and systems engineers can change the objects to SoI (black box) in the description result with the state that set necessary objects. Therefore, systems designers and systems engineers can clearly understand which object changes to a managed object (SoI). In other words, as a result of deciding the object defined as SoI to a monolithic system, it is possible to prevent the systems designers and systems engineers from misunderstanding whether to regard the object as a Monolithic system or SoS \[11\]. In the previous study \[13\], by regarding SoS to the inside of SoI, as a result of expressing the structure of System-of-Systems as the inside of SoI (black box), it is difficult to determine which combination of SoI is SoS. The reason is that SoI is not a minimum unit by expressing the structure of System-of-Systems as the inside of SoI (black box), there are various combinations that use part of SoI. The systems designers and systems engineers need to describe diagrams’ objects being conscious of minimum units.

In procedure 3, systems designers and systems engineers can decide the boundary in which the managed objects allowed to change. If current objects can't realize the purpose by deciding the boundary in which the managed objects allowed to change, systems designers and systems engineers need to rewrite the objects of the description result other than the SoI on minimum units.

In procedure 4, systems designers and systems engineers can confirm the external objects (X) connected to the changing managed objects (SoI). In procedure 5, systems designers and systems engineers can decide the boundary in which the external objects (X) allowed to change. From the above, the proposal of this study is an architecture description method that clarifies the scope of changing managed objects. The way to clarify that scope is to decide the others' objects except of that managed objects in context diagram.

### 4. Evaluation Method

This study evaluates descriptability and explainability based on the evaluation procedure in Table 2. Therefore, we evaluate the descriptability by confirming whether subjects are able to describe architecture descriptions. Additionally, we evaluate the explainability by confirming whether subjects are able to explain among subjects using the description results.

Table 2 The relationship among evaluation procedure and evaluation items.

| Evaluation procedure | Evaluation items                   |
|----------------------|-----------------------------------|
| Explain the description method to the subject. | Descriptability |
| Subjects describes architecture descriptions. | Explainability |
| Subjects explain among subjects using the description results. | |

First, we communicated how to describe life cycle, context diagram, use case description, function flow block diagram, and allocation to 10 subjects who were not systems designers and systems engineers in the manufacturing industry. Second, we confirm whether the subjects can describe SoI and X. The reason of this confirmation is that if someone other than systems designers and systems engineers can describe the above, the results will indicate describing the above is not difficult. Additionally, this study does not specify the description themes determined by the subjects. That is because it is to confirm whether the subjects can describe various themes. Third, we confirm whether subjects can explain among subjects using the description results which set SoI and X. The reason is that we confirm to not only describe but also
explain the description results used SoI and X by the subjects. Table 3 shows an attribute of 10 subjects.

5. Evaluation Results

Table 4 shows the results, which described SoI1, SoI2, and X by each subject. Additionally, Table 4 includes the number of phases for the life cycle, the number of objects described in the context diagram, the number of descriptions of the use case specification, and the number of functions. We show the number of objects described by each phase for life cycle using the addition equation in Table 4. (For instance, in subject A, the number of life cycle is 4. Therefore, we described $3 + 3 + 3 + 3$ using the addition equation, in order to show the number of objects described in the context diagram for each phase of life cycle.) We recognize that number to utilize for allocation include both the number of objects described in the context diagram and the number of functions. Therefore, we only showed whether subjects can describe allocation. As a result of performing the evaluation, since only the subject G described the FFBD and the allocation directly without describing the use case specification, we do not show the number of descriptions for use case specification of subject G. We show in Figure 1 a description example of the context diagram in the state of procedure 4 of the proposed architecture description method. For example, in Figure 1, a subject assumes that Measurement server changes at the operation stage as an abstract SoI1 rather than a concrete object. As well as, in Figure 1, the subject assumes that Measurement target changes at the operation stage as an abstract SoI2 rather than a concrete object. Additionally, in Figure 1, the subject assumes that Monitoring software changes at the operation stage as an abstract X rather than a concrete object. Therefore, in Figure 1, the other subjects can understand to assume that other objects do not change at the operation stage.

Table 3 Cross-tabulation table of "Sex" and "Age".

| Sex / Age | male | female | Total |
|-----------|------|--------|-------|
| 20's      | 1    | 1      | 2     |
| 30's      | 3    | 3      | 6     |
| 40's      | 2    | 1      | 3     |
| 50's-     | 3    | 3      | 6     |
| Total     | 9    | 1      | 10    |

We show concrete examples rewriten in procedures 2 and 4, in order to replace objects which subjects described, to SoI1, SoI2 and X. In the description example of Use case specification, the subjects changed “A department determines the measurement conditions for Measurement server.” to “A department determines the measurement conditions for SoI1.”. In the description example of FFBD, the subjects similarly changed “The function of determining the measurement conditions for Measurement server.” to “The function of determining the measurement conditions for SoI1.”. At the time of describing allocation, the subjects rewrite objects to SoI1, SoI2, and X set in the context diagram. Identically, the subjects rewrite a function that needs to change toward a rewritten function as shown in the description example of FFBD.

Next, we show in Table 5 the description examples, which are the prerequisites for establishing open SoS, derived from procedure 3. We show in Table 6 the description examples, which are the prerequisites for establishing the external object (X), derived from procedure 5. In order to analyze the comments obtained from the subjects, we show divided into comments on the SoS as Table 7 and comments on the Open SoS as Table 8.

6. Discussion

First, we explain descriptability of the proposed method. Since 9 out of 10 subjects were able to describe SoI1, SoI2 and X, we consider that the subject can describe the managed object almost without mistake if the subjects understand the proposed method. Concretely, Figure 1 shows that which object is determined as SoI or X and which object is not SoI or X. Additionally, in Figure 1, the subject can describe how the X connects to the SoI by using the proposed method. Also, the subjects were possible to indicate by replacing the objects described to the use case specification, FFBD, and allocation with SoI or X. Table 5 suggests that the subjects can consider what
prerequisites need to be satisfied to establish open SoS by using the proposed method. Additionally, Table 6 suggests that the subjects were able to consider the prerequisites for which the external object X connects to the SoI which is an open system.

Next, we discuss explainability of the proposed method. Each subject discussed among the subjects the description results which set SoI and X and followed by proposed methods. As a result, we confirmed that each subject was able to explain while pointing with their finger [18]. Furthermore, we confirmed that subjects (audience) receiving the explanation were able to ask some questions of the subject (presenter), who explained the relation included SoI and X. The above result suggests through successfully asking questions that audience understood what SoI and X set within the description results.

Finally, we discuss comments on SoS and comments on open SoS as follows. No. 1 in Table 7 indicates that when designing SoS, it needs to recognize how SoI1 and SoI2 connect to each other. In other words, the comment suggests that the subjects were able to recognize the characteristics of SoS.

No. 2 in Table 7 indicates that the necessity to determine at the design stage what data both SoI1 and SoI2 can understand so that the subjects assume SoS to be going to evolve independently. Thus, the comment suggests that the subjects were able to recognize the characteristics of SoS, which the systems designers and systems engineers must consider at the design stage.

No. 3 in Table 7 indicates that in order for SoS to operate, if systems designers and systems engineers correctly design SoI1 and SoI2, each SoI operates within an assumed boundary, and each SoI cannot operate within an unassumed boundary.

Table 8 Comments on the open SoS.

| No. | Comments |
|-----|----------|
| 1   | When SoI connects to controllable external object (X), the SoI can show new function using X. If X is not connected to the SoI, it is difficult to determine the SoI which can show the existing functions of the SoI. |
| 2   | The designers need to determine what kind of connection between SoI and X is allowed. Therefore, the systems designers and systems engineers need to determine what kind of connection between SoI and X is allowed. Therefore, the |

Table 5 Examples of description in procedure 3.

| No. | Description results |
|-----|----------------------|
| 1   | SoI1 needs to be connected SoI2 via the Internet. |
| 2   | SoI1 and SoI2 need to know each other's IP addresses. |

Table 6 Examples of description in procedure 5.

| No. | Description results |
|-----|----------------------|
| 1   | X needs to know the command of SoI2 (the measurement target). |
| 2   | X needs to be connected via the Ethernet. |

Table 7 Comments on the SoS.

| No. | Comments |
|-----|----------|
| 1   | The designers need to determine the interface between SoI1 and SoI2. |
| 2   | SoI1 and SoI2 need to be able to exchange the data between SoI1 and SoI2. |
| 3   | When the designers design the SoS, we found the SoS does not always operate based on inferred combination of prerequisites. That is, designers would bear in mind that on SoS design, there are combinations of conditions in which the SoS does not operate. |
comment suggests that the subjects recognized that open SoS has a characteristic that connects to external objects.

No. 2 in Table 8 indicates that open SoS will connect some objects in the future. Additionally, the comment suggests that the systems designers and systems engineers recognize that there are functions that show even if it is not connected between external objects and open SoS. From the above comments, we consider that subjects were able to show that SoS was a combination of SoI1 and SoI2 by rewriting the two objects to two SoIs. Additionally, we consider that the subject recognized that the SoS works even if X that the SoI may connect to does not connect the SoI.

As results, the limitations of this study are as follows. We clarified that the proposed description method is valid for both descriptability and explainability of all of Context Diagram, use case specification, FFBD and Allocation, but future research topic needs to confirm whether it is valid for both descriptability and explainability of others.

The evaluation scope of this description method is that we confirmed by setting and rewriting existing objects as SoI and X. Therefore, future research topic needs to confirm whether systems designers and systems engineers can design open SoS by describing the managed objects (non-existing objects) to design in the future.

In order to assure designing open SoS, future research topic also needs an assurance case description method for explaining the results that systems designers and systems engineers were able to design managed objects which assumed open SoS.

The systems designers and systems engineers need to describe necessary objects being conscious of minimum units. This study does not provide a way for systems designers and systems engineers to recognize the minimum unit. Therefore, future research topic needs to provide a way for systems designers and systems engineers to recognize the minimum unit.

The evaluation of this study is conducted toward only 10 subjects who were not systems designers and systems engineers in the manufacturing industry. Therefore, future research topic needs to confirm whether systems designers and systems engineers can apply the proposed architecture description methods in other fields.

7. Conclusions

The aim of this study was to propose an architecture description method that clarifies the objects to reduce misunderstanding the scopes of managed objects at the designing open System-of-Systems. The results of this study suggested that the proposed architecture description method is effective for the subjects to recognize open SoS from the viewpoints of descriptability and explainability.

Future research topics are as follows:
- Future research topic needs to confirm whether it is valid for both descriptability and explainability of other architecture description methods.
- Future research topic needs to confirm whether systems designers and systems engineers can design open SoS by describing the managed objects (non-existing objects) to design in the future.
- Future research topic needs an assurance case description method for explaining the results that systems designers and systems engineers were able to design managed objects which assumed open SoS.
- Future research topic needs to indicate an effective description method depending on the type of System-of-Systems [19].
- Future research topic needs to provide a way for systems designers and systems engineers to recognize the minimum unit.
- Future research topic needs to confirm whether systems designers and systems engineers can apply the proposed architecture description methods in other fields.

References

[1] Mark W. Maier: Architecting Principles for Systems of Systems, Systems Engineering, Volume 1, Issue 4, 1998, 267-284
[2] Andrea Ceccarelli, Marco Mori, Paolo Lollini, Andrea Bondavalli: Introducing Meta-Requirements for Describing System of Systems, IEEE 16th International Symposium on High Assurance Systems Engineering, 2015, 150-157.
[3] Mario, Tokoro: Open Systems Dependability: Dependability Engineering for Ever-Changing Systems, Second Edition, CRC Press, 2015
[4] M. Trapp, "Assuring functional safety in open systems of systems", Habilitationsschrift Kaiserslautern Technische Universit" Kaiserslautern, 2016.
[5] IEC 62853: open system dependability, 2018
[6] ISO 15288:2015: Systems and software engineering -- System life cycle processes, 2015.
[7] DeMarco, Tom: Structured Analysis and System Specification, 1979.
[8] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, Gunnar Overgaard: Object-Oriented Software Engineering: A Use Case Driven Approach. Wokingham, England, Addison-Wesley., 1992
[9] MIL-STD-499: Functional Flow Diagrams, MIL-STD-499, USAF, 1968.
[10] MIL-STD-499: MILITARY STANDARD Engineering Management, MIL-STD-499, USAF, 1968.
[11] N. Kobayashi, A. Nakamoto, M. Kawase, F. Susan, M. Ioki, S. Shirasaka, “Managing a monolithic system or a System-of-Systems? An assurance case approach to reach intra-organizational consensus”, proceedings 2018 7th International Congress on Advanced Applied Informatics (IIAI-AAI 2018), 2018, pp. 688-693.
[12] Michael Fischinger, Christian Neureiter, Christoph Binder, Norbert Egger and Michael Renoth: FREDOSAR: Towards a Security-aware Open System Architecture Framework Supporting Model based Systems Engineering, Proceedings of the 8th International Conference on Smart Cities and Green
13. Lane, J. and T. Bohn.: Using SySML to Evolve Systems of Systems. Los Angeles, CA, USA: USC CSSE Technical Report, USC-CSSE-2010-506, 2010.

14. Imane Cherfa, Nicolas Belloir, Salah Sadou, Régis Fleurquin, Djamal Bennour: Systems of systems: From mission definition to architecture description, Systems Engineering, Vol 22, Issue 6, (2019), 437-454

15. Mark W. Maier: Architecting principles for systems-of-systems. In INCOSE 1996 6th Annual International Symposium of the International Council on Systems Engineering. INCOSE, 1996.

16. C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, J. Peleska: Systems of systems engineering: basic concepts, model-based techniques, and research directions. ACM Computing Surveys, Volume 48, Issue 2, 2015, 1–41.

17. A. Kinder, V. Barot, M. Henshaw, C. Siemieniuch: System of Systems: “Defining the system of interest.”, In Proceeding 7th International Conference on System of Systems Engineering, (2012), 463–468.

18. N. Kobayashi, M. Kawase, S. Shirasaka: A Proposal of Assurance Case Description Method for Sharing a Company’s Vision, Journal of Japan Association for Management Systems, Vol.34, No.1, (2017), 85-94.

19. Department of Defence: Systems Engineering Guide for Systems of Systems. Technical Report, Department of Defence. (2008), Available at http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf.