11-1-2018

System of Systems Architecting Problems: Definitions, Formulations, and Analysis

Hadi Farhangi

Dincer Konur
Missouri University of Science and Technology, konurd@mst.edu

Recommended Citation

H. Farhangi and D. Konur, "System of Systems Architecting Problems: Definitions, Formulations, and Analysis," *Procedia Computer Science*, vol. 140, pp. 29-36, Elsevier B.V., Nov 2018.
The definitive version is available at https://doi.org/10.1016/j.procs.2018.10.289

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 140 (2018) 29–36

Procedia Computer Science

www.elsevier.com/locate/procedia

Complex Adaptive Systems Conference with Theme: Cyber Physical Systems and Deep Learning, CAS 2018, 5 November – 7 November 2018, Chicago, Illinois, USA

System of Systems Architecting Problems: Definitions, Formulations, and Analysis

Hadi Farhangi[a]*, Dincer Konur[b]

[a]*Savannah State University, 3219 College St, Savannah, GA 31404, USA*
[b]*McCoy College of Business Administration, Texas State University, San Marcos, TX 78666, USA*

Abstract

The system of systems architecting has many applications in transportation, healthcare, and defense systems design. This study first presents a short review of system of systems definitions. We then focus on capability-based system of systems architecting. In particular, capability-based system of systems architecting problems with various settings, including system flexibility, fund allocation, operational restrictions, and system structures, are presented as Multi-Objective Nonlinear Integer Programming problems. Relevant solution methods to analyze these problems are also discussed.

Keywords: System of Systems; Architecting; SoS Structure; Multi-objective Optimization

1. Introduction and Literature Review

One of the earliest studies in System of Systems (SoS) is the work of Jackson and Keys [1] that discusses SoS as a methodology to tackle complex problems. Currently, SoS concepts are widely accepted as a tool to understand complex systems and help decision makers prepare organizations in response to new challenges and changes in the environment [2,3]. According to [4], service operations, space exploration, electric power systems grids, renewable energy, environmental management, healthcare, national security, and infrastructure are among the areas that can

* Corresponding author. Tel.: +1-912-689-0107.
E-mail address: farhangih@savannahstate.edu

10.1016/j.procs.2018.10.289

benefit from SoS concepts, in addition to transportation [5] and military applications [6,7,8]. All aspects of decision making in the areas noted above, face an increasing complexity and subject to fast changes that the traditional management techniques cannot handle [9] and SoS has emerged as a method to handle these issues.

This paper explores some definitions and properties of SoS and architecting of a SoS. Furthermore, a generic mathematical formulation is presented for a domain-independent SoS architecting problem. We acknowledge that the literature of SoS architecting is much larger than what is covered in this text. This text extends the formulation of SoS architecting problems to cover variety of interfaces one may use to create a SoS.

Kotov [10] define SoS as a system that is comprised of complex systems, while in [11], SoS is defined as a set of systems to perform a set of functions with at least one function that a single system cannot perform, individually. A similar approach to define SoS is followed in other works by using the concept of capabilities of systems instead of a functional concept [7,8,12]. In other definitions of SoS, one can see the presence of properties and characteristics of SoS within the definition. For example, both [13,14] define a SoS considering five characteristics, which are the operational independence of systems, managerial independence, geographical distribution, emergent behavior, and evolutionary development. Based on these definitions and characteristics, and being inspired by Bertalanffy's General System Theory [15], SoS and its properties are defined separately as follows:

Definition 1. SoS is a set of interacting systems.

Definition 2. SoS properties are:

i. SoS is capable or functional [7,8,11,12],
ii. SoS is connected [12,17,18],
iii. SoS exhibits emergent behavior [13,14],
iv. SoS develops and evolves over time [13,14],
v. SoS temporarily assembles systems [16].

Property (i) announces the most important property of a SoS. A SoS should be designed to have a specific set of capabilities. This property is utilized in this work to formulate a generic SoS problem. Property (ii) is related to interface design, i.e., connectivity of systems in the SoS is managed through interfaces. A variety of interfaces types are discussed in [18] and a SoS can have different layers of interfaces with different structures. Designing different interface structures is important as it dictates how systems are connected and communicate with each other; and, the main contribution in this study is analyzing different design approaches to connecting the systems. Property (iii) highlights a very important property of a SoS; emerging behavior, which means SoS exhibits some characteristics and behavior that it is not designed for; see e.g. [19] for more discussion on the emergence property of systems. Although very important, this property is not being addressed in this paper. Property (iv) dictates the development of SoS over time: a SoS develops and evolves through time. Property (v) states the temporarily nature of the SoS, which means systems will come together and connect temporarily, otherwise systems will dissemble the components and subsystems that do not contribute to the SoS capabilities to reduces their costs. As a result, systems will reduce to subsystems and SoS will reduce to a complex system.

In the literature of SoS, a variety of methods exists for creating and developing a SoS, referred to as SoS architecting. For example, Caffall and Michael [7] provide a design of a network of systems that are connected through interfaces and controlled by an architect. The work of [18] similarly presents a network design for SoS architecting. Another example is [12], where an abstract model is described for the SoS architecting. Among others, Khalil et al. [20] describe a hypergraph for architecting a SoS, Ricci et al. [21] discuss the application of the Theory of Ilities in SoS architecting, an application of graph theory in SoS architecting can be found in [18,22], an application of Multilevel Bong Graph can be seen in [23], and mathematical programming approaches can be seen in [24,25,26,27]. In this work, similar to [24,25,26,27], we follow an analytical method by using Mathematical Programming tools. More precisely, we architect a SoS based on the Definition 1 and Properties (i) and (ii). In other words, we select a subset of available systems for the SoS to provide a set of capabilities, and we connect them through interfaces. The assembly of systems and the connections created should be temporary. It is important to note that our analytical model does not quantify emergent and evolutionary properties of the SoS at this point. We post such analyses as future research areas.

Flexibility in systems and SoS is crucial as a system or SoS needs to respond to the environmental changes [29,30]. Flexibility can be defined as a system or a SoS ability to respond to internal or external changes [29,30] within acceptable time and cost ranges [28,29]. By this definition, one can incorporate the responding time and cost in the

corresponding model by adding a higher responding time and cost to inflexible systems and a lower responding time and cost to flexible systems. Konur et al. [24] define flexibility as a systems' response to provide capabilities, i.e. if a system cannot drop a capability, it is inflexible. In the problem formulation section, we use this definition. Although this definition is extended by considering different dissembling costs for dropping a capability in [25], we only consider the former case.

In Section 2, a capable and connected SoS is presented based on set covering problem, SoS structure constraints, and the availability of performance improvement funds. Section 3 discusses different problem formulations, solution methods, some drawback of solution methods, and future research directions in mathematically formulating the SoS architecting problem. Section 4 concludes the text with a summary.

2. Problem Formulations

In this section we formulate the SoS architecting problem. We use a similar notation for the indices, sets, and variables as of [24,25,26] and later extend the models by considering various structures for connecting the systems. The set of systems is $J = \{1, \dots, m\}$, an example is shown in Figure 1. (a), and it is indexed by j. The set of capabilities is $I = \{1, \dots, n\}$ and it is indexed by i and $r \in I$ is a special capability of interest. The set of inflexible systems is J_1 and the set of flexible systems is J_2 such that $J_1 \cap J_2 = \emptyset$ and $J_1 \cup J_2 = J$. The set $R \subseteq J_2$ is a set that contains systems with r capability and N is a non-empty subset of J. Let A be the capability matrix of a_{ij}'s such that $a_{ij} = 1$ if system j can provide capability i and it is 0, otherwise. In addition, let p_{ij} be the performance of system j in providing capability i, c_{ij} be the cost of system j in providing capability i, t_{ij} be the deadline for system j to provide capability i. Moreover, three sets of variables are defined as follows: z_j is equal to 1 if system $j \in J$ is selected for the architecture and it is 0, otherwise. Variable x_{ij} is equal to 1 if system $j \in J_2$ is asked to provide capability i, otherwise it is 0. A set of selected systems is shown in Figure 1. (b). Variable u_k is equal to 1 if system $k \in R$ provides the capability r, otherwise it is 0. Based on this initial definition of variables, following constraints, which are adapted from [24,25], should hold:

$$z_j \leq \sum_{i \in I} x_{ij} \qquad \forall j \in J_2 \qquad (1)$$

$$n\, z_j \geq \sum_{i \in I} x_{ij} \qquad \forall j \in J_2 \qquad (2)$$

$$x_{ij} \leq a_{ij} \qquad \forall j \in J_2, \forall i \in I \qquad (3)$$

Constraints (1) and (2) guarantee that if a flexible system is asked to provide a capability, it should be included in the architecture. Constraints (3) guarantee that we do not ask a flexible system to provide a capability that it cannot provide. The resulting SoS should be capable based on property (i), in which, every capability should be provided by at least one system. This property can be captured by the following constraints which cover all the required capabilities and are set covering constraints (adapted from [25]):

$$\sum_{j \in J_1} a_{ij} z_j + \sum_{j \in J_2} a_{ij} x_{ij} \geq 1 \qquad \forall i \in I \qquad (4)$$

The resulting SoS should also be connected based on property (ii). The connectedness of SoS is achieved through the interface design for the selected systems. One may define several types of interfaces, each of which exhibits a different type of structures to connect systems. Borrowing network topology concepts of [31,32,33], we define the Complete, Star, Tree, and Ring structures as shown in Figure 1. (c)-(f), noting that an architect may use different types of structures for connecting systems.

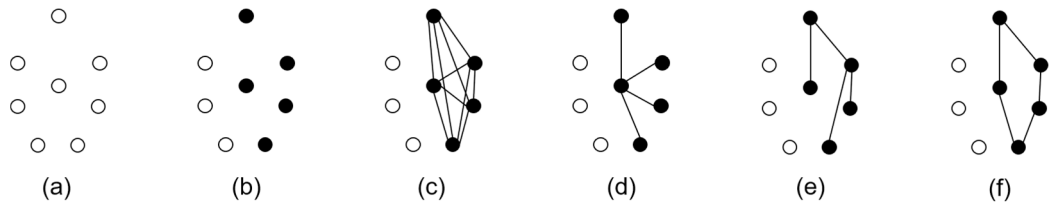

Figure 1. (a) a set of systems in circles; (b) a set of selected systems in filled circles; (c) Complete structure of selected systems; (d) Star structure; (e) Ring structure; (f) Tree structure

Complete Structure. This structure is adapted from [24,25,26,27]. This connection, i.e. the interface as depicted in Figure 1. (c) as edges that connect all selected systems, is defined by variable y_{jk}^1 which is equal 1 if two systems j, k are selected ($z_j = z_k = 1$) and it is 0, otherwise. The constraints for the complete structure are as follows:

$$y_{jk}^1 \geq z_j + z_k - 1 \qquad \forall j \in J, \forall k \in J, k > j \qquad (5)$$

$$y_{jk}^1 \leq z_j \qquad \forall j \in J, \forall k \in J, k > j \qquad (6)$$

$$y_{jk}^1 \leq z_k \qquad \forall j \in J, \forall k \in J, k > j \qquad (7)$$

Constraints (5) enforce $y_{jk}^1 = 1$ if both systems j and k are selected ($z_j + z_k = 2$) and constraints (6) and (7) enforce $y_{jk}^1 = 0$ if at least one of the systems j, k is not selected ($z_j + z_k \leq 1$) [24].

Star Structure. There may exist some capabilities, such as command-and-control, that only one system should provide. This structure is depicted in Figure 1. (d) as edges that create a Star structure from the selected systems. The set of systems with these special capabilities should be flexible because, if included in the architecture only one of them should provide this special capability and the rest should not provide it. We use variable y_{jk}^2 that is defined similar to y_{jk}^1 in the complete structure. The constraints for the complete structure read as follow.

$$u_k \leq z_k \qquad \forall k \in R \qquad (8)$$

$$\sum_{k \in R} u_k = 1 \qquad (9)$$

$$y_{jk}^2 \geq z_j + u_k - 1 \qquad \forall k \in R, \forall j \in J \backslash \{k\} \qquad (10)$$

$$y_{jk}^2 \leq z_j \qquad \forall k \in R, \forall j \in J \backslash \{k\} \qquad (11)$$

$$y_{jk}^2 \leq u_k \qquad \forall k \in R, \forall j \in J \backslash \{k\} \qquad (12)$$

$$x_{rk} \leq u_k \qquad \forall k \in R \qquad (13)$$

Constraints (8) and (9) use new variable u_k to enforce that only one system with the special capability is selected. Constraints (10), (11), (12) connect all the other systems to this system and the resulting structure has a star shape. Constraints (13) guarantee that the other systems that can provide the special capability are not asked to provide it. It is important to note that the star structure is very suitable for communication interfaces among systems, where the special capability becomes the command-and-control capability.

Tree Structure. For the tree structure, as depicted in Figure 1. (e) as edges that create a Ring structure from the selected systems, the variable y_{jk}^3 is defined similar to y_{jk}^1. The constraints for the undirected Tree structure, adapted from [31,32,33], reads as follows:

$$\sum_{j \in J} \sum_{k \in J, k > j} y_{jk}^3 = \sum_{j \in J} z_j - 1 \qquad (14)$$

$$\sum_{j \in N} \sum_{k \in N, k > j} y_{jk}^3 \leq \sum_{j \in N \backslash \{s\}} z_j \qquad \forall s \in N, N \subseteq J, 2 \leq |N| \leq |J| - 2 \qquad (15)$$

Based on the definition of a tree [31], the total number of edges must be equal to the number of nodes minus one. Constraints (14) impose this definition. Furthermore, every subset of nodes N ($N \subseteq J, 2 \leq |N| \leq |J| - 2$) must be connected by at most $|N| - 1$ edges ($\sum_{j \in N \backslash \{s\}} z_j \leq |N| - 1$), which is addressed in constraints (15).

Ring Structure. In the ring structure, as depicted in Figure 1. (f), every selected system is connected to exactly two other systems such that there exists only one cycle that passes through all selected systems. By this characteristic, it is evident that the ring is an undirected Hamiltonian cycle over the selected systems. The constraints that can capture this characteristic of the ring are as follows which are adapted from [31], where the variable y_{jk}^4 is defined similar to y_{jk}^1.

$$\sum_{k\in J,k>j} y^4_{jk} + \sum_{k\in J,k<j} y^4_{jk} = 2z_j \qquad \forall j \in J \tag{16}$$

$$\frac{1}{2}\left(\sum_{k\in J,k>j} y^4_{jk} + \sum_{k\in J,k<j} y^4_{jk}\right) + \left(1 - z_p\right) + (1 - z_q) \geq 1 \qquad \begin{matrix}\forall p \in N, \forall q \in J\backslash N,\\ N \subseteq J, 3 \leq |N| \leq |J| - 3\end{matrix} \tag{17}$$

If a system is not selected, constraints (16) impose that all structure variables coming to that systems and going out are zeros. If a system is selected, these constraints enforce two interfaces to connect the selected system to two other selected systems. Constraints (17) are sub-tour elimination constraints for the resulting Hamiltonian cycle. Note that constraints for a Hamiltonian cycle is the same as Traveling Salesman constraints, since the Traveling Salesman problem finds the optimum Hamiltonian cycle, given an objective function.

The SoS architect may want to improve the systems' performances by allocating funds. We formulated the constraints related to the fund allocation similar to the [26,33]. Particularly, we defined variables v_{ij} as the amount of fund that we can allocate to system j to improve its performance in providing capability i at γ_{ij} rate. Furthermore, we assume that system j cannot improve its capability i performance more that $\overline{p_{ij}}$. Let $\hat{v}$ be the total available fund and we assume there is upper bound on the amount of funds the we can allocate to system j to improve its performance in capability i that we show by $\overline{v_{ij}}$. Then, we have the following constraints:

$$\sum_{i\in I}\sum_{j\in J} v_{ij} \leq \hat{v} \tag{18}$$

$$\sum_{i\in I} v_{ij} \leq \hat{v}\, z_j \qquad \forall j \in J \tag{19}$$

$$P\left(z_j, v_{ij}\right) \leq \overline{p_{ij}} \qquad \forall i \in I, \forall j \in J_1 \tag{20}$$

$$P\left(x_{ij}, v_{ij}\right) \leq \overline{p_{ij}} \qquad \forall i \in I, \forall j \in J_2 \tag{21}$$

$$0 \leq v_{ij} \leq a_{ij}\overline{v_{ij}} \qquad \forall i \in I, \forall j \in J \tag{22}$$

Constraints (18) guarantee that the total allocated fund is less than the available fund f. Constraints (19) guarantee that we only allocate fund to the selected systems. Constraints (20) provide an upper bound to the achieved performances after allocating fund. The performance of the inflexible system can be improved by any function of z_j, i.e. $P\left(z_j, v_{ij}\right)$ can have different forms. One can use a linear form as $P\left(z_j, v_{ij}\right) = a_{ij}p_{ij}z_j + \gamma_{ij}v_{ij}$. The same is true for constraints (21) as the linear version will be $P\left(x_{ij}, v_{ij}\right) = a_{ij}p_{ij}x_{ij} + \gamma_{ij}v_{ij}$ for flexible systems. Finally, constraints (22) force a lower bound and an upper bound to v_{ij} variables, i.e. these variables cannot be negative and we only assign funds to a system to provide a capability that it originally can provide with the maximum amount of $\overline{v_{ij}}$.

As discussed in [25], the architect may have different objectives (cost, performance, and deadline are some examples) to construct a SoS. In what follows, some forms for these objective functions will be presented. Similar to [25], let $\boldsymbol{AP}$, $\boldsymbol{AC}$, and $\boldsymbol{AD}$ denote the aggregate performance, aggregate cost, and the aggregate deadline of the SoS, respectively.

To capture the aggregate performance of the SoS, one may use different objective functions. In [25], a linear form is used that we adapt as follows, where the first two parts of $\boldsymbol{AP}$ are the summation of systems performances in providing the required capabilities and the last part is the improvement in systems performances after allocating funds.

$$\boldsymbol{AP} = \sum_{i\in I}\sum_{j\in J_1} a_{ij}p_{ij}z_j + \sum_{i\in I}\sum_{j\in J_2} a_{ij}p_{ij}x_{ij} + \sum_{i\in I}\sum_{j\in J}\gamma_{ij}v_{ij} \tag{23}$$

To formulate the aggregate cost, one can address the flexibility of systems by assigning penalty cost of not including a capability from a flexible system [25]. Here, we neglect that and formulate the cost as a linear function of the cost to provide a capability by a system plus the cost of interfaces between systems and total allocated funds. This function is an extension of $\boldsymbol{TC}$ function in [26].

$$\begin{aligned}\boldsymbol{AC} = {} & \sum_{i\in I}\sum_{j\in J_1} a_{ij}c_{ij}z_j + \sum_{i\in I}\sum_{j\in J_2} a_{ij}c_{ij}x_{ij} + \sum_{j\in J}\sum_{k\in J,k>j} h^1_{jk}y^1_{jk} + \sum_{k\in H}\sum_{j\in J\backslash\{k\}} h^2_{jk}y^2_{jk} + \\ & \sum_{j\in J}\sum_{k\in J,k>j} h^3_{jk}y^3_{jk} + \sum_{k\in J,K>j} h^4_{jk}y^4_{jk} + \sum_{k\in J,K<j} h^4_{jk}y^4_{jk} + \sum_{i\in I}\sum_{j\in J} v_{ij}\end{aligned} \tag{24}$$

The first two terms in $\boldsymbol{AC}$ are the cost of providing capabilities by inflexible and flexible systems, respectively. The third part is the cost of connecting systems via a complete structure, where h_{jk}^1 is the cost of connecting system j to system k in complete structure. The fourth part is the cost of star structure. The fifth part is the cost of tree structure; and the sixth and seventh parts are the cost of ring structure. The corresponding cost coefficients are shown by h_{jk}^2, h_{jk}^3, and h_{jk}^4, respectively, for the Star, Tree, and Ring structures. The last part is the total allocated funds.

To formulate the aggregate deadline, one may use different functions. For example, in [25], it is assumed that systems work in parallel to provide capabilities and every individual system works on each capability in parallel, as well. As the result, [25] formulated the aggregate deadline as follows:

$$\boldsymbol{AD} = \max_{i \in I} \left\{ \max_{j \in J_1} \{a_{ij} t_{ij} z_j\}, \max_{j \in J_2} \{a_{ij} t_{ij} x_{ij}\} \right\} \tag{25}$$

Based on how systems work on their capabilities, this function can take different forms. For example, if we assume that systems work in parallel in the SoS, but they work on providing their capabilities in series, then the function takes the following form.

$$\boldsymbol{AD} = \max_{j \in J} \{ \textstyle\sum_{i \in I} a_{ij} t_{ij} z_j , \sum_{i \in I} a_{ij} t_{ij} x_{ij} \} \tag{26}$$

In the next section, different variation of the SoS architecting problems will be discussed.

3. Analysis and Future Research

The generic SoS architecting problem (SoS-AP) can be stated as follows:

SoS-AP $\max \boldsymbol{AP}, \min \boldsymbol{AC}, \min \boldsymbol{AD}$

s.t. Constraints (1) - (22)

Binary Definition of $z_j, x_{ij}, u_k, y_{jk}^1, y_{jk}^2, y_{jk}^3, y_{jk}^4$ and $v_{ij} \geq 0$

In what follows, we use SoS-AP to address a generic SoS architecting problem disregarding the SoS structure type. Note that, depending on SoS structure, the architect may decide to remove some variables and constraints or add more. For example, if architect only considers a complete structure, then only constraints (1) - (7) and variables z_j, x_{ij}, y_{jk}^1 should be considered for SoS-AP [24,25,34]. Similarly, having all types of interfaces will force the architect to use all variables and constraints (1) – (22). Moreover, if the SoS needs two similar interfaces, e.g. two Start structures, then we need to define additional variables such as u_k^1, y_{jk}^5 and duplicate constraints (8) – (13) using these new variables. Finally, if there is no fund allocation, variables f_{ij} and constraints (18) – (22) should be removed from the SoS-AP. Nevertheless, SoS-AP is a Multi-Objective Mixed-Integer Non-Linear Programming problem. In what follows, a short analysis of this problem, appropriate solution methods, and future research directions are discussed.

As noted in the first section, constraints (1) - (22) guarantee properties (i) and (ii). Indirectly, property (v) is also satisfied as long as solutions to the SoS-AP is being carried out for a limited period of time. Further research is needed to guarantee the temporary implementation of the SoS structure. In addition, the evolving nature of the SoS, as stated in property (iv), is not addressed in SoS-AP and it is a subject of the future research.

Note that problem SoS-AP is *NP-Hard* due to constraints (4) that come from Set Covering problem and constraints, constraints (14) and (15) that come from minimum spanning tree problem, and constraints (16) and (17) that come from Traveling Salesman problem. There exist various solution methods for solving such a problem. One may either search for the exact set of Pareto efficient solutions, or find a subset of efficient solutions, or approximate efficient solutions. One may read [35] for a review on the appropriate solution methods. If we choose to use linear functions for $\boldsymbol{AP}$ and $\boldsymbol{AD}$, or linearize them, then the problem will become a Multi-Objective Mixed-Integer Linear Programming problem. For the literature of solution methods for solving such problems, one may study [36].

Problem SoS-AP faces several issues to be implemented effectively, which includes different types of interfaces, funding allocation, number of objectives, and number of solutions. First, Problem SoS-AP may have fewer constraints, as formulated and solved in [24,25,26,27], or it may have more operational constraints that are application dependent.

Heuristic methods for solving such problems should be adjusted according to the new constraints. If there is no funding, the case that we only have binary variables, the problem will reduce to a Multi-objective Combinatorial Optimization problem. For a survey of appropriate solution methods, one may read [37,38].

Second, an architect may decide to use more or fewer objective functions. The resulting model may remain a Multi-Objective Mixed-Integer Linear Programming problem or it may reduce to a Mixed-Integer Linear Programming problem. The case of two objectives is recently solved in [26] via an adaptive epsilon-constraint method and a two-stage evolutionary algorithm.

Finally, when number of objectives increases, generating efficient solutions becomes difficult. More precisely, generating efficient solutions may not be tractable even for approximation methods. Hence, it is critical to reduce the number of objective functions and the number of efficient solutions. To reduce the number of objective functions, one can use ranking methods similar to [39]. To reduce the number of efficient solutions, it is necessary to develop post-processing tools based on the decision maker's preferences. Furthermore, Pareto-efficient solutions cannot capture the dynamic and stochastic environment [9]. This means that we need to search for solutions that can address the rapidly changing environment and the evolving property of the SoS. To tackle this issue, it is important to address the Property (iv) of the SoS, i.e. the evolutionary development of the SoS. Mathematically formulating evolving SoS architectures is another future direction for this research.

4. Conclusion

In this paper, a short review of the SoS definitions and properties is conducted. Using the definitions of SoS and its properties, SoS architecting problems with different interface layers are presented. It is discussed the SoS architecting problem in this sense is a Multi-objective Mixed-Integer Non-Linear Programming problem that can be changed to a linear problem and reduced to combinatorial problems. A short review on the existing solution methods for the linear case of the problem is presented. It is discussed that formulating the evolution of the SoS in the form of mathematical programming is one of the future direction of SoS architecting problems.

References

[1] Jackson, M. C., & Keys, P. (1984). Towards a system of systems methodologies. Journal of the operational research society, 35(6), 473-486.

[2] Agarwal, S., Pape, L. E., Kilicay-Ergin, N., & Dagli, C. H. (2014). Multi-agent based architecture for acknowledged system of systems. Procedia Computer Science, 28, 1-10.

[3] Allison, M., Batdorf, R., Chen, H., Generazio, H., Singh, H., & Tucker, S. (2004). The characteristics and emerging behaviors of system of systems. New England Complex Systems Institute, Boston, MA.

[4] Sage, A. P. (2011). System of systems engineering: innovations for the 21st century (Vol. 58). John Wiley & Sons.

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

[6] Ender, T., Leurck, R. F., Weaver, B., Miceli, P., Blair, W. D., West, P., & Mavris, D. (2010). Systems-of-systems analysis of ballistic missile defense architecture effectiveness through surrogate modeling and simulation. IEEE Systems Journal, 4(2), 156-166.

[7] Caffall, D. S., & Michael, J. B. (2005, October). Architectural framework for a system-of-systems. In Systems, Man and Cybernetics, 2005 IEEE International Conference on (Vol. 2, pp. 1876-1881). IEEE.

[8] Dahmann, J. S., & Baldwin, K. J. (2008, April). Understanding the current state of US defense systems of systems and the implications for systems engineering. In Systems Conference, 2008 2nd Annual IEEE (pp. 1-7). IEEE.

[9] Haimes, Y. Y., & Anderegg, A. (2015). Sequential Pareto□Optimal Decisions Made During Emergent Complex Systems of Systems: An Application to the FAA NextGen. Systems Engineering, 18(1), 28-44.

[10] Kotov, V. (1997). Systems of systems as communicating structures (Vol. 119). Hewlett Packard Laboratories.

[11] DeLaurentis, D., & Callaway, R. K. (2004). A system□of□systems perspective for public policy decisions. Review of Policy research, 21(6), 829-837.

[12] Baldwin, W. C., & Sauser, B. (2009, May). Modeling the characteristics of system of systems. In System of Systems Engineering, 2009. SoSE 2009. IEEE International Conference on (pp. 1-6). IEEE.

[13] Sage, A. P., & Cuppan, C. D. (2001). On the systems engineering and management of systems of systems and federations of systems. Information knowledge systems management, 2(4), 325-345.

[14] Maier, M. W. (2009). The art of systems architecting. CRC press.

[15] Von Bertalanffy, L. (1968). General system theory. New York, 41973(1968), 40.

[16] Ceccarelli, A., Bondavalli, A., Froemel, B., Hoeftberger, O., & Kopetz, H. (2016). Basic Concepts on Systems of Systems. In Cyber-Physical Systems of Systems (pp. 1-39). Springer, Cham.

[17] Boardman, J., & Sauser, B. (2006, April). System of Systems-the meaning of of. In System of Systems Engineering, 2006 IEEE/SMC International Conference on (pp. 6-pp). IEEE.

[18] Gorod, A., Sauser, B., & Boardman, J. (2008). System-of-systems engineering management: A review of modern history and a path forward. IEEE Systems Journal, 2(4), 484-499.

[19] Chalmers, D. J. (2006). Strong and weak emergence. The reemergence of emergence, 244-256.

[20] Khalil, W., Merzouki, R., Ould-Bouamama, B., & Haffaf, H. (2012). Hypergraph models for system of systems supervision design. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 42(4), 1005-1012.

[21] Ricci, N., Fitzgerald, M. E., Ross, A. M., & Rhodes, D. H. (2014). Architecting systems of systems with ilities: An overview of the SAI method. Procedia Computer Science, 28, 322-331.

[22] Harrison, W. K. (2016). The Role of Graph Theory in System of Systems Engineering. IEEE Access, 4, 1716-1742.

[23] Kumar, P., Merzouki, R., & Bouamama, B. O. (2017). Multilevel Modeling of System of Systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems.

[24] Konur, D., Farhangi, H., & Dagli, C. H. (2014). On the flexibility of systems in system of systems architecting. Procedia Computer Science, 36, 65-71.

[25] Konur, D., Farhangi, H., & Dagli, C. H. (2016). A multi-objective military system of systems architecting problem with inflexible and flexible systems: formulation and solution methods. OR spectrum, 38(4), 967-1006.

[26] Farhangi, H., Konur, D., & Dagli, C. H. (2016). Multiobjective System of Systems Architecting with Performance Improvement Funds. Procedia Computer Science, 95, 119-125.

[27] Farhangi, H., Konur, D., & Dagli, C. H. (2016). Combining Max-min and Max-max Approaches for Robust SoS Architecting. Procedia Computer Science, 95, 103-110.

[28] Gorod, A., Gandhi, S. J., Sauser, B., & Boardman, J. (2008). Flexibility of system of systems. Global Journal of Flexible Systems Management, 9(4), 21-31.

[29] Nilchiani, R., & Hastings, D. E. (2007). Measuring the Value of flexibility in space systems: A six□element framework. Systems Engineering, 10(1), 26-44.

[30] Ross, A. M., Rhodes, D. H., & Hastings, D. E. (2007). Defining system changeability: reconciling flexibility, adaptability, scalability, and robustness for maintaining system lifecycle value.

[31] Chamberland, S., Sansò, B., & Marcotte, O. (2000). Topological design of two-level telecommunication networks with modular switches. Operations Research, 48(5), 745-760.

[32] Kim, J. G., & Tcha, D. W. (1992). Optimal design of a two-level hierarchical network with tree-star configuration. Computers & industrial engineering, 22(3), 273-281.

[33] Lee, C. H., Ro, H. B., & Tcha, D. W. (1993). Topological design of a two-level network with ring-star configuration. Computers & operations research, 20(6), 625-637.

[34] Konur, D., & Dagli, C. H. (2015). Military system of systems architecting with individual system contracts. Optimization Letters, 9(8), 1749-1767.

[35] Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. Structural and multidisciplinary optimization, 26(6), 369-395.

[36] Alves, M. J., & Clímaco, J. (2007). A review of interactive methods for multiobjective integer and mixed-integer programming. European Journal of Operational Research, 180(1), 99-115.

[37] Ehrgott, M., & Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. OR-Spektrum, 22(4), 425-460.

[38] Ehrgott, M., Gandibleux, X., & Przybylski, A. (2016). Exact methods for multi-objective combinatorial optimisation. In Multiple Criteria Decision Analysis (pp. 817-850). Springer, New York, NY.

[39] Moreno-Centeno, E., & Escobedo, A. R. (2016). Axiomatic aggregation of incomplete rankings. IIE Transactions, 48(6), 475-488.