A Meta-Model for Representing System-of-Systems Ontologies

Young-Min Baek, Jiyoung Song, Yong-Jun Shin, Sumin Park, Doo-Hwan Bae
Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, Republic of Korea
{ymbaek,jysong,yjshin,smpark,bae}@se.kaist.ac.kr

ABSTRACT

A System-of-Systems (SoS) is a large-scale complex system that integrates multiple constituent systems, which have managerial and operational independence. In order to achieve higher-level common goals of an SoS, it is important to systematically integrate independent constituent systems by thoroughly analyzing and designing the target SoS as a whole. But before conducting these engineering activities, a number of various SoS stakeholders and engineers should be able to understand their SoS. In order to provide a holistic view as a common knowledge base, this paper focuses on developing a conceptual meta-model that represents SoS ontologies. By investigating several documents for Mass Casualty Incident (MCI) response systems, we identified essential objects and required features for SoS descriptions. Based on the investigation, we generalized the objects into SoS entities, and we develop a meta-model, called *M2SoS* (Meta-model for System-of-Systems). To design our *M2SoS*, we borrowed organizational concepts from meta-models for multi-agent systems, and entities and relationships are redefined to specify SoS concepts in *M2SoS*. Finally, *M2SoS* is analyzed with respect to SoS characteristics, and we evaluate if *M2SoS* can represent high-level ontologies for two SoS case scenarios.

KEYWORDS

System-of-Systems, SoS engineering, model-based software engineering, meta-model, ontology, conceptual model

1 INTRODUCTION

A System-of-Systems (SoS) is a type of large-scale complex system that utilizes collaborative capabilities from the integration of component systems. An SoS consists of multiple Constituent Systems (CSs) as components, which have managerial and operational independence [1], and CSs might be designed and developed to achieve their own independent system-level goals. Also, CSs could be heterogeneous systems and geographically dispersed, while moving towards agreed SoS-level common goals [12]. This type of systems has become more widely required in many different domains, such as national defense, climate observation, and disaster response domains. Unlike system engineering of other types of systems, CSs of an SoS are highly autonomous and they have more independent system-level features, so the integration of multiple CSs can lead to the extremely high complexity. For this reason, there have been a lot of needs to handle SoS-specific engineering issues, and recently a number of analysis, design, development and maintenance techniques have been actively researched [2, 8, 10].

To address the scale and complexity issues of an SoS, model-based approaches are essentially required as fundamental engineering activities for SoS analysis, design, and V&V phases [2]. By systematically generating models of an SoS, analyzers and designers can focus on specific parts of the SoS (i.e., separation of concerns), and they can be guided for further engineering activities. Furthermore, to perform comprehensive analysis and design of an SoS, various stakeholders and engineers in an SoS should be able to understand their target system(s) and to establish a common knowledge base. For this purpose, building a meta-model could be one well-established means for creating a knowledge repository that includes SoS-specific features. Using defined vocabularies, SoS engineers can be guided to analyze and design their SoS by referring to classes, relations, and functions.

As an essential step for this purpose while pursuing model-based SoS engineering, we focus on developing a meta-model, called *M2SoS (Meta-model for SoS)*, that can provide a holistic view of an SoS. To identify and define components comprising an SoS, we conducted an investigation of a real SoS case scenario, and identified objects were generalized into SoS-based terms and concepts. By using *M2SoS*, SoS engineers can be guided for their analysis and modeling activities, and they can also inspect completeness or correctness of their models considering SoS-specific characteristics. Also, another *M2SoS*'s capability is to represent SoS ontologies for various SoS domains at the highest-level, and this enables SoS stakeholders (both SoS-level stakeholders and CS-level stakeholders) to establish a common knowledge base of their target SoS. If the ontology generated based on *M2SoS* is stored as a repository, vocabularies defined in *M2SoS* could be accessed and utilized for various engineering purposes.

In this paper, we develop *M2SoS* in the following order. First, we selected a mass casualty incident (MCI) response system as a real SoS case system—we call this system *MCIRSoS*, and relevant documents and papers are collected for review. We analyzed the documents to identify objects comprising of MCIRSoSs and their operations, and parsed objects were generalized to represent SoS-oriented entities. At the same time, we defined several requirements for developing an SoS meta-model based on the investigation. Although we aim at developing a general meta-model for SoSs, there are too many SoSs for us to investigate. Our approach is first to focus on SoSs of a specific domain, and to propose a meta-model and then analyze it with general SoS characteristics. In more detail, to represent organizational aspects of an SoS, we additionally studied meta-model-based approaches for multi-agent systems (MASs).

SESoS'18, May 29, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5747-0/18/05…$15.00
https://doi.org/10.1145/3194754.3194755

Finally, *M2SoS*'s ability to represent an SoS is evaluated with respect to the expression of SoS characteristics and two SoS scenarios.

2 RELATED WORK

In this section, we first take a look into some meta-model approaches for multi-agent systems (MAS). Since an MAS and SoS have some similarities, we reviewed meta-models for MAS to come up with basic ideas and concepts from them. Secondly, we researched some literatures on meta-model or architecture-based methods for SoS.

Meta-modeling approaches of Multi-agent Systems (MAS). A multi-agent system (MAS) shares a lot of common characteristics SoS because an MAS also utilizes emergent capabilities from multiple intelligent component systems (i.e., agents) to fulfill a high-level goal, which is difficult to be achieved by an individual agent. While SoS engineering (SoSE) is an emerging research area, MASs have long been a subject of study in artificial intelligence and software engineering fields. In particular, a number of model-based approaches for MASs have been studied [11], and many meta-models were proposed to develop the model-based techniques [3]. There are two representative meta-models for representing an MAS and modeling integrated agents: *Gaia* Methodology [5], and *O-MaSE* meta-model [4]. Both meta-models focus on organizational aspects of MASs and they consider an MAS as an organization consisting of autonomous and intelligent agents, which have specific roles to participate in higher-level operations. Since those organizational aspects also exist in an SoS to integrate CSs into a higher-level organization, concepts and relationships of MAS meta-models can be borrowed for representation of an SoS organization. Component entities of those two meta-models are briefly listed in Table 1.

First, *Gaia* Methodology was preliminarily proposed to represent an organization of MASs in 2000, and it was improved to additionally cover open agent-based systems in 2003. To describe an organization specifically, *Gaia* contains several organization-centered entities, such as Organization Rule, Structure, and Pattern. And, *Gaia* defines each agent's role, activities, responsibilities, permissions, and properties to describe agent's possible actions. As a fundamental study, *Gaia* has strengths to represent organizational aspects and agents' behaviors at higher levels, but it does not support the goal-oriented approach, which is essentially required for SoS engineering. Secondly, *O-MaSE* is also an organization-centered meta-model, and it focuses on modeling methods to specify goal-role-capability for multiple agents. By additionally providing construction guidelines and method fragments based on *O-MaSE* meta-model, MAS engineers are also enabled to understand the whole activities to produce work products. However, *O-MaSE* is not able to represent managerial and operational independence of CSs either. Therefore, we borrow organization-related concepts and goal-role-capability relationships from both meta-models to represent the integrated CSs and their infrastructure. However, some SoS-specific entities should be additionally defined for representing SoS ontologies, considering SoS-specific characteristics. Even though those two MAS meta-models had been developed relatively long time ago, they can be considered as representative meta-models for describing holistic and organizational views of MASs.

Table 1: MAS meta-models and their component entities

Meta-model	Major component entities
Gaia	Organization, Organization Rule, Organizational Structure, Organizational Pattern, Role, Protocol, Activity, Responsibility, Permission, Safety/Liveness Property
O-MaSE	Actor, Organization, Agent, Role, Goal, Capability, Policy, Environment Object Type, Environment Property, Domain Model, Protocol, Message, Plan, Action

Modeling and Architecture of SoS. Architectures for representing SoS structures have been broadly studied across the academic and industrial fields. A study by Eusgeld *et al.* [6] proposed a general architecture for modeling of SoS, and it consists of SoS-level, CI-level, and system-level components hierarchically. The most well known application of SoS architectures to industrial problems could be the *Department of Defense Architecture Framework* (*DoDAF*, currently version 2.02) [13]. As a guideline to design a large-scale complex system for the national defense, *DoDAF* provides a wide variety of architectural viewpoints and models. DoDAF mainly focuses on the utilization of data for SoS-level operations, several viewpoints and data representations were defined in its architectures. For another architecture, *AMADEOS European project* developed conceptual models to design a Cyber-Physical SoS (CP-SoS) using SysML packages (profiles), and it supports viewpoint-based architecting [10]. *AMADEOS* introduced multiple viewpoints with subpackages, and a set of vocabularies is provided by defining detailed components.

To implement SoSs, many researchers also have studied SoS reference architectures [7, 9, 14]. The reference architecture is an architecture that provides a common vocabulary for system implementation. It derives domain commonality among stakeholders. Perez *et al.* [7] took advantage of their project experience and presented a reference architecture for the smart grid SoS, which consists of 6 layers: physical, communication, component, service, orchestration, and (Big) data layer. The main characteristics of Perez *et al.* 's reference architecture are scalability and dynamicity of adding functionalities. Oliveira *et al.* [14] suggested a reference architecture called *RefTEST-SOA*. *RefTEST-SOA* comprises how to structure testing tools, considering integration, scalability, and reuse. Its purpose is to support software testing based on service-oriented architecture. Another paper of Oliveira *et al.* [9] suggested *ArchSORS* that guides the systematic design of service-oriented robotic systems. Unlike the above researches, our approach is more likely to focus on understanding SoS itself rather than on testing and implementing SoS.

3 ELICITATION OF REQUIREMENTS FOR DEVELOPING SOS META-MODEL

In order to develop a meta-model for SoS representation as shown in the overall approach (See Figure **??**), we first investigated several literatures related to Mass Casualty Incident (MCI) response systems which are real cases of SoSs. Through the investigation, we identify a number of objects comprising the response systems, and they are generalized into SoS-specific terms for developing a meta-model. Although our meta-model is built based on a specific domain (i.e., MCI response), every identified object is generalized to represent domain-general SoS entities. Therefore, the meta-model can be applied to various SoS domains using generalized entities,

and then the entities can be specialized to represent concepts in specific domains. In this chapter, we explain the MCI case scenario and why the MCI response systems are real SoSs in Section 3.1, and we define four requirements for developing an SoS meta-model in Section 3.2.

3.1 MCIRSoS: Real SoS Case Analysis

Generally, a Mass Casualty Incident (MCI) refers to a large-scale incident that the number and severity of casualties are so high that medical and rescue resources are lacking [15]. Major causes of an MCI can be both huge (human-made) disasters such as explosion, building collapse, and natural disasters like earthquakes or forest fires. Unlike typical accidents, MCIs are declared in the case of incidents that have explosively growing casualties. When an MCI situation is declared, there must be the shortage of resources which are responsible for incident handling, thus existing systems have a limit to support appropriate rescue and treatment operations in time. Because of these reasons, the MCI situation is difficult to be solved by individual systems and it is necessary to construct a high-level system to continuously provide complex MCI-response services based on the collaboration of appropriate constituent systems. In other words, in order to respond to MCI effectively, we need to develop an SoS to achieve high-level complex requirements and goals beyond the capabilities of existing systems.

In this paper, we call the SoS for coping with an MCI an *MCIRSoS* (*MCI-Response SoS*). The first reason why we select this MCI response case is that an MCIRSoS has major characteristics and features of SoS, such as autonomy, belonging, connectivity, diversity, and emergence. The second reason is because there are a number of specialized materials (documents) for mass casualty response and planning, so we can easily access to them for referencing real SoS case scenarios. We first collected about 30 MCI-related documents and we selected reliable materials only which were written by professional organizations (e.g., *WHO*, counties, official committees or associations). Those documents mainly include MCI response procedures or disaster response plans, and they also establish several policies to control component systems (or agencies) and their communications. The investigated documents are listed on our web page[1].

3.2 Requirements for Developing a Meta-Model

After investigating MCI response documents, we could determine some characteristics that should be considered to build a meta-model for SoS representations. To do that, we specify four major requirements that *M2SoS* should consider and fulfill as follows:

Requirement 1: Distinguishing SoS-level Entities and CS-level Entities. According to the documents of MCI response systems, we could find that entities utilized for MCI response can be classified into two levels: SoS-level entities and CS-level entities. Most of SoS-level entities are created, performed, and managed by SoS-level managers such as an incident commander of MCIRSoS. Also, SoS-level entities exist to integrate lower-level systems and resource, and they move towards to achieve a high-level global goal of an SoS. For this purpose, those entities are usually included in the SoS-level infrastructure. On the other hand, CS-level entities are including functional aspects of CSs. Since CSs should provide their functionalities and resources to perform higher-level missions, they do the assigned tasks or interact with other CSs for the exchange of information. Detailed information of CSs is independent from SoS-level management, thus it is not necessarily provided to the SoS-level managing team. Therefore, our meta-model should be able to explicitly distinguish SoS-level entities and CS-level entities so that CS-level and SoS-level managers (or engineers) can focus on their own parts.

Requirement 2: Defining SoS-level Goals, Requirements, Services, Behaviors and relationships among these entities. An SoS exists to achieve a specific SoS-level goal(s), and the goal is established by analyzing a specific target problem. This means that SoS engineering must conform to goal-oriented approaches. To achieve the goal, SoS-level engineers and other stakeholders should elicit SoS-level requirements thoroughly and define services to be provided or performed. The services can be decomposed into a set of behaviors, which are done by cooperation of multiple CSs. Most MCI response documents define those soft entities (i.e., goal, requirements, services, and behaviors) and they analyze the relationships among them. Therefore, in order to specify SoS-level entities adequately, *M2SoS* should provide the definition of the entities and specify the relationships among them to guide modeling activities.

Requirement 3: Defining SoS-level Environmental Factors. For the MCI response, MCI situation is the most crucial part and the infrastructure of MCIRSoS should be able to effectively monitor and analyze it. Since an MCI is closely related to the environment where the incident occurs, environmental factors can highly influence overall MCIRSoS operations. Actually, most MCI documents listed possible situations and environmental inputs, and they plan specific procedures to respond them. In MCI documents, environmental factors could be divided into four categories—*resource*, *physical environment*, *human factor*, *threat*, and they can be classified (i.e., generalized) into SoS-level *Resource*, *Society*, and *Physical Environment*. For example, *Resource* refers to the materials, money, systems, or people that an SoS can utilize when performing SoS-level services. *Physical Environment* refers to the environmental factors that exist in physical forms and are measurable such as temperature, volume of traffic, etc. Consequently, those environmental entities should be contained as explicit components in our meta-model so as to identify uncertainties in the environment and make decisions for the SoS-level operations.

Requirement 4: Developing a Meta-model for Representing SoS Ontologies. An ontology is one of the specification methods to represent specific domain knowledge as a set or a model by defining classes, relations, and functions. Most MCI response documents try to define common dictionaries and they build global glossaries for the communication between systems, however there is no systematic way to build the common knowledge base. By defining an ontology, SoS-level engineers can be enabled to obtain overall domain knowledge of a subject SoS, and an ontology enables CS-level managers and engineers to perform communication. In particular, in order that our meta-model supports various model-based techniques for diverse domains, developing a meta-model that can represent SoS ontologies could be an effective and domain-general way to represent an SoS.

[1] http://se.kaist.ac.kr/starlab/research/m2sos/mci-investigation/

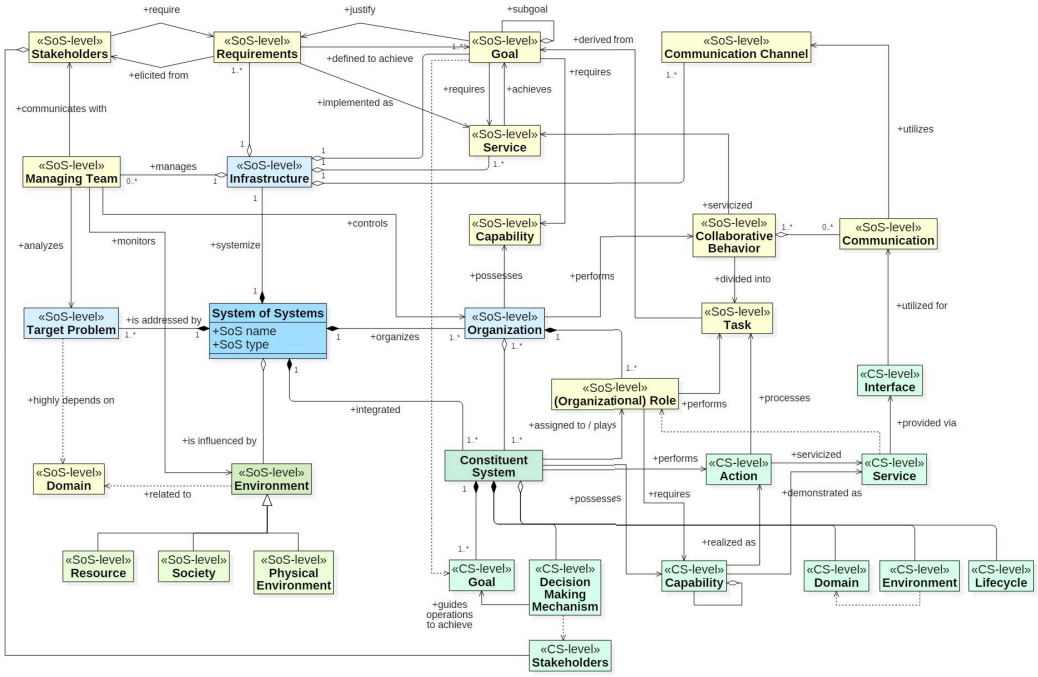

Figure 1: Simplified version of *M2SoS*

4 DEVELOPMENT OF A META-MODEL

In this section, we develop a meta-model for SoS analysis and design, called *M2SoS* (Meta-model for System-of-Systems). Based on our investigation explained in Chapter 3, we define requirements for developing *M2SoS* and we explain steps for defining entities and relationships of *M2SoS*.

Basically, we build the structure of *M2SoS* based on MAS meta-models, which were explained in Chapter 2, and we borrow several concepts and relationships they use for representing organizations. In order to include and represent SoS-specific characteristics, we add some generalized concepts identified by MCI response documents. In this section, steps to define entities of *M2SoS* are explained and the overall *M2SoS* is introduced. A simplified version of *M2SoS* including high-level entities is shown in Figure 1, and the original version is uploaded on our web site[2].

Step 1. Definition of Primary Components of an SoS.

We first divide identified entities into two levels: SoS-level entities and CS-level entities. At the highest level, an SoS consists of four major components: *SoS-level Target Problem*, *SoS-level Infrastructure*, *SoS-level Environment*, and (multiple) *Constituent Systems*.

- **SoS-level Target Problem** refers to an explicitly defined high-level problem, to be solved by integrating multiple CSs' capabilities, which is closely related to SoS-level goal(s).
- **SoS-level Infrastructure** refers to a social and physical infrastructure that is established to solve the SoS-level Target Problem by integrating and orchestrating multiple CSs. This infrastructure determines a boundary of an SoS, and it provides network facilities for CS communications.
- **SoS-level Environment** refers to external entities that interact with SoS-level infrastructure and constituent systems
- **SoS-level Organization** refers to an organization that integrates multiple constituent systems to perform specific collaborative behaviors. To derive higher-level capabilities from the organization, specific Roles are defined and assigned to CSs.
- **Constituent System** refers to an independent unit system of an SoS that performs partial functions of SoS-level services based on the assigned role. Both a legacy (i.e., existing) system and a new system can be a constituent system.

Step 2. Definition of SoS-level Entities.

SoS-level entities are defined, created, managed, and analyzed by an SoS-level managing team (or engineering team) at the infrastructure level. In other words, to effectively manage the SoS-level infrastructure for the goal achievement, SoS-level entities should be analyzed and designed throughout the whole phases of SoS engineering. In *M2SoS*, SoS-level entities are defined in Figure 1, and each entity is described as follows.

- **SoS-level Goal** refers to a common and agreed-upon global goal of an SoS to solve the SoS-level Target Problem(s).
- **SoS-level Requirements** refers to the specific conditions or capabilities to be satisfied for SoS-level goal achievement. The requirements are specialized into three categories: Operational Requirements, Development Requirements, Design Requirements.
- **SoS-level (Organizational) Role** refers to a particular position of a CS(s) in an SoS to partially function SoS-level operations. The Organizational Role is assigned based on CS's capability, and it determines specific CS-level Services (and Actions) as a part of an SoS.
- **SoS-level Service** refers to the high-level service that can only be performed through multiple CSs' cooperation. A series of SoS-level behaviors forms a single SoS-level service, and the behaviors are classified into sequential, resultant, and emergent behaviors.
- **SoS-level Capability** refers to the high-level capability demonstrated by SoS-level Services based on CSs' cooperation.

[2]http://se.kaist.ac.kr/starlab/research/m2sos/conceptual-models/

- **SoS-level Communication Channel** refers to the (virtual or physical) facility of the SoS-level infrastructure that is used for the communication between CSs and between CS and SoS-level entities.
- **SoS-level Managing Team** refers to the managing or engineering team who performs administrative duties such as management of CSs, resources, risks/failures, dynamicity (dynamic reconfiguration, evolution), V&V, planning, and monitoring.
- **SoS-level Domain** refers to the high-level field where the SoS belongs to. A domain model represents domain knowledge by including domain-specific concepts and relations between concepts.
- **SoS-level Stakeholders** refer to any people who have an interest in either SoS-level development or SoS-level goal achievement.

Step 3. Definition of CS-level Entities.

The lower-level entities of an SoS are to specify the components of a CS. Since a CS itself is a complete system, it contains concrete property information of a general type of system. Additionally, to represent CS's managerial and operational independence, some self-contained and independent information is represented in *M2SoS*. Most importantly, CS-level entities may not be fully provided to the SoS-level managing team because they are not originally designed to be managed by higher-level systems. Entities defined are described below, and they are represented in *M2SoS* as Figure 1.

- **CS-level Goal** refers to CS's independent goal that the CS originally targets for to create specific business (or service) value.
- **CS-level Capability** refers to the capability possessed by a CS, and CS's capabilities are used to analyze and assign CS's role in an SoS.
- **CS-level Service** refers to the set of functionalities served by a CS by demonstrating its own capabilities. To perform MCI-level operations, each service is provided via specific interfaces.
- **CS-level Interface** refers to the functional boundary of a CS to provide CS-level Service or to share the information.
- **CS-level Decision Making Mechanism** refers to the mechanism for CS's decisions (e.g., cost-benefit analysis, belief-based decision).
- **CS-level Stakeholders** refers to people who have interest in CS-level services and goal achievement.
- **CS-level Domain** refers to the field where an individual CS belongs to. A CS-level domain can be represented as a domain model to define concepts and relations.
- **CS-level Lifecycle** refers to the independent lifecycle for CS development, maintenance, and phase-out description.
- **CS-level Environment** refers to the entities that exist outside of a CS and interact with the CS in some way.

Step 4. Definition of SoS-level Environment Entities.

As Requirement 4 specified in Section 3.2, we need to thoroughly analyze environmental factors of an SoS because an SoS and CSs actively interact with the environment. Based on the investigation, environmental factors are classified into resource, physical environment, society, and threat. These factors have high uncertainty, thus these entities should be identified and analyzed to plan SoS-level behaviors and to prevent the SoS infrastructure from deviating from expected behaviors as well.

- **SoS-level Resource** refers to the materials, money, systems, or people that an SoS can utilize when performing SoS-level services.
  - *Resource State: available resource* or *potential resource*
- **SoS-level Physical Environment** refers to the environmental factors that exist in physical and measurable forms (e.g., temperature, scale of severity, volume of traffic)
- **SoS-level Society** refers to the external organization or system that includes human factors and affects SoS's decision making (e.g., related authorities, government, the public).

Table 2: Analysis of *M2SoS* based on 8 dimensions

Dimension	Related entities in *M2SoS*	
Autonomy Independence	Distinguishing SoS and CS-level entities	CS-level Service, Capability, Role
		CS-level Goal, Decision Making Mechanism, Domain, Lifecycle
Distribution	SoS-level Environment and CS-level Environment	
Evolution	Dynamicity Management by Managing Team	
Dynamic Reconfiguration	SoS-level Plan created by Planning operation, Dynamicity Management by Managing Team	
Emergence of Behavior	SoS-level Behaviors classification: Resultant, Emergent Behaviors	
Interdependence	SoS-level Communication, CS-level Capability	
Interoperability	All entities defined in SoS-level Infrastructure, CS-level Service, CS-level Interface, SoS-level Communication, Communication Channel	

- **SoS-level Threat** refers to the external causes (hazards) of SoS's malfunction or failure.

5 ANALYSIS OF M2SOS

5.1 Representations of SoS-specific Features

Nielsen *et al.* introduced 8 dimensions for model-based SoS engineering [2], and the dimensions include SoS-specific characteristics and features, which were defined in existing studies. 8 dimensions are autonomy, independence, distribution, evolution, dynamic reconfiguration, emergence of behavior, interdependence, and interoperability. Because we can consider *M2SoS* can reflect overall SoS characteristics if *M2SoS* has means of representing 8 dimensions, we analyze our *M2SoS* based on the 8 dimensions.

Table 2 summarizes the analysis of *M2SoS* based on 8 dimensions. We checked how *M2SoS* covers each dimension using entities defined in the meta-model. For example, *M2SoS* represents autonomy and independence, by explicitly classifying SoS-level and CS-level entities to distinguish tasks covered by SoS-level engineers and CS-level engineers. Also, evolutionary development and dynamic reconfiguration could be managed by *Dynamicity Management* operated by *SoS-level Managing Team*. For the SoS-level behaviors, *M2SoS* explicitly includes two types of behaviors, which are resultant and emergent behaviors, and interrelationships are represented by defining entities related to *SoS-level Communication*, *CS-level Capability*, *Service*, and *Interface*. The results show that each dimension can be adequately represented using entities defined in *M2SoS*, thus *M2SoS*-based models or ontologies can be utilized for further model-based analysis and design of SoS. Also, if an engineer focuses on a specific dimension, he or she can figure out whether related entities are designed well or not.

5.2 Scenario-based Analysis of *M2SoS*

M2SoS should be able to represent all the related components of an SoS so that the *M2SoS* can provide a complete ontology of the SoS. We analyzed the *M2SoS* with two SoS scenarios, MCI-Response SoS (MCIRSoS) and Research Team SoS (RTSoS) to show whether the *M2SoS* can represent all components of SoS or not. The MCIRSoS scenario and the RTSoS scenario show a collaboration of independent systems to respond an MCI situation and to publish papers, each. While MCIRSoS cases are real-world examples, RTSoS cases are frequently implemented as general multi-agent and SoS cases [16]. We identified that all components of two SoS scenarios correspond to one of the entities of *M2SoS*. Table 3 shows

Table 3: Analysis of two example SoS case scenarios using *M2SoS*

		MCI Response SoS (MCIRSoS)	Research Team SoS (RTSoS)
SoS Scenario		The MCIRSoS scenario shows collaboration of independent systems to respond mass casualty incident situation. Each system (e.g. fire fighting system and emergency medical system) has its own capabilities and goals of rescuing, treating, or transporting patients. Although constituent systems (CSs) operate autonomously and provide their own services, when a mass casualty incident occurs, they collaborate to achieve SoS-level goal that is not achievable by an individual CS.	The RTSoS scenario shows collaboration of researchers for research and publishing. Each researcher represents an independent system and has an SoS-level goal of publishing. Each researcher has own capabilities and CS-level goals which are related to his or her research interest. When more than one researchers have same goals or interests, they connect with each other for the collaboration. Once they achieve the SoS-level goal, they try to achieve their own CS-level goals again.
SoS-level Target Problem		• Occurrence of patients in the MCI situation	• Need of publications for graduation
SoS-level Domain		• Response to MCI, large accident response plan	• Engineering research, software engineering, business process management
SoS-level Infrastructure	SoS-level Requirements	• Fire fighting systems and medical systems should follow instructions given by the SoS manager. • Emergency medical systems should be able to transport patients as quickly as possible through communication with the medical center. • An SoS manager should operate resources and CSs effectively.	• The researcher should attend research team's official schedule. • Minimum number of researchers are needed to operate the laboratory. • Researcher should be able to communicate in Korean and English.
	SoS-level Goal	• To rescue more than 95% of patients • To save patients more than 80% of survival in the MCI situation	• To publish 10 papers including five international journals per year • To perform more than two projects per year
	SoS-level Service	• Shortening the rescue time through collaboration of 911 dispatch systems and fire fighting systems • Rescuing and transferring patients by cooperation of fire fighting systems and emergency medical systems	• Writing international and domestic papers • Creating yearly project reports and presentations • Holding a workshop in the laboratory
	SoS-level Communication Chanel	• Directed channels for data transmission such as disaster message, telephone, and radio	• Regular and occasional meetings, ordinary communications through emails, and regular seminars
	SoS-level Stakeholders	• Government, residents of the MCI region, and donors etc.	• Professors, students, laboratory staff, affiliated school, and research project parties etc.
	SoS-level Managing Team	• A command center consisting of MCI managers	• A laboratory management team consisting of professors and doctoral students
SoS-level Environment	SoS-level Resource	• Potential transportation such as buses and taxis • Financial resources such as government funds and donations	• Funds for the operation of the laboratory • Material resources of laboratory such as computers, printers, and office supplies
	SoS-level Society	• Various groups to response to MCI such as MCI disaster response headquarters and disaster response teams	• Software engineering research communities
	SoS-level Physical Environment	• Geographical environment (e.g. road condition and traffic lights) and physical environment (e.g. climate and weather)	• Laboratories, laboratory buildings, and school buildings
Constituent Systems (CSs)		**[911 dispatch system]** • Analyzing incidents using reported information • Identifying scale and severity of MCI and sharing tasks to each CS	**[Professor]** • Guiding the research direction to the students as an advisor of laboratory • Teaching writing skill to improve the quality of the student's paper, and obtaining projects to run the laboratory
		[Fire fighting system] • Rescuing the patients by utilizing the patients' location information • Extinguishing a fire by using suitable equipment to the MCI situation	**[Doctoral student]** • Leading the research teams and projects • Publishing the papers and conducting meetings with Master's students
		[Emergency medical system] • Providing first aid to the patients • Transporting patients to the medical center in short period	**[Master's student]** • Participating in ongoing projects in the laboratory • Publishing the papers and assisting in writing project reports
		[Medical center] • Accommodating patients in hospitals • Providing appropriate treatments after analyzing the condition of transferred patients	**[Patent attorney]** • Providing advices related to patent applications • Searching patents to prevent patent violations

the analysis results of matching the two SoS scenarios to the *M2SoS*. Every entry in Table 3 is derived by the entities in simplified version of the *M2SoS*.

We found out that the *M2SoS* can support SoS modeling by providing general SoS entities and ontology for the two example of SoS scenarios. In Table 3, the requirements of developing the meta-model for SoS is satisfied. Every component in the example SoS scenarios can be described in ontologies of the *M2SoS*, which was not emphasized in the existing SoS meta-model researches [6, 7, 9, 10, 13, 14]. In other words, we construct the SoS scenarios by the entities of *M2SoS*. Therefore, SoS managers can generate a model of SoS with various modeling techniques after defining all the components of a target SoS conforming to *M2SoS*.

6 THREATS TO VALIDITY

For now, *M2SoS* was not applied to other real SoS cases or industrial scenarios. However, it was preliminarily developed to represent an SoS to be analyzed and designed. We expect models or ontologies, which are developed based on *M2SoS*, to be a common knowledge base for engineers' communications. For further engineering processes, it is clearly obvious that *M2SoS* should be improved by applying multiple viewpoints or by describing an SoS in multi-layered architectures. 8 dimensions can be used as reference points to extend *M2SoS*'s expressive power. After the extension, we plan to apply the improved *M2SoS* to various types of other industrial and enterprise SoS cases.

We are not providing specific modeling methods or guidelines for modeling the entities defined in *M2SoS*. Current version of *M2SoS* is technique-neutral meta-model, and it can be used for both modeling activities and ontology definitions. Even so, to support complete and systematic analysis and design of an SoS, we are now developing modeling methods for goal and policy specifications, collaboration and negotiation mechanisms, and so on. Also, *M2SoS* will be first applied to generate a minimized set of executable SoS models for simulation-based verification.

7 CONCLUSION

In this paper, we introduce a meta-model for representing SoS ontologies, called *M2SoS* (Meta-model for System-of-Systems), based on the investigation of Mass Casualty Incident (MCI) response systems. According to the investigation, we could define four requirements to build a meta-model: (i) Distinguishing SoS-level and CS-level entities, (ii) Defining SoS-level Goals, Requirements, Services, Behaviors and relationships between these entities, (iii) Defining and classifying SoS-level environmental factors for uncertainty analysis, (iv) Developing a meta-model for representing SoS ontologies. To satisfy the requirements, we developed *M2SoS* that can adequately represent major characteristics of SoS and ontologies of SoS case scenarios. Entities of *M2SoS* were defined based on the investigation of a real SoS case, thus it could be a meaningful result to analyze SoS documents and SoS-specific characteristics. We expect *M2SoS* to provide a holistic view for SoS engineers and to be utilized for representing an ontology of a target SoS at the design phase and during runtime as well. For further application, we plan to develop specific modeling methods for each entities defined in *M2SoS*, and the modeled entities will be inputs of a tool for simulation and verification.

ACKNOWLEDGEMENTS

This research was supported by the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. R0126-18-1101, (SW Star Lab) Software R&D for Model-based Analysis and Verification of Higher-order Large Complex System), and Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT (2017M3C4A7066212).

REFERENCES

[1] J. Boardman and B. Sauser. 2006. System of Systems - the meaning of of. In *2006 IEEE/SMC International Conference on System of Systems Engineering*. 6 pp.–. https://doi.org/10.1109/SYSOSE.2006.1652284

[2] J. S. Fitzgerald J. Woodcock C. B. Nielsen, P. G. Larsen and J. Peleska. 2015. Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions. *ACM Comput. Surv.* 48 (2015), 18:1–18:41.

[3] M. P. Gleizes P. Turci C. Bernon, M. Cossentino and F. Zambonelli. 2005. A Study of Some Multi-agent Meta-models. In *Agent-Oriented Software Engineering V*, James Odell, Paolo Giorgini, and Jörg P. Müller (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 62–77.

[4] S. A. DeLoach and J. C. Garcia-Ojeda. 2010. O-MaSE: a Customisable Approach to Designing and Building Complex, Adaptive Multi-Agent Systems. *Int. J. Agent-Oriented Softw. Eng.* 4, 3 (Nov. 2010), 244–280. https://doi.org/10.1504/IJAOSE.2010.036984

[5] N. R. Jennings F. Zambonelli and M. Wooldridge. 2003. Developing Multiagent Systems: The Gaia Methodology. *ACM Trans. Softw. Eng. Methodol.* 12, 3 (July 2003), 317–370. https://doi.org/10.1145/958961.958963

[6] C. Nan I. Eusgeld and S. Dietz. 2011. "System-of-systems" approach for interdependent critical infrastructures. *Reliability Engineering & System Safety* 96 (06 2011), 679–686.

[7] J. Garbajosa A. Yagüe E. Gonzalez M. Lopez-Perea J. Pérez, J. Díaz. 2015. Towards a reference architecture for large-scale smart grids system of systems. In *Proceedings of the Third International Workshop on Software Engineering for Systems-of-Systems*. IEEE Press, 5–11.

[8] M. Jamshidi. 2008. System of systems engineering - New challenges for the 21st century. *IEEE Aerospace and Electronic Systems Magazine* 23, 5 (May 2008), 4–19. https://doi.org/10.1109/MAES.2008.4523909

[9] K. R. Felizardo F. Oquendo L. B. R. Oliveira, E. Leroux and E. Y. Nakagawa. 2017. ArchSORS: A Software Process for Designing Software Architectures of Service-Oriented Robotic Systems. *Comput. J.* 60, 9 (2017), 1363–1381.

[10] P. Lollini B. Frömel F. Brancati M. Mori, A. Ceccarelli and A. Bondavalli. 2017. Systems-of-systems modeling using a comprehensive viewpoint-based SysML profile. *Journal of Software: Evolution and Process* (2017), e1878–n/a. https://doi.org/10.1002/smr.1878 e1878 JSME-16-0093.R2.

[11] A. Eberlein M. Moshirpour, N. Mani and B. Far. 2013. Model Based Approach to Detect Emergent Behavior in Multi-agent Systems. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS '13)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1285–1286. http://dl.acm.org/citation.cfm?id=2484920.2485185

[12] M. W. Maier. 1996. Architecting Principles for Systems-of-Systems. *INCOSE International Symposium* 6, 1 (1996), 565–573. https://doi.org/10.1002/j.2334-5837.1996.tb02054.x

[13] U.S. Department of Defense. 2010. DoD Architecture Framework (DoDAF) ver 2.02. (2010). http://dodcio.defense.gov/Library/DoD-Architecture-Framework/

[14] L. B. R. Oliveira and E. Y. Nakagawa. 2011. A service-oriented reference architecture for software testing tools. In *European Conference on Software Architecture*. Springer, 405–421.

[15] World Health Organization. 2007. *Mass casualty management system : strategies and guidelines for building health sector capacity*. Geneva : World Health Organization.

[16] B. Sauser W. C. Baldwin and R. Cloutier. 2015. Simulation Approaches for System of Systems: Events-based versus Agent Based Modeling. *Procedia Computer Science* 44 (2015), 363 – 372. https://doi.org/10.1016/j.procs.2015.03.032 2015 Conference on Systems Engineering Research.