(2018 KCSE 우수논문) 시스템 오브 시스템즈 온톨로지 구축을 위한 사례 분석 기반의 메타모델 개발

Analysis of Case Scenario to Develop a System of Systems Meta-model for Ontology Representation

백 영 민[1] (Young-Min Baek) 박 수 민[2] (Sumin Park) 신 용 준[3] (Yong-Jun Shin) 배 두 환[4] (Doo-Hwan Bae)

요 약 온톨로지(ontology)는 시스템과 정보 도메인 상에 존재하는 다양한 개념과 관계를 체계적으로 정의하는 기법으로, 특정 목적을 위한 공통된 지식 기반을 구축하고 상호작용의 불일치를 줄이기 위해 활용된다. 시스템 오브 시스템즈(System-of-Systems, SoS)는 여러 독립적인 구성 시스템들의 협력을 통해 목표를 달성하는 매우 크고 복잡도가 높은 시스템으로, SoS 전반에 걸친 설계와 개발을 위해서는 앞서 설명한 온톨로지의 구축이 요구된다. 하지만 이러한 온톨로지 구축을 위해서는 도메인 전문가가 개발 대상 시스템과 정보 도메인을 효과적이고 체계적으로 표현하기 위한 도구가 필요하다. 본 연구는 이러한 목적을 달성하기 위해, SoS 온톨로지 구축을 위한 모델 기반 기법으로 M2SoS (Meta-model for SoS)을 사례 분석을 바탕으로 개발하고자 한다.

키워드: 시스템 오브 시스템즈, 메타모델, 소프트웨어 시스템 모델링, 온톨로지, 사례 분석 연구

Abstract Ontology is a formal and explicit specification technique that defines concepts and relationships of a system. It is utilized to establish a common knowledge base and to reduce mismatches or inconsistencies during communication. Since a System-of-Systems (SoS) is a large-scale complex system that achieves higher-level common goals by the collaboration of constituent systems, ontologies need to be established for overall SoS development and operations. In other words, refined development and communications among various stakeholders of an SoS can be achieved based on the conceptualization power of an ontology. However, in order to build an ontology effectively, SoS engineers require a systematic means to provide a guideline for domain analysis and ontology establishment. To fulfill these requirements, this study proposes a meta-model, called the Meta-model for System-of-Systems (M2SoS), which enables systematic specifications of ontologies for SoS development. M2SoS is developed based on existing studies on meta-modeling approaches in the multi-agent system domain, but M2SoS is improved to meet SoS-specific requirements by SoS case analysis.

Keywords: system-of-systems (SoS), meta-model, software system modeling, ontology, case study

1. 서 론

현대의 많은 소프트웨어 중심 시스템들은 서로 연결 및 통합되어 복잡한 기능과 상위 수준의 서비스 제공하기 위한 요구가 점차 늘어나고 있다. 이에 따라 공통된 상위 수준의 목표를 달성하기 위한 서비스 제공을 위해, 상위 수준의 대규모 복잡 시스템(large-scale complex system)을 구성하고 다수의 컴포넌트 시스템을 통합하여 구성 시스템들의 협력을 통한 시너지 효과(synergy effect)를 얻는다. 이러한 상위 수준의 목표는 독립적인 각 구성 시스템들의 역량으로는 달성할 수 없지만 여러 시스템들의 협력을 통해 달성할 수 있는 수준의 목표를 의미하며, 다양한 분야에서의 복잡한 문제 해결을 위해 위와 같은 시스템의 요구가 많아지고 있다. 이처럼 컴포넌트 시스템들의 통합을 필요로 하는 시스템이 활용되는 대표적인 분야로 국방 시스템 [1], 기후 관측 시스템 [2], 재난 및 사고 대응 시스템 [3], 공공 시스템 및 서비스 [4] 등이 있다.

이러한 요구에 따라, 상위 수준의 공통 목표를 달성하기 위해 여러 시스템들을 통합한 다양한 종류의 시스템(IoT, CPS 등)이 개발되고 있다. 그 중 여러 독립적인 시스템들의 통합으로 만들어지는 대형 복잡 시스템을 시스템 오브 시스템즈(System-of-Systems, 이하 SoS)라고 정의하는데, SoS는 개별 시스템이 달성할 수 없는 상위 수준의 공통 목표를 이루기 위해 여러 구성 시스템(Constituent System, 이하 CS)들을 통합하고 그로부터 상위 수준의 기능 및 서비스 수행 역량을 기대하는

• 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No. 2015-0-00250, (SW 스타랩) 모델 기반의 초대형 복잡 시스템 분석 및 검증 SW 개발)
• 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보·컴퓨팅기술개발사업의 지원을 받아 수행된 연구임(NRF-2017M3C4A7066212)
• 이 논문은 2018년도 한국 소프트웨어공학 학술대회(KCSE 2018)에서 '시스템 오브 시스템즈(SoS) 사례 분석을 통한 온톨로지 기반의 SoS 메타모델 개발'의 제목으로 발표된 논문을 확장한 것임

[1] 학 생 회 원 : 한국과학기술원 전산학부
ymbaek@se.kaist.ac.kr (Corresponding author임)
[2] 학 생 회 원 : 한국과학기술원 전산학부
smpark@se.kaist.ac.kr
[3] 학 생 회 원 : 한국과학기술원 전산학부
yjshin@se.kaist.ac.kr
[4] 종 신 회 원 : 한국과학기술원 전산학부 교수
bae@se.kaist.ac.kr

대규모 복잡 시스템을 말한다 [5, 14]. 일반적으로 이러한 SoS는 효과적인 상위 목표 달성을 위해 체계적인 설계와 개발이 요구되는데 [8, 9], SoS를 이루는 각 CS는 독립적으로 운영과 관리가 이루어지는 이종의 시스템들이기 때문에 이를 위한 분석의 복잡도가 매우 높으며 많은 비용이 필요하다 [6].

이처럼 복잡하고 규모가 큰 SoS의 공학적인 개발, 운영 및 검증을 위해서는 다수의 이해관계자와 엔지니어가 개발 대상 시스템에 대해 일정 수준 이상의 공통된 지식을 가지고 있어야 한다. 만약 독립적인 여러 CS들의 관리자가 상위 수준 SoS와 목표를 일관성 없고 올바르지 않게 파악할 경우, 상위 수준의 목표를 달성하기 위해 수행하는 CS간의 협업을 통한 운영이 어려워질 수 있다. 온톨로지(ontology)는 이와 같은 시스템의 개발 및 운영의 문제를 해결하기 위해 시스템과 정보 도메인 상에 존재하는 여러 개념과 관계를 정의하는 기법이며, 관심 도메인에 대한 공유된 이해(shared understanding)를 가능하게 한다. 특히 여러 이종의(heterogeneous) 독립적인 시스템이 연합되어 있는 대규모 SoS에서는, 체계적인 통합 개발과 운영 관리를 위해 공통된 지식 베이스를 구축해야 할 필요성이 있으므로 온톨로지의 필요성이 더욱 강조된다. 그러나 이러한 온톨로지 구축을 위해서는 도메인 전문가와 엔지니어가 개발 대상이 되는 시스템과 정보 도메인(information domain)을 효과적이고 체계적으로 표현하기 위한 기법과 도구가 함께 지원되어야 한다.

본 논문은 온톨로지 구축을 위한 다양한 방법 중 모델 기반 기법에 초점을 맞춘다. 그리고 SoS 온톨로지 표현 및 구축을 위한 지침(guideline)으로 활용할 수 있는 모델 기반 도구인 메타모델을 개발하고자 한다. 기존의 여러 모델 기반 연구처럼, 대상 SoS를 온톨로지 형식의 개념화(conceptualization)를 수행하기 위해 메타 수준에서 정의된 모델과 규칙을 활용한다. 이어서 온톨로지를 하나의 인스턴스 모델로 간주하여, 온톨로지를 정의하기 위한 정보(모델 요소와 규칙)를 메타모델(meta-model) 형식으로 정의하고자 한다. 본 논문에서 제안하는 메타모델인 M2SoS (Meta-model for SoS)는 SoS와 가장 유사한 시스템 도메인인 다중 에이전트 시스템(multi-agent system)의 메타모델 기법에 기반하여 개발된다. 또한, M2SoS가 SoS 도메인의 다양한 온톨로지를 효과적으로 표현할 수 있도록, 실제 SoS 사례 시나리오 조사를 수행하여 얻어진 구체적인 메타모델 개발 요구사항에 따라 M2SoS의 요소와 관계를 정의한다.

본 논문의 구성은 다음과 같다. 2장에서 본 연구의 대상이 되는 시스템 오브 시스템즈와 다중에이전트 시스템(MAS)을 메타모델 관점에서 분석한다. 3장에서는 SoS 분석을 위한 메타모델을 개발하기 위한 사례로 다중손상사고 사례 시나리오를 조사 및 분석하고, 4장에서는 구체적인 요구사항에 기반해 SoS 분석 및 설계를 위한 메타모델(M2SoS)을 정의한다. 5장에서는 현재 개발된 M2SoS의 활용 방안에 대해 논의하고, 6장에서는 결론 및 향후 연구를 기술한다.

2. 배경: 다중 에이전트 시스템 메타모델

대표적인 SoS이자 가장 유사한 시스템으로 다중 에이전트 시스템(Multi-agent system, MAS)이 있다 [15]. 자율적인 에이전트 기반 시스템인 MAS는 다수의 에이전트(agent)로 구성된 시스템을 가리키며, 각 에이전트는 매우 자율적이고 적극적인 의사 결정 메커니즘(decision-making mechanism)을 기반으로 특정 환경 상에서 상위 목표를 달성하기 위한 구체적인 임무를 역할에 따라 수행한다. MAS는 SoS와 같이 비선형적 행위를 보이는 에이전트로 조직(organization)이 구성될 뿐만 아니라 내/외부의 활발하고 복잡한 상호작용을 통해 창발적(emergent) 현상과 행위가 발생할 수 있는 시스템이다. 또한, MAS를 이루는 에이전트는 자율성(autonomy) 및 자발성(proactivity), 적응성(adaptiveness), 사회성(sociality)을 바탕으로 다른 에이전트와 전체적 목표(global goal)를 공유할 수 있는 특성을 가지는 컴포넌트 시스템으로 정의 및 활용된다. 이처럼 MAS는 2.1에서 설명했던 SoS와 많은 공통 특성을 공유하는데, MAS와 SoS의 시스템적 특징의 차이를 표 1에서 설명한다.

표 1. 다중 에이전트 시스템(Multi-agent System, MAS)과 시스템-오브-시스템즈(System-of-Systems)의 공통점 및 차이점

(**Table 1.** Commonalities and differences between a Multi-Agent System (MAS) and a System-of-Systems (SoS))

		MAS	SoS
Comm.		Higher-level common (agreed-upon) goal	
		Autonomous and proactive behavior of component systems (i.e., agents & CSs)	
		Integration (Federation, Orchestration) of component systems' capabilities to provide higher-level functionalities and services	
		Openness, Situatedness, and Dynamicity	
		Emergent behaviors from complex interactions	
		Requirements for higher-level quality assurance	
Diff.	**Complexity**	Relatively low	Relatively high
	Managerial independence of component systems	No or low: Originally designed to be affiliated to a higher-level system	Very high: Not originally designed to belong to a higher-level system (and be integrated with other systems
	Engineering approach	Top-down approach	Bottom-up approach (& Top-down approach)

*Comm.: Commonalities, Diff.: Differences

MAS를 대상으로 한 메타모델(meta-model) 기반 방법론은 소프트웨어 공학 분야와 인공지능 분야에서 다양하게 제안되어 왔으며 [13], 가장 대표적인 두 가지 메타모델로는 Gaia 방법론과 [12], O-MaSE 방 법론 [11]이 있으며, 두 방법론 모두 MAS를 분석하거나 설계하기 위한 메타모델 기반 기법을 제안한다.

그 중 가장 대표적인 MAS 분석 및 설계 기법인 Gaia 방법론은 2000년도에 처음으로 객체지향 기법의 아이디어에 기반하여 제안되었으며, 2003년에 저자에 의해 공식적으로 개선된 방법론이 다시 소개되었다. Gaia의 방법론은 MAS를 하나의 조직으로 취급하며, 에이전트 간 상호 관계를 중심으로 분석 단계(analysis phase)와 설계 단계(design phase)에서 필요한 컴포넌트 모델을 정의한다. 수집된 요구사항을 바탕으로 분석 단계에서는 환경 모델, 기초 역할 모델, 기초 상호작용 모델을 명세하며, 설계 단계에서는 앞서 만든 모델을 바탕으로 조직 구조 모델, 역할 모델, 상호작용 모델, 에이전트 모델, 서비스 모델 등을 명세한다. Gaia 방법론은 모델링 기술에 중립적이기 때문에 본 방법론에 따라 모델러가 적절하게 모델링 언어와 기법을 선택할 수 있다는 것이 또 다른 특징이다.

O-MaSE는 조직 기반의 다중 에이전트를 대상으로 한 소프트웨어 공학(Organization-based multi-agent SE) 방법론이며, 기존의MaSE

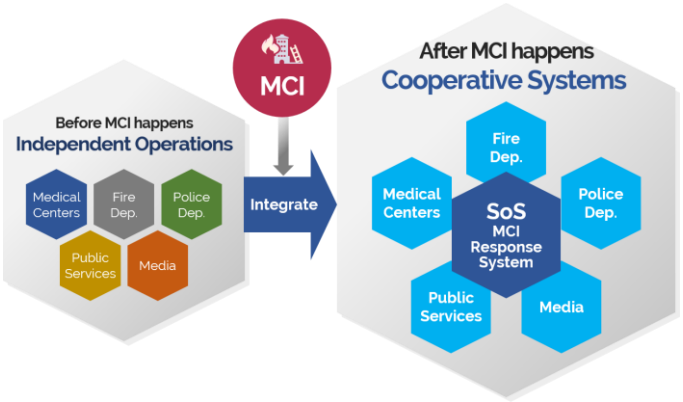

그림 1. 다중손상사고(MCI)와, MCI 대응 SoS (MCIRSoS)
(**Figure 1.** Mass Casualty Incident (MCI) and MCI Response SoS)

방법론을 에이전트의 조직 및 상호작용 관점에서 확장하여 MAS의 표현력을 높이고자 제안된 기법이다. O-MaSE의 핵심 컴포넌트는 Gaia와 마찬가지로 MAS의 에이전트들로 이루어진 조직이며, 각 에이전트가 특정한 목표를 가진 조직에 소속되기 위해 필요한 역량(Capability)과 역할(Role)을 정의할 수 있도록 체계적인 가이드라인과 일부 모델링 기법 및 언어를 제공한다. 뿐만 아니라 O-MaSE는 기존의 MaSE를 확장해 열린 조직을 표현하고 환경과의 상호작용을 고려하여 만들어졌다. 하지만 저자의 이전 연구에 따르면 [16], O-MaSE 기법은 SoS와 달리 에이전트의 역할이 무조건 상위 수준의 목표에 종속적으로 표현되기 때문에, 운영 및 관리 독립성을 갖는 CS의 개별 목표를 메타모델 상에서 분석하지 못한다는 한계점이 존재한다.

3. SoS 사례 시나리오 기반 메타모델 개발

3.1 SoS 사례: 다중손상사고(MCI) 및 MCI 대응 SoS

다중손상사고(Mass Casualty Incident, 이하 MCI)는 일반적으로 사상자 및 피해자의 수와 심각성이 매우 높아 의료 및 구조 자원이 부족한 모든 사고를 의미한다 [10]. MCI가 발생하는 대표적인 원인으로는 대형 사고(예. 화재/폭발, 건물 붕괴, 대형 교통사고)와 자연 재난(예. 지진, 화산, 산불, 홍수) 등이 있다. 이러한 사고는 일반 사고 상황과 다르게 MCI로 선언되는데, 그 이유로는 매우 많은 환자 수(number of casualties), 폭발적인 환자 증가(explosively growing casualties), 환자의 부상 정도의 심각성(severity of casualties) 등이 있다. MCI가 선언되는 상황에는 사고 처리를 담당하는 기존 조직/부처의 의료 및 구조 자원 부족 상태가 발생할 가능성이 높으며, 기존 자원으로는 적절한 치료와 이송을 시기적절하게 지원하기 위한 한계가 있다. 이처럼 MCI 상황은 단시간 내에 통합되지 않은 개별 시스템으로 해결되기 어렵고 자원이 부족하기 때문에, 그림 1과 같이 적절한 시스템들의 협업에 기반해 지속적으로 의료/구조 자원을 제공하기 위한 시스템 구축이 요구된다.

다시 말해, MCI에 효과적으로 대응하기 위해서는 기존 자원 및 시스템의 기능을 넘어서는 상위 수준 요구사항과 목표를 달성해야 하며, 이는 상위 수준의 통합된 시스템 개발이 필요하다는 것을 의미한다. 따라서, MCI 대응 시스템은 현실에 존재하는 SoS의 대표적인 실사례라고 판단할 수 있으며, 우리는MCI 대응 SoS (MCI Response SoS, 이하 MCIRSoS)라고 부른다. MCIRSoS는 사전에 존재하는 관제 시스템, 의료 시스템, 소방 시스템, 경찰 시스템, 교통관리 시스템, 의료 센터 등

표 2. 다중손상사고(MCI) 문헌 분석을 통해 파악한 주요 구성 요소 요약
(**Table 2.** Major component entities identified by the survey of MCI related documents (MCI response plans/strategies))

Component	MCIRSoS entities	Example objects in MCIRSoS
SoS	MCI Response SoS (MCIRSoS)	Building collapse response SoS, Forest fire response SoS
SoS-level target problem	MCI situation that cannot be solved by individual systems' capabilities	Grenfell Tower MCI (2017) Wooshin Golden Suites fire in Busan (2010) Lacrosse Tower fire in Melbourne (2014)
SoS-level infrastructure	Social and physical infrastructure that is established to respond a specific MCI situation	Infrastructure that manages communication channels for CSs and assigns roles to CSs based on SoS-level missions and tasks
SoS-level environment	External entities that interact with MCIRSoS infrastructure and related CSs	Resource, Physical environment, Human factors, Threats (Risks, Hazards)
Constituent System (CS)	Independent unit systems or agencies that performs partial functions of MCI response based on the assigned roles	Command center (Incident Commander (IC))
		Rescue dispatch system
		Fire department / system
		Emergency Medical System (EMS)
		Patient Transport System (PTS)
		Medical center (Hospital)
		Police department / system
		Specialized team(s)
		Public service / system
		Non-governmental system
		Media

을 CS로 통합하여 상위 수준의 임무를 수행한다. MCIRSoS를 SoS로 정의할 수 있는 이유를 Nielsen의 연구에서 정의한 SoS 공학을 위한 8가지 차원(dimension)에 기반해 분석할 수 있다 [7]. 우선, MCIRSoS를 이루는 각 CS는 독립적인 관리자와 이해관계자에 의해 관리 및 운영되는 관리 독립성(managerial independence)과 운영 독립성(operational independence)을 갖는 자율적인 시스템이다. 그리고 MCIRSoS를 이루는 CS는 다양한 이종(heterogeneous/diverse) 시스템들로 구성되어 있고, 대부분의 CS는 지리적으로 분산(geographical distribution)되어 있다. 또한 SoS 수준 목표인 MCI 대응(예. MCI 현장의 95%이상 환자 구조)을 효과적으로 수행 위해서는 서로 기능적 상호의존성(interdependence)을 가질 뿐만 아니라, 정상적이고 신뢰성 있는 시스템 운영을 위해서는 상호운용성(interoperability)을 반드시 가질 수 있어야 한다. 더불어 여러 CS들의 자율적인 행위와 서비스를 수행하는 동안 다양한 변화 발생 요인(SoS 내부 요인, 외부 환경 요인)들이 있는데, 이러한 변화에 대응하고 보다 효과적인 SoS 수준 서비스 제공 및 목표 달성을 위해 짧게는 동적 재구성(dynamic reconfiguration)이 이뤄지고 길게는 SoS의 목표나 서비스가 변화하는 등의 진화적 개발(evolutionary development)을 계획 하에 수행할 수 있어야 한다.

3.2 MCIRSoS 사례의 요소 도출 및 분석

'재난 대응' 혹은 '다중손상사고 대응' 도메인의 온톨로지 분석을 위해서는 조사 대상 사례에 있는 모든 요소/개체(entity)를 철저히 조사하는 것이 중요하다. 우리는 이를 위해 2000년 이후에 발행된 약 30개의

MCI 관련 문헌(부록 A 참고)을 대상으로 조사하였으며, 이들은 MCI 대응 프로시져에 대한 정보를 포함하거나 재난 대응 계획에 대해 정리해 놓은 문헌들이다. 실사례 조사를 위한 문헌의 신뢰성을 얻기 위해 전문 기관(글로벌 단체(WHO 등), 자치주(county), 위원회 및 협회)에서 작성한 문헌만을 골라 선정하였다. 그리고 문헌 조사로부터 도출 가능했던 주요 SoS 요소는 지면이 부족한 관계로 표 2에 요약하여 정리하였으며, MCIRSoS 조사를 바탕으로 온톨로지 기반의 SoS 메타모델을 설계한 구체적인 내용은 4장에서 설명한다. MCIRSoS 사례 조사는 6명의 연구실 학생이 함께 진행하였으며, 도메인 모델링과 요소 도출 및 분석에 약 25일 정도의 시간이 소요되었다.

선정된 문헌들로부터 SoS의 실제 사례인 통합 MCI 대응 시스템, 통합 재난 대응 시스템, 통합 재난 의료 지원 시스템 등을 구성하는 요소(entity) 혹은 객체(object)를 분석할 수 있었으며, 시스템을 이루는 구성 시스템들이 수행하는 임무와 서비스를 도출해 낼 수 있었다. 조사 결과, 대부분의 요소들은 SoS 수준에서 CS들의 전반적인 협업을 관리하고 통제하기 위해 정의한 것들이었으며, 그 예로 SoS 수준의 목표, 요구사항, 서비스, 행동, 통신 채널 등이 파악되었다. 특히 SoS 수준 목표를 달성하기 위해 CS들의 각기 다른 역량에 기반해 역할을 할당하고 SoS 수준 임무를 수행하기 위한 서비스를 제공할 수 있도록 체계적으로 설계된 것을 확인할 수 있었다. MCIRSoS의 가장 큰 특징은 SoS 수준 관리자(SoS-level Manager) 혹은 관리 팀(Managing Team)이 SoS 전반을 직접 지휘하고 관리한다는 것이며, 이로 인해 SoS 수준 관리 및 운영과 CS 수준 서비스 제공이 명확하게 분리되는 것을 알 수 있었다.

4. M2SoS: 시스템 오브 시스템즈 메타모델

4.1 사례 시나리오 분석 결과에 기반한 메타모델 설계 요구사항의 정의

본 장에서는 앞 장에서 조사한 SoS 사례 시나리오를 기반으로 온톨로지 형식의 메타모델을 만들기 위해 필요한 요구사항을 정의한다. MCI 대응 시스템은 일종의 실제 SoS이므로, 현존하는 MCI 대응 시스템이 가진 요소의 속성과 상호작용 등을 분석함으로써 SoS가 가진 특성을 반영할 수 있는 모델 기반 기법 개발의 요구사항을 정리할 수 있다. 우리가 MCIRSoS로부터 분석할 수 있었던 주요 특성은 다음과 같다.

SoS 수준 요소와 CS 수준 요소의 구분

MCIRSoS를 이루는 다양한 시스템들은 일종의 협력을 통해 상위 수준의 서비스를 제공하는데, 그 과정에 있어 SoS 수준 요소와 CS 수준 요소를 구분할 수 있었다. 대표적인 SoS 수준 요소는 상위 목표 달성을 위해 MCIRSoS가 지휘 본부(incident commander)가 수행하는 모든 인프라 관리 및 CS 관리를 위한 것들이었다. MCIRSoS 수준의 이해관계자(정부 및 단체)들은 목표를 달성할 수 있도록 SoS 수준 요구사항을 통해 서비스를 요구하며, 다수의 CS들로부터 얻어지는 협력적 역량에 집중한다. 반면, CS 수준 요소들은 CS들이 특정 역할을 맡아 임무를 수행하는 동안 갖는 운영/관리 독립성과 관련되어 있다. 대부분의 MCIRSoS 사례에서 SoS 수준 관리자 및 지휘 센터는 CS들의 구체적인 기능 수행에 대해 관여하지 않고 그들의 인터페이스(telephone console system 등)를 통한 임무 할당과 정보 교환을 수행한다. 이처럼 CS 수준 요소는 각 CS의 엔지니어에 의해 독립적으로 정의되며 모든 정보가 SoS 수준 관리자에게 제공되지 않을 수 있기 때문에, SoS 분석을 위한 메타모델은 SoS 수준과 CS 수준 요소를 명시적으로 구분할 수 있도록 가이드라인을 제공할 수 있어야 한다.

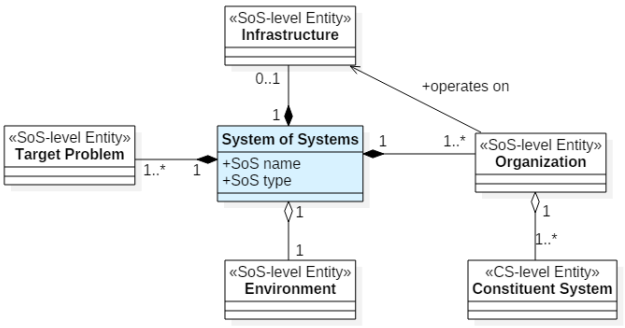

그림 2. M2SoS에서 정의한 SoS 상위 구성 요소
(**Figure 2.** Primary components of an SoS in M2SoS)

SoS 수준 목표, 요구사항, 서비스, 행위 간의 관계 정의

앞서 설명한대로 MCIRSoS의 SoS 수준 요구사항은 SoS 수준 이해관계자와 SoS 수준 관리자가 정의하며, CS들이 협력적으로 서비스를 제공할 수 있도록 요구사항에 따라 인프라를 구축한다. 이러한 요구사항은 SoS 수준 서비스로 구현되어야 하며, 이는 여러 CS들의 서비스를 통합한 SoS 수준의 역량으로부터 도출된다. 이러한 SoS 수준 요소들 사이의 관계는 O-MaSE에서 정의한 목표, 역할, 에이전트, 역량의 관계에 기반하여 정의할 수 있다. 또한 MCIRSoS 사례 문헌에서는 시스템 운영 중 필요한 정책, 전략 등의 운영 요구사항, 기능적/비기능적 개발 요구사항, 데이터와 인터페이스 등을 정형화하기 위한 설계 요구사항(Design Requirements) 등 다양한 형태의 요구사항을 세분화하여 정의하고 있는 것을 확인할 수 있었다. 따라서, 개발하고자 하는 SoS 메타모델은 다양한 요구사항의 종류를 명시적으로 명세할 수 있도록 지원해야 한다.

SoS 운영 시 분석이 필요한 환경 요소 분류 및 정의

대부분의 SoS는 명확한 대상 문제를 해결하기 위해 개발되며, 대상 문제는 넓은 범위의 복잡한 환경(environment)을 포함한다. MCIRSoS에서의 대상 문제는 바로 MCI 상황(혹은 MCI scene)이며, 환자 발생 및 구조를 위해서는 다양한 환경 요소를 고려해야 한다. 문헌 조사를 통해 확인한 MCI의 환경 요소의 예로는 환자 정보와 구조 정보를 포함하는 MCI 상태(MCI status), 금전적/물리적/인적 자원(resource), 지리적 요소나 날씨 등을 포함하는 물리적 환경(physical environment), 인적 요소(human factor), 위협(threat) 등이 있었다. 이러한 환경 요소는 SoS 운영에 있어 매우 높은 불확정성(uncertainty)을 가진 요소들이므로, 분석 및 설계 단계에서의 철저한 파악과 확인이 요구된다. 따라서, SoS 메타모델은 가능한 환경 요소를 명시적으로 분류하여 정의하고, 모델러가 환경 요소에 대해 추가적인 분석을 할 수 있도록 지침을 제공해야 한다.

온톨로지 표현이 가능한 SoS 메타모델 개발

온톨로지는 앞서 설명한 바와 같이 특정 시스템과 도메인 지식을 표

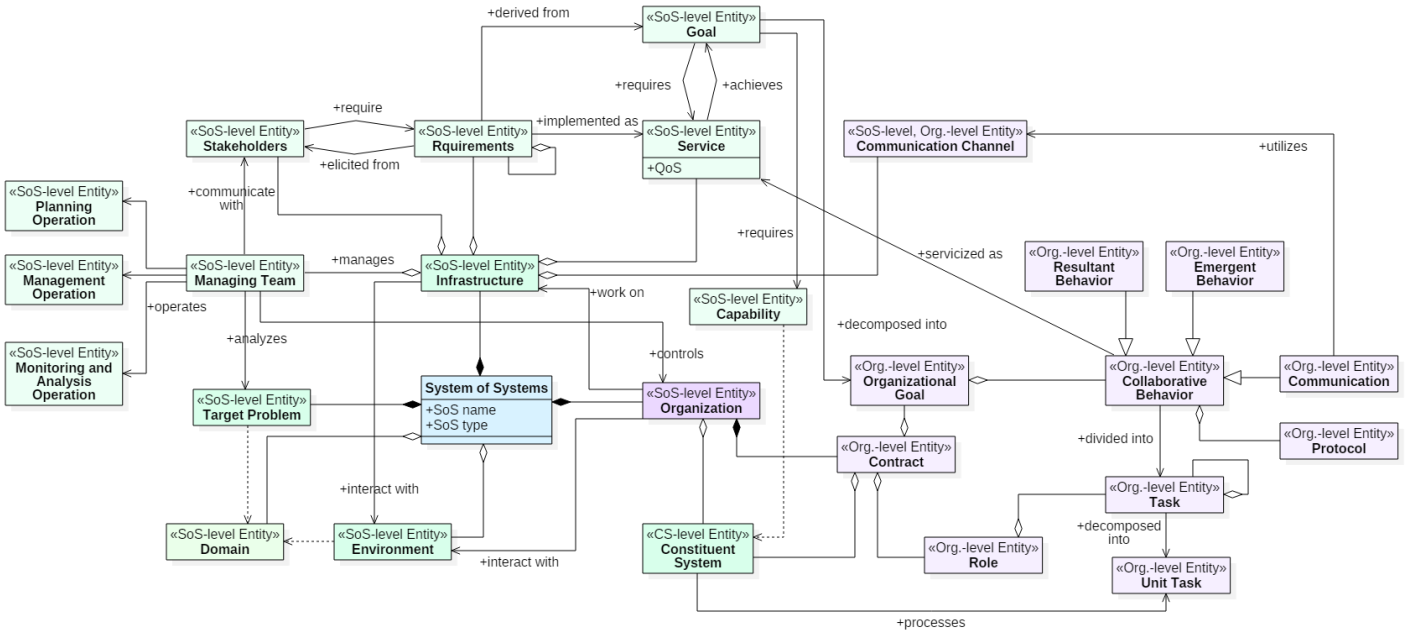

그림 3. M2SoS에서 정의한 SoS 수준 구성 요소 및 조직 수준 구성 요소

(**Figure 3.** Definition of SoS-level entities and organization-level entities in M2SoS)

현하기 위해 클래스(class), 관계(relation), 기능(function) 등을 집합이나 모델 형태로 정의한 것을 가리킨다 [17]. 온토롤지 정의를 통해 상위 수준의 SoS 관리자/공학자는 시스템 전반의 도메인 지식을 획득할 수 있으며, 하위 수준의 CS 관리자/공학자는 상위 SoS가 포함하고 있는 데이터와 도메인 지식에 대한 공통의 이해를 바탕으로 시스템 간 일관성 있는 통신이 가능해진다. 특히, SoS의 경우 대상 도메인에 따라 다양한 모델 기반 기법(아키텍쳐, 모델링 언어, 모델링 프레임워크/도구)을 지원할 수 있어야 하므로, 대상 SoS에 대한 포괄적인 요소를 모델링하기 위해서 온톨로지 기반으로 메타모델을 정의하는 것이 효과적일 수 있다. 앞서 MCIRSoS 사례 분석을 통해 '대규모 재난 대응' 도메인에 대한 지식을 표현할 수 있는 요소들을 분석하였고, 이를 일반화(generalization)한 형태의 SoS 요소들을 도출해 낼 수 있었다. 우리는 이와 같은 방식을 온톨로지 기반의 메타모델 개발이라고 부르며, 도메인-일반적인(domain-general) 메타모델을 개발하기 위한 가장 효과적인 방법이라고 기대한다.

4.2 M2SoS의 수준별 개념 모델 정의

본 논문의 결과물인 SoS 메타모델은 Meta-model for SoS의 약자를 따라 M2SoS라고 이름 지었으며, 앞서 조사한 MAS 메타모델에 기반하여 주요 구성 요소를 SoS에 맞게 변형하고 MCIRSoS에서 도출한 중요한 요소들을 포함하도록 개념 모델(conceptual model) 형태로 정의되었다. 4.2에서는 네 단계에 걸친 SoS 구성 요소를 포함하는 과정을 설명하며, M2SoS의 전체 그림은 부록 B에 첨부하고 구체적인 개념 모델은 웹페이지에 업로드한다[1].

1단계. SoS의 상위 구성 요소 정의

SoS를 분석하기 위한 제일 상위 단계에서는 SoS의 주요 구성 요소를 정의하며, 그림 2와 같이 크게 다섯 가지 요소를 포함한다. SoS는 기존의 단일 시스템으로 해결하기 어려운 상위 수준(SoS-level)의 대상 문제를 다루기 위해 개발되기 때문에 이러한 문제를 명시적으로 표현할 수 있어야 한다. 이를 M2SoS에서는 SoS 수준 대상 문제(SoS-level Target Problem)이라고 정의하며, 이는 SoS 수준 목표를 설정하기 위해 활용된다. SoS가 달성하고자 하는 상위 수준의 목표 및 요구사항은 이러한 대상 문제 분석에 기반해 만들어져야 하기 때문에 명시적인 SoS 주요 구성 요소로 분류되었다.

SoS를 구성하고 행동을 결정하는 가장 중요한 두가지 요소는 SoS 수준 조직(SoS-level Organization)과 조직을 이루는 다수의 CS (Constituent System)들이다. SoS의 조직은 최소 하나의 조직을 포함해야 하며, 다중 계층(multi-hierarchy) 구조를 가질 수 있다. 각 조직은 상위 수준 목표를 달성하기 위해 조직 수준 목표를 설정하고, 협력적 역량을 끌어내기 위해 여러 CS들을 통합한 개체이다. CS는 새로 개발될 시스템과 이미 개발되어 운영중인 레거시 시스템(legacy system)을 모두 포함하는 개념이다. 이어서 여러 CS들을 통합하기 위한 주요 구성 요소로 M2SoS는 SoS 수준 인프라(SoS-level Infrastructure)를 정의한다. SoS 수준 인프라는 다수의 CS를 통해 하나의 상위 수준 시스템을 만들 수 있도록 지원하며, CS 간 상호작용 및 통신을 위한 물리적/가상의 네트워크와 기반 시설(시스템 및 인적 자원)을 제공하는 역할을 한다. 이처럼 인프라를 명시적으로 표현함으로써 개발 대상인 SoS와 환경과의 경계를 명시적으로 정의할 수 있으며, 여러 CS들의 역량으로부터 상위 수준의 목표를 달성할 수 있도록 하는 인프라의 설계와 개발에 활용될 수 있을 것이다.

마지막으로 SoS 수준 인프라 외부의 요소들을 SoS 수준 환경(SoS-level Environment)으로 정의한다. SoS 상에서 수행되는 다양한 시스템 수준의 행위와 서비스들은 환경과 직간접적으로 상호작용하므로, 환경

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

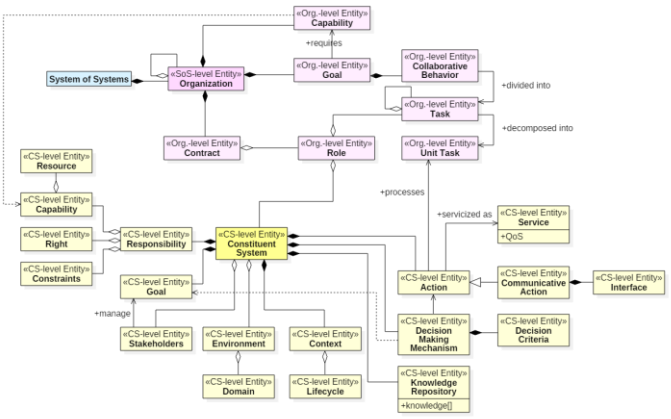

그림 4. M2SoS에서 정의한 CS 수준 구성 요소
(**Figure 4.** CS-level entities of an SoS defined in M2SoS)

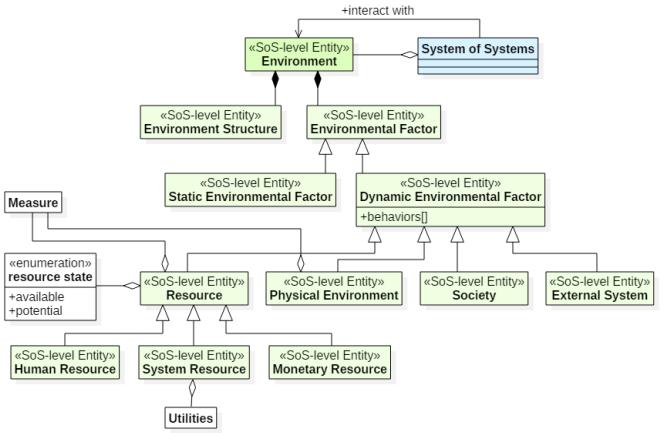

그림 5. M2SoS에서 정의한 환경 요소
(**Figure 5.** Environmental entities of an SoS defined in M2SoS)

과의 상호작용을 명시적으로 표현하기 위해 환경 요소 정의가 필요하다. 즉, SoS는 환경 내에 존재하는 다양한 요소에 대해 효과적으로 대응할 수 있어야 하며, 이를 온톨로지로 표현함으로써 다양한 SoS 구성 컴포넌트들(시스템 및 이해관계자)에게 정보를 제공할 수 있어야 한다. 또한 앞서 요구사항에서 정의되었던 불확실성을 분석하고, 환경 상에서 발생 가능한 위협 등을 파악하는 데에 활용될 수 있다.

2단계. SoS 수준 요소(SoS-level Entities) 및 조직 수준 요소(Organization-level Entities) 정의

상위 구성 요소가 정의된 후, 가장 먼저 정의된 요소들은 SoS 수준 요소들(SoS-level entities)이다. SoS가 목표 지향 시스템이기 때문에 SoS 수준 요소들은 구체적인 대상 문제와 목표에 기반하여 정의된다. MCIRSoS 사례 조사에 따르면, 이러한 SoS 수준 요소는 일반적으로 상위 수준 관리자(커맨드/관제 센터 등)에 의해 분석되고 정의되며, SoS를 이루는 여러 조직과 CS들에게 전달된다. 각 요소들과 요소 사이의 관계를 개념 모델로 정의한 결과는 그림 3과 같다. 요약하자면, SoS 수준 대상 문제를 해결하기 위해 구체적인 SoS 수준 목표(SoS-level Goal)와 요구되는 역량(SoS-level Capability)이 결정된다. 이를 바탕으로 SoS 수준 인프라에서는 목표를 달성하기 위한 구체적인 기능적/비기능적 요구사항(SoS-level Requirements)과 서비스(SoS-level Service)가 명세된다. 대부분의 요구사항과 서비스는 CS의 행동을 결정하거나 SoS 수준 인프라에 구현되며, SoS 수준 관리팀 및 이해관계자(SoS-level Managing Team, Stakeholders)는 이를 관리한다. SoS 타입에 따라 SoS 수준 관리팀과 이해관계자는 존재하거나 존재하지 않을 수 있으며, 이러한 관리팀의 존재 여부가 SoS 수준 요소의 정의 및 관리 수준에 영향을 미칠 수 있다. 사례 조사에서는, MCIRSoS와 같이 중요도(criticality)가 높은 목표를 갖는 SoS는 체계적인 상위 수준 요구사항과 서비스를 관리하기 위해 철저한 관리 체계를 갖는 것을 확인할 수 있었다. 그림 3에서는 대표적인 SoS 수준 임무로써, 계획(Planning Operation), 관리(Managing Operation), 모니터링 및 분석(Monitoring and Analysis Operation)을 포함하고 있다..

SoS에서 상위 수준 목표를 달성하기 위해 실질적으로 미션이나 임무를 수행하는 주체는 바로 다수의 CS들이다. 또한, 여러 CS들이 단일 역량으로는 달성하기 어렵거나 제공하지 못하는 기능과 서비스를 제공해야 하므로 이들은 조직을 이루어 협력적 행동(Collaborative Behavior)을 수행한다. 한 조직 내에서 각 CS들은 특정 역할(Org.-level Role)에 기반하여 조직 수준의 목표를 달성한다. 이 때 조직 내에서 이루어지는 명시적이거나 암시적인 계약(Org.-level Contract)이 조직 구성 및 계약-기반 분석(contract-based analysis)을 위해 선택적으로 활용될 수 있다. 협력적 행동은 CS들이 구체적인 행동(CS-level Action)을 수행함으로써 처리할 수 있어야 하는데, 이 연결 관계를 정의하기 위해 임무(Org.-level Task) 요소가 정의된다. 이는 미션(Mission)과 동일한 개념이며, 대상 시스템의 도메인(국방, 공공 서비스, 로보틱스 등)에 따라 지칭하는 용어가 다를 수 있다. 협력적 행동 중 SoS 내에서 가장 중요할 수 있는 또다른 요소는 바로 커뮤니케이션(Communication)인데, CS 간 상호작용을 표현하기 위해 위 메타모델에서는 커뮤니케이션을 협력적 행동의 특수한 형태로 정의하고 있다. 앞서 설명한 정의에 따라, M2SoS가 두번째 요구사항인 SoS와 조직 수준의 목표, 요구사항, 서비스 명세에 활용될 수 있다.

3단계. CS 수준 요소(CS-level Entities) 정의

SoS 개발을 위해 분석해야 할 하위 계층의 요소들은 그림 4와 같이 CS 수준 요소(CS-level entities)로 구성되며, 이들은 SoS에 통합된 CS들의 구체적인 속성(property) 정보를 포함한다. 각 CS들은 앞서 설명한 대로 운영 독립성과 관리 독립성을 갖기 때문에, CS 수준 요소들은 독립적으로 분석된다. 그러므로 CS 수준 요소의 모델링 및 온톨로지 구축은 대부분 CS 수준 시스템 엔지니어에 의해 이루어지며, 각 CS는 경우에 따라 주어진 역할과 역량에 따른 서비스 정보를 SoS 수준 분석을 위해 제공해야 할 의무를 가질 수 있다. 특히 상향식(bottom-up) 개발이 일반적인 SoS에서는, SoS 수준 관리자와 엔지니어가 수집된 여러 CS들의 정보를 바탕으로 시스템의 구성과 역량을 분석하며, 이를 바탕으로 CS들에게 적절한 역할을 할당하고 인프라를 구축하는 역할을 담당한다. CS를 이루는 여러 구성요소는 아래에서 설명한다.

첫째, 독립적으로 개발 및 운영되는 CS들은 비즈니스 가치(business value)를 갖는 독립적인 목표(CS-level Goal)를 가지고 있으며, 그 목표를 달성하기 위한 서비스들을 수행할 수 있는 역량을 갖는다. M2SoS에서 정의한 CS 수준 목표(Goal), 역량(Capability), 서비스(Service), 역할(Role)은 O-MaSE의 연구에 기반하고 있다 [11]. CS 수준의 역량(CS-level Capability)은 CS 수준 책임(CS-level Responsibility)으로부터 권한(Right) 및 제약 조건(Constraint)과 함께

정의된다. 또한, CS 수준의 역량은 실질적으로 기능을 수행하는 CS 수준 서비스(CS-level Service)로 구현되며, 이를 통해 SoS 조직 내부에서 효과적으로 역량을 수행할 수 있도록 CS 수준 역할(CS-level Role)이 결정된다. 이 외에도 M2SoS는 CS 수준 인터페이스(CS-level Interface), CS 수준 이해관계자 (CS-level Stakeholder), CS 수준 의사결정 메커니즘(CS-level Decision-making Mechanism), CS 수준 도메인(CS-level Domain), CS 수준 환경(CS-level Environment) 등이 정의되었다. 위 요소들은 SoS가 블랙박스(black-box) CS로 구성될 경우, SoS 관리자 입장에서 실질적으로 분석이 가능한 요소가 아닐 수도 있기 때문에 반드시 정의된다고 보장할 수 없다. 그리고 SoS 개발이 상향식으로 이루어질 경우에는, 하위의 CS 수준 요소들을 우선적으로 정의한 이후 상위의 조직과 SoS 수준 요소를 정의하는 순서로 설계와 개발을 수행할 수 있을 것이다.

4단계. SoS 수준 환경(SoS-level Environment) 요소 정의

4.1에서 정의한 메타모델 설계의 요구사항과 같이, 대규모 SoS는 환경과 복잡한 상호작용을 수행하기 때문에 환경 요소의 철저한 분석이 요구된다. 그림 5와 같이 MCIRSoS가 상호작용하는 환경은 여러 환경 요소(Environmental Factor)와 환경 구조(Environmental Structure)로 이루어진다. 그리고 환경 요소는 각 요소의 행동 수행 여부에 따라 정적 요소와 동적 요소로 구분될 수 있다. MCIRSoS 문헌 조사에서는 자원(Resource), 사회(Society), 물리적 환경(Physical Environment), 외부 시스템(External System)의 네 가지 환경 요소 유형을 확인할 수 있었다. 자원 요소는 구체적으로 인적 자원(Human Resource), 시스템 자원(System Resource), 금전적 자원(Monetary Resource)으로 구분이 가능하며, 이러한 자원들은 사용 가능 상태와 잠재적 사용 가능 상태의 두 가지 자원 상태(Resource State)를 가질 수 있다. 두번째 환경 유형인 사회 요소는 SoS의 운영에 영향을 미치는 외부의 모든 개인, 조직, 기관 등을 포함하며, 이러한 요소들을 분석함으로써 SoS 수준 관리자는 SoS 운영 계획과 정책 등을 수립할 수 있다. 세번째 환경 유형인 물리적 환경은 SoS 운영에 영향을 미치는 외부의 모든 물리적 요소를 정의하기 위한 요소이며, 특정한 타입(element type)과 단위(element unit)를 포함한다. MCIRSoS의 예제에서는 날씨, 온도, 바람, 교통량 등이 이에 해당하며 이러한 정보를 SoS 운영에 활용하기 위해서는 명시적인 측정(measure)이 이뤄질 수 있는 모델링이 수행되어야 한다.

4.3 MAS 메타모델과 M2SoS의 비교 분석

본 절에서는 4.1에서 정의한 요구사항을 바탕으로 4.2에서 단계별로 정의한 M2SoS를 MAS의 두 가지 메타모델과 비교하고자 한다. 표 3은 세 가지 메타모델의 표현력을 MCIRSoS에서 도출한 SoS 관련 요소를 기준으로 분석하였으며, MAS 메타모델에서 명시적으로 표현되지 않았던 많은 요소들이 M2SoS에서는 표현 가능하다는 점을 확인할 수 있다. MAS의 메타모델인 Gaia와 O-MaSE는 여러 컴포넌트 에이전트로 이루어진 조직을 중심으로 하여 역량, 역할, 기능 수행에 초점을 맞추어 표현한 반면, SoS는 이러한 아이디어와 기본 구성을 바탕으로 보다 SoS에 적합한 표현력을 얻기 위해 추가적인 컴포넌트를 정의했다. 특히 MAS 메타모델은 상위 수준 요소와 하위 수준 요소를 구분하여 표현하고 있지 않기 때문에 하위 수준 시스템의 정보가 부족하거나 블랙박스(black-box) 형태의 시스템일 경우 설계하기 어려운 한계점이

표 3. MAS 메타모델(Gaia, O-MaSE)와 M2SoS의 표현력 비교 분석
(**Table 3.** Comparison between M2SoS and MAS meta-models)

Entities	MAS meta-models		M2SoS
	Gaia [12]	**O-MaSE** [11]	
Component system-level (Agent-level / CS-level) entities			
Component System	Agent (Organizational Agent)		Constituent System (CS)
Role	Role AgentType	Role	CS-level Role
Goal	×	×	CS-level Goal
Service	Service Activity Action Responsibility Permission	Action	CS-level Service
Capability	×	Capability	CS-level Capability
Interface	×	×	CS-level Interface
Decision Making	Permission	Plan, Action	CS-level Decision Making Mechanism
Stakeholders	×	×	CS-level Stakeholder
Domain	×	Domain Model	CS-level Domain
Lifecycle	×	×	CS-level Lifecycle
Environment	Environment	Environment Object, Property	CS-level Environment
Higher-level system (MAS / SoS) entities			
Higher-level System	MAS		SoS, Organization
Organization	Organizational Rule, Structure	Organization	Organization
Infrastructure	×	×	SoS-level Infrastructure
Service / Behavior	×	×	SoS-level Service
Communication	Communication, Protocol	Protocol	SoS-level Communication
System Manager	×	(Actor)	SoS-level Managing Team
(Common) Goal	×	Goal	SoS-level Goal, Target Problem, Requirements
Policy	×	Policy	SoS-level Policy
Domain	×	×	SoS-level Domain
Environment	×	×	SoS-level Environment
Management Mechanisms (Operations)	×	×	SoS-level Managing Team's Operations,
Emergent Risks / Threats	×	×	SoS-level Emergent Behavior

존재했다. 반면, 상위의 SoS 수준 요소와 CS 수준 요소를 구분한 M2SoS는 각각 SoS 수준 공학자(관리자, 설계자 등)가 수행해야 하는 분석의 대상과 CS 수준 공학자가 수행해야 하는 분석 대상 요소를 명확하게 구분함으로써 SoS의 CS가 갖는 운영적/관리적 독립성을 나타낼 수 있게 되었다.

5. M2SoS의 한계점 및 확장 방안

5.1 M2SoS 활용을 위한 관점 정의 및 불확정성 표현

메타모델링의 가장 큰 목적은 바로 '모델링을 위한 가이드라인 제공'이며, M2SoS 역시 SoS의 효과적인 분석과 설계를 위한 가이드라인을

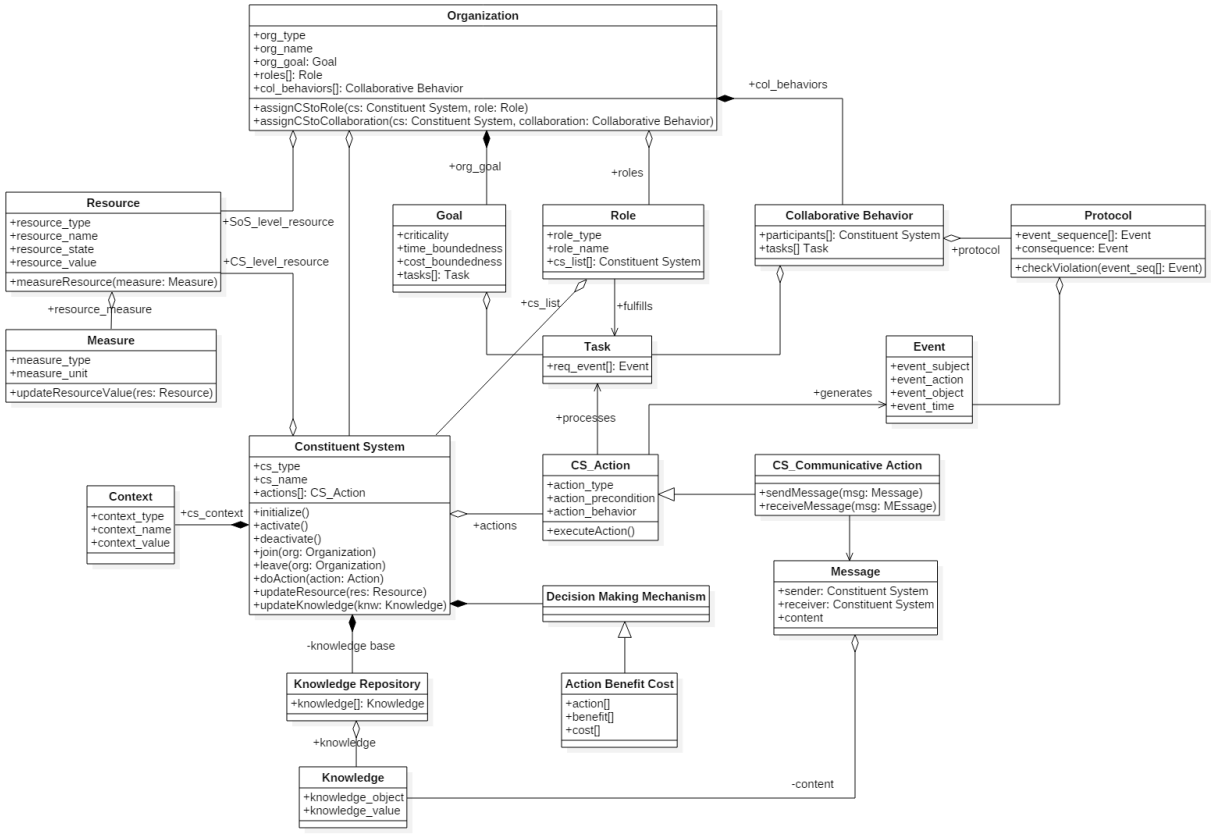

그림 6. M2SoS 기반의 시뮬레이션 모델 설계

(**Figure 6.** Design of simulation model based on M2SoS)

제공하는 목적으로 활용할 수 있다. 하지만, 현재의 M2SoS는 SoS를 사례 시나리오를 바탕으로 분석하여 온톨로지 기반의 메타모델로 개발되었기 때문에 구체적인 모델링 방법(여기서의 모델링 방법은 모델링 언어, 모델링 프로세스 등을 의미함)을 함께 제시하지 못하는 한계가 존재한다. 그러나 M2SoS는 도메인 일반적이고(domain-general) 기술 중립적인(technique-neutral) 메타모델이기 때문에, 공학자와 다양한 이해관계자가 분석, 설계 및 개발의 대상이 되는 SoS를 포괄적인 범위에서 이해하기 위한 도구로 활용될 수 있다. 또한, 여러 다양한 지식과 정보를 포함해야 하는 온톨로지 구축을 위해서 본 메타모델이 체계적인 방법을 통해 도출한 요소들을 활용할 수 있을 것이다.

M2SoS를 보다 구체적인 목적에 따라 활용하기 위해서는 SoS 분석 관점(viewpoint)을 확장하는 것이 필요하다. 본 논문의 저자는 SoS의 아키텍쳐 연구를 수행한 AMADEOS 프로젝트의 문헌을 참고하여 SoS 분석을 위한 9가지 관점을 정의할 수 있었다 [18]: 현재 버전의 M2SoS의 목적과 동일하게 SoS를 이해하고 포괄적인 그림을 그리기 위한 건설적 관점(construction view), CS들의 협력적 행위를 통해 제공할 수 있는 SoS 수준 서비스 관점(service view), 개발된 SoS 모델을 시뮬레이션에 활용하기 위한 시뮬레이션 관점(simulation view), 검증에 활용하기 위한 검증 및 확인 관점(verification & validation view), SoS 수준 관리자가 CS 및 자원 관리, 계획, 동적 재구성, 위험 관리, 모니터링 등을 수행하기 위한 관리적 관점(governance view), CS들의 행위를 통해 상위 수준의 프로세스를 정의하는 프로세스 관점(process view), 데이터 및 자원 관점(data/resource view), SoS의 진화적 개발을 위한 진화 관점(evolution view), 안전 필수 시스템(safety critical system) 등을 위한 시간적 관점(time view)이다. 그 중에서 저자들은 시뮬레이션 관점과 검증 관점을 중심으로 SoS의 행동을 관찰하고 분석할 수 있도록 검증을 위한 M2SoS를 개발하는 연구를 진행중에 있다 (5.2 참조).

M2SoS를 실질적인 SoS의 분석과 모델링에 사용하기 위해서는 SoS에 내재되어 있는 다양한 불확정성(uncertainty)을 표현할 수 있어야 한다. 현재 개발된 M2SoS는 SoS 수준과 CS 수준의 다양한 요소들을 정의하고 있지만, 각 요소들이 갖는 불확정성에 대한 표현력이 부족하다. 객체관리그룹(OMG)이 소프트웨어 시스템이 갖는 불확정성을 표현하고 테스팅하기 위해 개발한 불확정성 모델(Uncertainty Model) [19] 등을 활용하면, M2SoS에서 정의한 요소들이 갖는 불확정성을 분석 단계에서 도출해 낼 수 있을 것으로 기대한다.

5.2 SoS 시뮬레이션을 위한 M2SoS 활용

CS들의 협력적/경쟁적 행위로 인해 상위 수준에서 발생하는 창발적 행위(emergent behavior)를 최대한 예측하기 위해서는 SoS의 시뮬레이션 기법이 요구된다 [7, 18]. 또한, 시뮬레이션을 통해 추출된 실행 데이터는 통계적 검증과 테스팅에 활용될 수 있다 [20]. M2SoS는 시뮬레이션을 위한 SoS 모델과 시뮬레이션 데이터를 정의하기 위해 활용될 수 있다. 예를 들어, MCI 상황을 시뮬레이션하기 위해서 M2SoS에서 정의한 요소들을 기반으로 SoS 수준 인프라, SoS 수준 조직, SoS 수준 환경 및 다수의 CS 모델을 생성해 낼 수 있을 것이다. 또한 시뮬레이션에서 확인하고자 하는 검증 속성(verification property)은 SoS 수준 대상 문제(SoS-level Target Problem)와 SoS 수준 목표(SoS-level Goal)로부터 도출이 가능하다.

SoS와 같은 목표 지향(goal-oriented) 시스템들은 특히 구체적인 목표 달성 가능성을 분석할 수 있어야 한다. SoS는 큰 시스템을 이루고 있는 여러 조직 내 CS의 협업을 통해 상위 수준의 역량(capability)을 가지며, 이를 통해 SoS 수준의 서비스를 제공할 수 있다. 이러한 점은 기존의 에이전트 기반 시뮬레이션(agent-based simulation) 연구와 유사하다고 할 수 있지만 [13], 높은 운영/관리 독립성을 갖는 SoS의 여러 CS의 협업을 분석한다는 점에서 차이점을 가진다. M2SoS에 기반하여 독립적인 CS와 조직을 시뮬레이션 할 수 있도록 그림 6과 같은 SoS 시뮬레이션 모델 설계가 가능하다. 모델의 주요 구성 요소는 조직이며, SoS 수준 목표를 달성하기 위한 협력 행동(Collaborative Behavior)이 정의된다. 협력 행동은 구체적인 업무(Task)의 집합으로 정의될 수 있으며, 각 업무는 CS의 행동(Action)에 의해 처리된다. 이 외에도 CS의 독립성을 표현하기 위한 지식 저장소(Knowledge Repository) 및 의사 결정 메커니즘(Decision Making Mechanism)을 가지며, 서로 상호작용하기 위한 메시지 기반의 커뮤니케이션을 지원한다. 또한 본 모델은 시뮬레이션 기반의 분석을 지원하기 위해 설계되었기 때문에, CS들의 모든 행동은 구체적인 이벤트(Event)를 발생시키며, 생성된 이벤트의 리스트 혹은 로그를 분석함으로써 SoS 목표 달성 여부를 확인할 수 있다. 이처럼, 현재 저자는 M2SoS를 SoS의 시뮬레이션 및 통계적 검증을 위한 입력 모델 생성을 위해 사용할 수 있도록 연구를 진행중이다.

6. 결론 및 향후 연구

대표적인 대규모 복잡 시스템인 시스템 오브 시스템즈(System-of-Systems, SoS)의 개발을 위해서는 개발 대상 SoS의 특성을 이해하기 위한 체계적인 분석 기법이 요구되는데, 본 연구는 이러한 문제를 해결하기 위한 방법으로 메타모델 기반 접근 방식에 초점을 맞췄다. 본 논문에서 제안한 SoS 메타모델인 M2SoS (Meta-model for SoS)는 SoS와 가장 유사한 시스템 도메인인 다중 에이전트 시스템(Multi-agent System, MAS) 분야의 메타모델 분석에 기반하여 기본 뼈대를 구성하고, 재난 대응 SoS의 대표적인 사례 시나리오인 다중 손상사고 대응 SoS (Mass Casualty Incident Response SoS)의 문헌 분석을 통해 SoS 특징적인 요소를 반영하기 위한 메타모델 설계 요구사항을 정의하였다. 이를 바탕으로 개발된 온톨로지 기반의 M2SoS를 대표적인 두 가지 MAS 메타모델인 Gaia 및 O-MaSE과 비교함으로써 SoS 사례 시나리오에서 분석한 다양한 요소들을 포괄할 수 있는지에 대한 표현력을 분석할 수 있었다. MAS 메타모델에 기반하여 확장한 M2SoS인 만큼, SoS 메타모델 설계의 요구사항을 포괄적이고 충실히 반영할 수 있다는 것을 확인할 수 있었다. 5.2에서 설명한 바와 같이 M2SoS는 향후 SoS의 모델 기반 시뮬레이션 및 통계적 검증에 우선적으로 활용할 계획이다. 이를 위해 시뮬레이션 기반 분석을 위한 테스팅 및 검증 메타모델인 M2SoS for SIMVA (SIMulation-based Verification and Analysis)를 현재 개발중에 있으며, 이는 최종적으로 SoS의 정량적 검증을 수행할 수 있는 시뮬레이션 도구의 입력으로 사용될 것이다. 또한, 지속적인 SoS 사례 시나리오 분석을 통해 M2SoS를 개선하고 메타모델에 정의된 각 요소에 대한 구체적인 모델링 기법을 개발할 예정이다.

참 고 문 헌

[1] M. Hause, "The Unified Profile for DoDAF/MODAF (UPDM) Enabling Systems of Systems on Many Levels," Systems Conference, San Diego, CA, USA, 2010.

[2] M. L. Butterfield, and J. S. Pearlman, "A System-of-Systems Engineering GEOSS: Architectural Approach," IEEE Systems Journal, vol. 2, issue 3, 2008.

[3] F. Fiedrich, and P. Burghardt, "Agent-based Systems for Disaster Management," Communications of the ACM - Emergency Response Information Systems: Emerging Trends and Technologies," vol. 50, issue 3, 2007.

[4] M. Jamshidi, "System of Systems Engineering - New Challenges for the 21st Century," IEEE Aerospace and Electronic Systems Magazine, vol. 23, issue 5, 2008.

[5] J. Boardman, and B. Sauser, "System of Systems - the Meaning of Of," IEEE/SMC International Conference on System of Systems Engineering (SoSE), 2006.

[6] W. C. Baldwin, and B. Sauser, "Modeling the Characteristics of System of Systems," IEEE International Conference on System of Systems Engineering 2009 (SoSE 2009).

[7] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions," Journal of ACM Computing Surveys (CSUR), vol. 48, issue 2, 2015.

[8] B. Zhou, A. Dvoryanchikova, A. Lobov, and J. L. M. Lastra, "Modeling System of Systems: A Generic Method based on System Characteristics and Interface," 9th IEEE International Conference on Industrial Informatics (INDIN), 2011.

[9] S. Karnouskos, and A. W. Colombo, "Architecting the Next Generation of Service-based SCADA/DCS System of Systems," 37th International Conference on IEEE Industrial Electronics Society (IECON), 2011.

[10] J. Morganwalp, and A. P. Sage, "A System of Systems Focused Enterprise Architecture Framework and an Associated Architecture Development Process," Journal of Information Knowledge Systems Management, vol. 3, no. 2-4, pp. 87-105, 2002/2003.

[11] S. A. DeLoach, J. C. Garcia-Ojeda, "O-MaSE: A Customizable Approach to Designing and Building Complex, Adaptive Multi-agent Systems," International Journal of Agent-Oriented Software Engineering, vol. 4, issue 3, 2010.

[12] F. Zambonelli, N. R. Jennings, M. Wooldridge, "Developing Multiagent Systems: The Gaia Methodology," Journal of ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 12, issue 3, 2003.

[13] C. Bernon, M. Cossentino, M. P. Gleizes, P. Turci, and F. Zambonelli, "A Study of Some Multi-agent Meta-models," Agent-Oriented Software Engineering V - AOSE 2004, Revised Selected

Papers, Vol. 3382. pp. 62-77, 2004.

[14] M. W. Maier, "Architecting Principles for Systems-of-Systems," Systems Engineering, vol. 1, issue 4, pp. 267-284, 1998.

[15] C. Bernon, M. Cossentino, and J. Pavon, "Agent-Oriented Software Engineering," 2005 Cambridge University Press, vol. 20, issue 2, pp. 99-116, 2005.

[16] Youlim Jung, Young-Min Baek, Ilchul Yoon, and Doo-Hwan Bae, "Analysis of Multi Agent Systems Meta Modeling Approaches for System-of-Systems Modeling," 2016 Journals of Korean Institute of Information Scientists and Engineers (ISSN 2466-0825), pp. 398-400, 2016.

[17] A. D. Nicola, M. Missikoff, and R. Navigli, "A Software Engineering Approach to Ontology Building," Journal of Information Systems, vol. 34, issue 2, April 2009, pp. 258-275.

[18] M. Mori, A. Ceccarelli, P. Lollini, B. Fromel, F. Brancati, and A. Bondavalli, "Systems-of-Systems Modeling using a Comprehensive Viewpoint-based SysML Profile," Journal of Software: Evolution and Process, Special Issue - HASE 2016, 2017.

[19] M. Zhang, B. Selic, S. Ali, T. Yue, O. Okariz, and R. Norgren, "Understanding Uncertainty in Cyber-Physical Systems: A Conceptual Model," European Conference on Modelling Foundations and Applications (ECMFA), 2016, pp. 247-264.

[20] W. Yun, D. Shin, D. H. Bae, "Mutation Analysis for System of Systems Policy Testing," 2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (JSOS), 2017.

백 영 민 (Young-Min Baek)

2014년 건국대학교 인터넷미디어공학부 학사. 2016년 한국과학기술원(KAIST) 전산학과 석사. 2016년 ~ 현재 한국과학기술원(KAIST) 전산학부 박사과정. 관심 분야는 모델 기반 소프트웨어 테스팅, 시스템 오브 시스템즈 공학, 소프트웨어 시스템 모델링 및 아키텍쳐.

박 수 민 (Sumin Park)

2017년 건국대학교 컴퓨터공학과 학사. 2017년 ~ 현재 한국과학기술원(KAIST) 전산학부 석사 과정. 관심 분야는 모델 기반 소프트웨어 공학, 시스템 오브 시스템즈 공학, 카오스 엔지니어링.

신 용 준 (Yong-Jun Shin)

2017년 한동대학교 전산전자공학부 학사. 2017년 ~ 현재 한국과학기술원(KAIST) 전산학부 석박통합과정. 관심분야는 시스템 오브 시스템즈 공학, 소프트웨어 모델링, 자가적응 시스템.

배 두 환 (Doo-Hwan Bae)

1980년 서울대학교 조선공학 학사. 1987년 Univ. Of Wisconsin-Milwaukee 전산학과 석사. 1992년 Univ. Of Florida 전산학과 박사. 1995년 ~ 현재 한국과학기술원(KAIST) 전산학부 교수. 관심분야는 소프트웨어 프로세스, 객체지향 프로그래밍, 컴포넌트 기반 프로그래밍, 임베디드 소프트웨어 설계, 관점 지향 프로그래밍, 시스템 오브 시스템즈 소프트웨어 공학.

부 록

부록 A. 다중손상사고(Mass Casualty Incident, MCI) 조사 문헌

번호	문헌 이름	문헌 작성 기관	연도
1	Rogers Fire Department Standard Operating Procedures - Policy Title: Mass Casualty Incidents	Rogers Arkansas	2013
2	Emergency Medical Services Policies and Procedures - Policy Title: Multi Casualty Incident Response	County of Ventura Health Care Agency	2014
3	Mass Casualty Incident	Shao Foong Chong	2013
4	Planning & Triage	Peds (Pediatrics in Disasters)	
5	Florida Field Operations Guide - Mass Casualty	Florida State	2012
6	Dane County Regional Mass Casualty Incident Response Plan	Metropolitan Medical Response System (MMRS)	2006
7	MCI PLAN – Franklin County Multiple Casualty Incident Response Plan	Franklin County EMS, MCI Planning Committee	2009
8	The Massachusetts Emergency Medical Services (EMS) Mass Casualty Incident (MCI) Plan	The Massachusetts Department of Public Health	2016
9	Integrated Explosive Event and Massive Casualty Event	Greater New York Hospital Association	-
10	Emergency Management Plan for Mass Casualty Incidents	-	-
11	Mass Casualty Incident Plan - Adopted by the Marion County Fire Defense Board	Marion County Fire Defense Board	2001
12	Mass Casualty Response: NHS Tactical Command Framework	NHS England Cumbria Northumberland Tyne and Wear Area Team	2014
13	Appendix N - Detailed Sample Scenarios	-	-
14	Multi-Casualty Incident (MCI) Response Plan	Santa Barbara County – Emergency Medical Services Agency	2013
15	Mass Casualty Management Systems - Strategies and Guidelines for Building Health Sector Capacity	World Health Organization (WHO)	2007
16	National Incident Management System	Homeland Security	2008
17	MCI Triage Drill	Alabama Fire College Workplace Safety Training Program	-
18	Utah Department of Health Bureau of EMS and Preparedness Emergency Operations Plan	Utah Department of Health	2011
19	Recommendations of the Standing Committee on Multiple Casualty Incident Planning and Evaluation	The Commonwealth of Massachusetts	2004
20	Emergency Medical Services - Multi-Casualty Incident Plan	Contra Costa County	2012
21	Multiple Casualty Incident (MCI) Response Plan	Monterey County EMS	2014
22	MCI Operational Policy - Clear Creek County Emergency Services	Clear Creek Fire Authority/ Clear Creek EMS	2011
23	Multi-Casualty Incident Policy	San Francisco Emergency Medical Services Agency	2005
24	Mass Casualties Incident Plan for NHS Scotland	NHS Scotland Resilience	2015
25	Multi-Casualty Incident Policy	NPS EMS Field Manual	2016
26	ODEMSA Mass Casualty Incident (MCI) Plan	Heidi M. Hooker, ODEMSA	2014
27	SUFFOLK Mass Casualty Plan	Suffolk NHRP on behalf of Suffolk Resilience Forum	2015
28	Operational Templates and Guidelines for EMS Mass Incident Deployment	FEMA	2012
29	Hospital Medical Surge Planning for Mass Casualty Incidents	Florida Department of Health	-
30	Mass Casualty Incident (MCI): An Overview	Jim Thomas, Captain	-
31	2017 소방전술 I: 신임교육과정(화재3)	중앙소방학교(NFSA)	2017

부록 B. M2SoS: 온톨로지 개발을 위한 SoS 메타모델(Meta-model for System-of-Systems)