석 사 학 위 논 문
Master's Thesis

ABC$^+$: 상호작용을 고려한 의사결정에 대한 시스템 오브 시스템즈 시뮬레이션 모델링 기법

ABC$^+$: A Simulation Modeling Approach for Decision-making Incorporating Interaction in System of Systems

2018

진 민 규 (陳 旻 奎 Jin, Mingyu)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

석 사 학 위 논 문

ABC+: 상호작용을 고려한 의사결정에 대한 시스템 오브 시스템즈 시뮬레이션 모델링 기법

2018

진 민 규

한 국 과 학 기 술 원

전산학부

ABC+: 상호작용을 고려한 의사결정에 대한 시스템 오브 시스템즈 시뮬레이션 모델링 기법

진 민 규

위 논문은 한국과학기술원 석사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2017년 12월 18일

심사위원장 배 두 환 (인)

심 사 위 원 고 인 영 (인)

심 사 위 원 유 신 (인)

ABC$^+$: A Simulation Modeling Approach for Decision-making Incorporating Interaction in System of Systems

Mingyu Jin

Advisor: Doo-Hwan Bae

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Daejeon, Korea
December 18, 2017

Approved by

Doo-Hwan Bae
Professor of School of Computing

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

MCS
20163628

진민규. ABC$^+$: 상호작용을 고려한 의사결정에 대한 시스템 오브 시스템즈 시뮬레이션 모델링 기법. 전산학부 . 2018년. 38+iii 쪽. 지도교수: 배두환. (영문 논문)

Mingyu Jin. ABC$^+$: A Simulation Modeling Approach for Decision-making Incorporating Interaction in System of Systems. School of Computing . 2018. 38+iii pages. Advisor: Doo-Hwan Bae. (Text in English)

초 록

시스템 오브 시스템즈 (SoS)는 자율적이고 독립적인 구성 시스템 (CS)으로 이루어진 대규모 복잡 시스템이며, 단일 CS가 만족할 수 없는 SoS 수준의 요구사항을 다룬다. SoS 엔지니어는 CS의 자율성과 독립성을 고려하여 SoS를 모델링하고 시뮬레이션함으로써 SoS 수준의 행위를 분석할 수 있다. 행동-이익-비용 (ABC) 모델링은 CS의 자율성과 SoS 매니저의 권한을 고려하여 관찰되는 행동, 이익과 비용에 초점을 맞추어 CS의 의사결정 과정을 모델링하였다. 하지만, CS가 고정된 비용편익 표를 따르며 서로 상호작용하지 않는 모델은 시뮬레이션 결과가 실제 SoS 수행을 모사하지 못하는 문제를 발생시킨다. 본 논문에서는 ABC+ 모델링을 제안하며, 상태 기반의 의사결정과 메시지 기반 상호작용 모델을 소개한다. 이를 통해 ABC 모델링이 갖는 문제를 해결하고, 다양한 형태의 SoS 모델을 보다 쉽게 만들 수 있도록 돕는다. 또한, 다중 손상사고 사례 시나리오를 통해 제안된 기법의 효과성과 SoS 분석에의 적용 방안을 정량적으로 보인다.

핵 심 낱 말 시스템 오브 시스템즈, 에이전트 기반 모델링, 상태 기반 의사결정, 상호작용, 시뮬레이션

Abstract

The system of systems (SoS) is a large-scale and complex system composed of autonomous and independent constituent systems (CSs). It deals with complex requirements as SoS-level goals that are not able to be satisfied by a single CS. To analyze SoS-level behaviors and SoS-level goal achievements considering CSs' autonomy and independence, SoS-level engineers and managers need to model and simulate an SoS properly. Action-Benefit-Cost (ABC) modeling [20] provides an effective and efficient way to model and simulate the autonomous and independent behaviors of CSs by focusing on their external actions, benefits, and costs. However, ABC modeling does not support internal state of a CS so that the CS cannot update its cost-benefit table and makes problem in their decision-making. In addition, ABC modeling does not support communications between CSs so that the CSs cannot share their information during the simulation. SoS engineers also have difficulty in reflecting all possible interaction into models in advance of simulation. In this paper, we propose ABC$^+$ modeling, which supports the three-phased stateful decision-making model and the message-based interaction model. The case study shows that ABC$^+$ modeling improves the expressiveness of SoS models and lets SoS engineers simulate and analyze the SoS-level behaviors by different interaction patterns, which cannot be even represented in the existing ABC modeling. The simulation execution time is examined about both ABC and ABC$^+$ modeling, including its relationship with message counts varying from different interaction models.

Keywords System of Systems, agent-based modeling, decision-making, interaction, simulation

Contents

List of Tables

List of Figures

Chapter 1. Introduction

The increasing scale and complexity of systems have reached a point that imposes qualitatively new demands on the existing system technology. The emerging system is characterized by distributed, decentralized, and networked combinations of autonomous sub-systems under large-scale and complex environments. This new "system" is a general concept of "System of systems" (SoS) [15]. Complex requirements and goals that are not able to be satisfied by a single system call for the concept of SoS. The combination and integration of single systems is expected to deal with those requirements called SoS-level goals. The term has arisen in various domains such as social infrastructures, smart grids, military services, and air traffic managements [31].

An SoS is composed of independent sub-systems, which are called as Constituent Systems (CSs). A CS provides its own services with independent ownership and management rules. CS has unique characteristics such as (1) managerial independence, i.e., each system is managed in most parts for its own goals rather than the goals of SoS, (2) operational independence, i.e., each system operates independently to achieve its own goals by itself [28] and (3) interoperability, i.e., each system is able to exchange information with another through agreed interfaces, protocols and standards [23], which helps to achieve its own goals or/and SoS-level goals. Monarch and Wessel [30] restates managerial and operational independence as managerial and operational autonomy. SoS has a different and new capability that is not able to be substituted by any single one of the constituent systems [16].

SoS engineers and managers, who integrate systems into an SoS, have the demand of analyzing the behaviors of SoS and verifying the satisfaction of properties. One of the most frequently used forms of analysis is execution [31]. Model simulation is a typical solution for systems whose real execution requires high cost but that can be modeled in some forms. Simulation provides an approximation of the behavior of real execution of the target system. Modeling and simulation provides SoS managers with a great tool to check whether the SoS-level goals of interest are satisfiable under certain conditions given as parameters. The modeling and simulation methods should have the ability to represent SoS characteristics and the power of expressiveness that is enough to approximate real execution results. Baldwin et al. [2] compare agent-based modeling paradigm (ABM) and event-based modeling paradigm (EBM) in the appropriacy to model SoS characteristics, and ABM is appropriate to model those characteristics. Individual agents are modeled considering independence and interoperability, and interactions between them are expected to produce emergent behaviors in simulating them all together [12].

Kim et al. [20] proposed Action-Benefit-Cost (ABC) modeling, where each CS is modeled with the economical concepts of benefit and cost. It states the lack of accessibility of SoS engineers to detailed information of CS's decision-making mechanism. Due to the SoS characteristics where CS's has managerial and operational autonomy, SoS managers have no authority to access the full information of CS's decision-making but some that are externally observable, i.e., behaviors or actions. Every factor

that affects decisions is interpreted and mapped to benefit and cost values as the minimum information to simulate CSs' behaviors. In simulation, CS models perform decision-making every moment according to the cost-benefit table given by domain experts. The abstraction on decision-making mechanism brings the advantage in securing autonomy by information hiding and reducing the simulation cost dramatically.

Even though ABC modeling has shown its usefulness, it has limitations in expressiveness of (1) one's autonomous decision-making mechanism and (2) of interactions between SoS entities such as CS or SoS manager. It is essential for an SoS model to include decision-making model that is able to represent entities' autonomy. Constituent systems are assumed independent in their management and operation. We say a system is autonomous when it has ability to access its situation and make own decisions based on gathered information according to its own goals or SoS-level goals. However, ABC model has no mechanism for one entity to assess the situation and maintain states, but simply reacts to the situation according to the fixed cost-benefit table given at the beginning of the simulation. It produces a series of unrealistic decisions and fails to approximate real execution of target SoS.

Interactions between SoS entities is important because collaboration is expected as the key means to achieve SoS-level goals, orchestrating CSs who are heterogeneous and independent with individual goals. Modeling interactions between SoS entities precisely is the requirement for SoS simulation model in order to approximate real execution of target SoS. However, ABC model does not include interaction model that enables information exchange between entities during the simulation. It assumes that all information is shared by all, which makes every information modeled in SoS environment and decision-making models reflect possible effects in advance of simulation. However, it is difficult or probably impossible to pre-evaluate every effect of possible interactions. In addition, we should make the whole new model from the scratch when the interaction pattern is changed even though CSs themselves' decision-making mechanisms are still identical to the existing model. It is a quite big burden for SoS engineers who want to analyze the impact of interactions between SoS entities.

In order to address the limitations of the existing ABC modeling, we propose ABC$^+$ modeling with new supporting models. We (1) introduce the concept of state and the phased decision-making mechanism, which helps autonomous decision-making of individual entity. State is defined as a memory of each entity in which one stores useful information such as history, environmental situation, social agreement, etc. The information is later evaluated in cost-benefit analysis to adjust the benefit and cost values in terms of the preference of actions according to its own and/or SoS-level goals. We also (2) introduce the interaction model to give interoperability to the entities. The information exchange or the request and reply on certain actions helps each entity to achieve its own goals and encourages the collaboration to achieve SoS-level goals. We consider message-based interaction model, which defines the structure and element semantics of messages coming and going between entities. To quantitatively examine the limitations of ABC modeling and the effectiveness of the proposed method, we (3) set up experiments that compare the SoS-level goal achievements according to the modeling methods. The limitations in expressiveness of autonomous decision-making and of interactions are also quantitatively described and the proposed method is evaluated as the solution for them. Finally, we give the comparative

analysis on the simulation cost, i.e., simulation execution time.

The contribution of this study is summarized as follows:

- We propose the stateful decision-making model and the message-based interaction model to solve the limitations in expressiveness of the existing ABC modeling.
- We provide case study experiments to check whether the proposed model is the solution for the existing problems; expressiveness of autonomous decision-making and interactions.
- We analyze the execution time of simulation when the two methods are applied.

In the following chapters, we first briefly review the concept of SoS and agent-based modeling, ABC modeling and its limitations. Chapter 3 introduces our modeling method, ABC$^+$ with two supporting models. After that, we provide quantitative case study to see the effectiveness of the proposed modeling method and the effect on the simulation execution time in Chapter 4. In Chapter 5, we compare our methodology with other modeling and simulation approaches. We conclude the paper with the future works in Chapter 6.

Chapter 2. Background

2.1 System of Systems

SoS-level engineers or managers who integrate CSs into an SoS should consider CSs' autonomy and independence because the SoS-level managers may have no authority to force CSs to act in order to achieve SoS-level goals. If CS-level goals are not aligned with SoS-level goals, the SoS manager needs to provide them with proper inducements. The difference relationships among SoS entities such as CSs and SoS managers, establish different types of SoSs. In case of SoS manager, its level of authority also distinguishes SoS types. Maier [28] originally proposes three types of SoS: *directed*, *collaborative*, and *virtual*. Dahmann et al. [7] later completes the classification by adding *acknowledged* type to distinguish it from the directed type due to its lower authority of SoS manager than the directed one.

Seo et al. [36] organizes characteristics of SoS types. In directed SoS, SoS manager has full authority on CSs, and manages them to achieve SoS-level goals while they still operate independently. For example, when a national disaster occurs, the control tower acts as the SoS manager and manages various resources such as firefighters, ambulances, hospitals, etc. SoS manager in acknowledged SoS request CSs to cooperate by proposing various inducements to achieve SoS-level goals. This type can be matched a case of middle-level disasters, where the control tower provides available information and guides on the situation. In collaborative SoS, CSs have explicit or implicit agreement on SoS-level goals without a SoS manager. Lastly, a virtual SoS refers to a situation where there is neither SoS manager nor agreed SoS-level goals. The last two types can be applied to middle- or low-level emergency situations, where none or a little of collaboration is required.

To help readers understand, we bring a simple but intuitive SoS scenario called the Mass Casualty Incident (MCI) response scenario. MCI refers to a situation where the available resources and capabilities required for optimal response is short for the number, severity or type of live casualties [27], earthquake, attacks in a public place, etc. We suppose at the center of a city a big explosion occurred and buildings are broken down, where people are trapped. Autonomous and independent firefighters (i.e., CSs) are dispatched over the city to deal with the MCI, and each firefighter has the ability to search for trapped people and pull out them. Since all firefighters have the top priority in pulling out as many trapped people as possible, the independent firefighters may collaborate with each other. This is an example for a collaborative SoS. Contrary, if all firefighters may have different priorities and do not collaborate, it refers to an example for a virtual SoS. Meanwhile, a government agency may set up the control tower (i.e., SoS manager) for more effective and efficient control for the firefighters. If the control tower remotely guides the search area for each firefighter, it is an example for an acknowledged SoS. On the other hand, if the control tower forces the search area for each firefighter, it is an example for a directed SoS.

2.2 Agent-based Modeling

There are two major approaches in modeling a system; bottom-up and top-down. Bottom-up approaches takes the view that a system consists of many components while top-down view the system as a whole. Agent-based modeling takes the bottom-up approach to the target system. It is known by many other names, including individual-based modeling (IBM) in biology fields. It fits well to a system where its components are able to behave individually with independence. ABM is a good method to express SoS characteristics since SoS consists of autonomous and independent CSs [2]. Multi-agent systems (MAS) also define a system with many individual agents as its components.

Agents are considered a discrete individual with its own goals and rules to perform decision-making by itself [26]. It is situated so that it observes the environment and interacts with other agents around itself. The flexible characteristic gives the ability to learn and adapt to various situations over time using the experience and states from the observation and interactions. To achieve the flexibility, it needs some form of memory which it makes or modifies rules based on. However, an agent also have some boundaries according geographic or relationship, etc. so that the agent can access information only within the boundaries.

2.3 Action-Benefit-Cost Modeling

The Action-Benefit-Cost (ABC) modeling, proposed by Kim et al. [20], takes agent-based modeling paradigm to model SoS entity's decision mechanism with a simplified model. It addresses the lack of accessibility of SoS engineers to detailed decision-making mechanism of individual CSs. The essence of ABC modeling is that it focuses on the external behaviors (i.e., actions) of each entity. In the view of simulation, it is quite reasonable to consider only observable issues for examining an SoS. The decision-making mechanism that the entity takes a certain action under which conditions should be known and discovered to make a proper model of an entity. ABC modeling provides the concept of Action, Benefit and Cost, which successfully replaces the internal complex states of entities. One entity is defined as a set of entries which consists of Actions and corresponding Benefit & Cost.

The key assumption behind ABC modeling is that an autonomous entity behaves to maximize its utility by considering the benefits and costs of its possible actions. In other words, by abstracting the autonomous decision-making mechanism of entities in terms of the benefits and costs of taking the actions, it is possible to simply and effectively represent the autonomous behaviors of entities. This enables to present quantitatively the independent decision-making of autonomous entities, reducing the simulation costs in the terms of volume and complexity. For the working example, let us assume that a firefighter CS in the MCI area has four possible actions: move up, move down, move left, and move right. If it is known by domain experts that firefighter has the benefits and costs of the actions as Table 2.1, we can simulate the decision-making process and external behaviors of the firefighter. Consider the 3x3 grid map as shown in Figure 2.1. In Figure 2.1, a triangle represents the firefighter and circles represent trapped people.

Table 2.1: Cost-benefit table for Firefighter CS

Action	Benefit	Cost	Utility
Left	4	1	3
Right	3	1	2
Up	4	1	3
Down	2	1	1

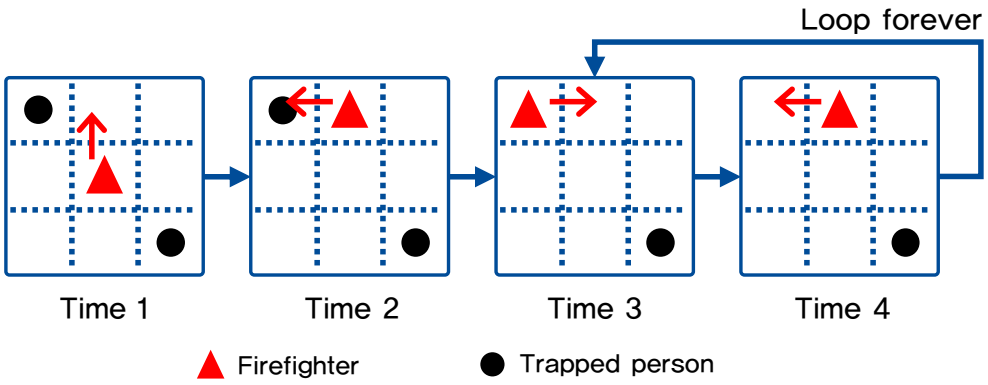


Figure 2.1: Problematic example of decision-making without memory. The firefighter keeps visiting the same area after Time 3.

The series of its decision-making and the behavior in ABC model follow the cost-benefit table given by the domain expert. Fixed cost-benefit table is a simple assumption to make a decision-making model, but it fails to express one's autonomy. An autonomous system should be able to make and adjust behavior rules to achieve its own goals effectively. However, the process requires some form of memory [26], the rational basis for the decision-making. ABC model suffers the unrealistic behavior of CSs, such as decision-making falling into the infinite loop. At time 1, the firefighter performs one of actions with the best utility at random, in this case it decides to move *UP*. At the next moment, however, moving *UP* is blocked since it faces the wall. It selects the second best action, moving *LEFT*. With these behaviors, the firefighter can achieve the pull-out goal on the trapped person at the left-top corner. Then, it takes the next action, moving *RIGHT* because left and up sides are blocked by walls. After and ever, it keeps moving *LEFT* and *RIGHT* according to the given cost-benefit table in advance of simulation. It visits the previously visited area again and again.

Even we tackle the first limitation, there is another hurdle in the existing ABC modeling–there is no communication method between entities. It assumes all information is known by all entities at the same time. This unrealistic assumption requires pre-evaluation of all possible interactions in advance of simulation. SoS entities exchange information or state memory at each moment during the simulation. It promotes one's decision-making to be more effective and efficient to the situation around him. For example, we send messages through the Internet or make call via. telecommunication. ABC model suffers the lack of expressiveness in entities communication, so that they can do nothing but behave

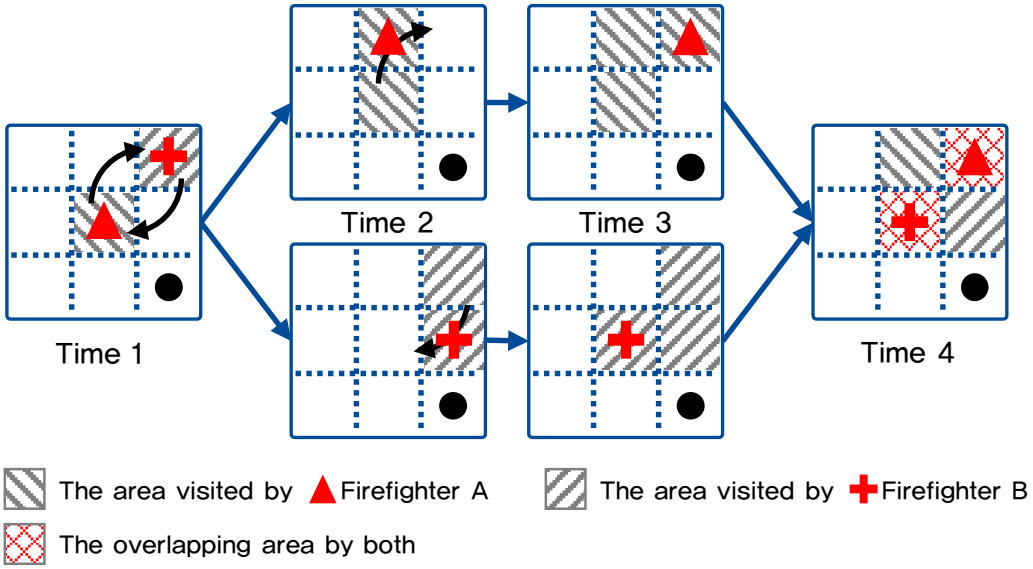


Figure 2.2: Problematic example of decision-making without communication. Since each firefighter's knowledge is not shared with the other through communication, there are overlapping areas visited by the two firefighters.

as in virtual SoS all the time. Figure 2.2 shows one example of the no-communication problem. Two firefighters search the MCI area to find out a trapped person. They move to visit not yet searched area as soon as possible. However, they have no communication channel so that they cannot share their search status. It results in the overlapping area for two firefighters to search the same places twice. This SoS-level behavior is observed during the simulation as the result of their autonomous decision-making mechanism. We cannot consider all the SoS-level behaviors in advance of SoS due to its uncertainty and non-determinism.

Chapter 3. ABC$^+$ Modeling

3.1 Models and Relations

3.1.1 Overview

We have stated that the existing ABC modeling has limitations in representing autonomous decision-making and interoperability of entities. ABC$^+$ modeling tries to solve these problems by introducing the concept of state into decision-making mechanism and the message-based interaction model. State as the storage of information is obtained by interaction with other CSs and SoS manager and by observing the environment. Figure 3.1 shows the overview of the models and their relations. The stateful decision-making model consists of State update, Cost-Benefit Analysis and Action Selection phases. An entity updates the state by observation and interactions, conduct cost-benefit analysis and select the best action according to the analysis result. The message-based interaction model represents the unit of interactions between entities. A message embodies the key elements required to define an interaction behavior, including the SoS-specific details in it.

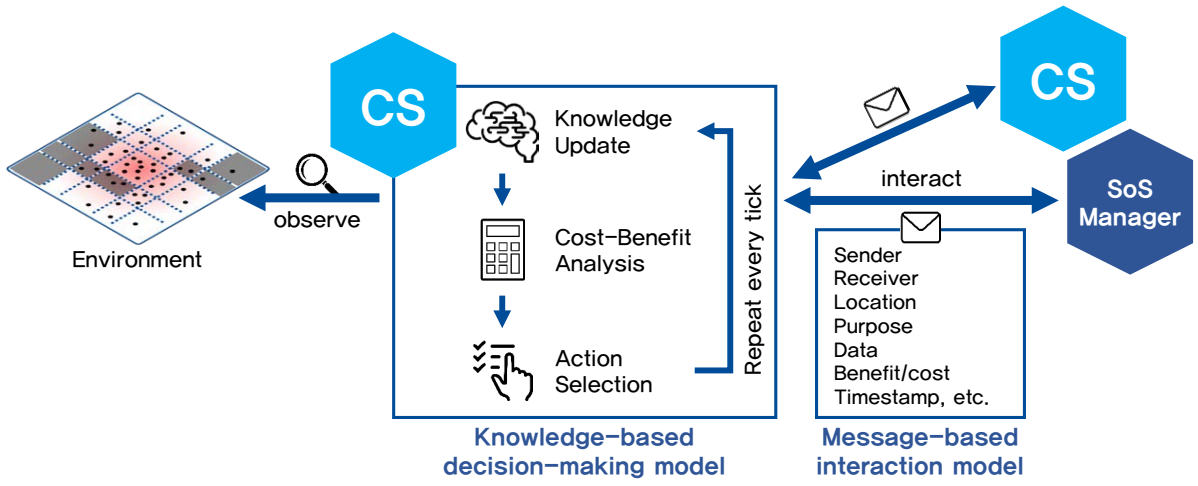


Figure 3.1: Overview of ABC$^+$ Modeling. Three-phased stateful decision-making is modeled in each entity and messaged-based interaction is modeled for communication between entities.

3.1.2 Stateful Decision-Making Model

To solve the first limitation (i.e., no support for decision-making based on state gained in the past), we attempt to introduce a state memory for each CS. Suppose one's decision-making mechanism in our daily life. We observe the world around us with the five senses. Especially, we communicate with other people and share information. With the states we have, we make every decision to do something. There are number of factors that affects our decision. Researchers have examined the decision-making

mechanism of human beings, organized and tried to implement the approximated mechanism in computer systems [34, 35, 13, 17].

ABC decision-making models provide the basic concepts that can be used in the process, being successful in considering the autonomy of entities and the limited accessibility on CSs in the SoS managers' point of view. It also has the advantage that the execution time is dramatically reduced in simulation because of the level of abstraction it provides. However, it lacks the essential element that is required to make decisions autonomously. Instead of using the benefits and costs predetermined by domain experts, an entity autonomously builds its own independent state, which it performs cost-benefit analysis with. Also, we define the three phases of one's decision-making mechanism with state update, cost-benefit analysis and action selection. The stateful decision-making model includes the specific steps of update state, the mechanism that the state is utilized in cost-benefit analysis and the ways of selecting the best action among the cost-benefit table. The detailed description of them and the pseudo-code that executes the model in simulation are as follows.

Phase 1: State Update

We use available information at every moment of decision-making. An entity of SoS as well may have individual and independent state in it, and utilize the information in decision-making. There can be various kinds of state information such as its own action history, resource availability, internal & environmental status (situation) or information obtained from other entities. The state changes over time. To update state, an entity observes the environment or interacts with other entities. One's state is the set of available information as Equation 3.1, where information is represented with the type, the value and several evaluation functions as shown in Equation 3.2. The single item, called *Information*, has the self-description of the type and the value as well as several evaluation functions which are used in the cost-benefit analysis phase. An evaluation function explains how the information should be quantitatively evaluated. Since the type of information is not limited to integer, it is necessary to provide the way that the information should be interpreted into integer value. At least one evaluation function should be provided. The resulting value is later used in determining either benefit or cost for one's capable actions.

$$\text{State } S = \text{the set of } \mathit{Informations} \tag{3.1}$$

$$\mathit{Information}\ \iota = (\mathit{type}, \mathit{value}, \mathit{eval}_1, \mathit{eval}_2, ..., \mathit{eval}_k) \text{ for } k \geq 1 \tag{3.2}$$

where: ι = Information, an item of one's state

type = The type of contents it contains

value = The value of contents it contains

eval_k = An evaluation function, which is used in cost-benefit analysis

In observation, an entity collects information of interest from the environment. Not every information of the environment but the available information which is accessible within certain boundaries such

as time, space and authority. Then, it receives and sends messages from/to other entities via. infrastructure that SoS provides. In real world, the message-based interaction may take several time due to physical procedures or transmission delays. However, we assume the message transmission takes place instantly in simulation. This process is described in Section 3.1.4.

Phase 2: Cost-benefit Analysis

An autonomous entity makes decisions by itself, considering its own state and goals. This analysis is a rational process, but bounded onto available information. This characteristic is called bounded rationality in agent-based modeling approaches [11]. In ABC$^+$ models, an entity performs cost-benefit analysis in advance of decision, referring to its own state. Benefit and cost is the values that the entity gains or pays when the corresponding action is taken. Phase 2 is the stage that generates the cost-benefit table which is referred in Phase 3, i.e., action selection. As shown in Equation 3.3 and 3.4, the table consists of ABCItems that each represents an *action* with the corresponding *benefit*, *cost* and *duration*, which represents the value in time unit about how long the action takes to complete in simulation. The evaluation of an action refers to calculating benefits and costs for ABCItems as shown in Equation 3.5 and 3.6. An action α has two weight matrices for benefit and cost, $weight_{\alpha b}$ and $weight_{\alpha c}$. Each value is the sum of weighted evaluation values of related information.

$$\text{Cost-benefit Table } A = \text{the set of } ABCItems \tag{3.3}$$

$$ABCItem\ \alpha = (action, benefit, cost, weight, duration) \tag{3.4}$$

where: α = ABCItem, an item of one's cost-benefit table

$action$ = The description of the action that the entity performs

$benefit$ = The value that the entity gains when the action is taken

$cost$ = The value that the entity pays when the action is taken

$weight$ = The weight matrix for benefit and cost calculation

$duration$ = The value in time unit how long the action takes to complete

$$Benefit\ benefit_\alpha = \sum weight_{\alpha b(\iota,k)} eval_{(\iota,k)} \tag{3.5}$$

$$Cost\ cost_\alpha = \sum weight_{\alpha c(\iota,k)} eval_{(\iota,k)} \tag{3.6}$$

where: $\iota \in I$ and $1 \leq k \leq |eval_\iota|$

$weight_{\alpha b(\iota,k)}$ = The weight in benefit for the action α of the evaluation function $eval_{(\iota,k)}$

$weight_{\alpha c(\iota,k)}$ = The weight in cost for the action α of the evaluation function $eval_{(\iota,k)}$

$eval_{(\iota,k)}$ = The kth evaluation function of the information ι

Phase 3: Action Selection

Given the cost-benefit table from Phase 2, an entity is finally able to make a decision. Compared to the previous phases, this phase is much simple and straightforward. The entity selects the best action considering the benefit and cost values. The simplest way is to take utility function that subtracts cost from benefit as shown in Equation 3.7. However, simple utility function may make numerous ties, so tie-breaking functions can be considered in addition. We can consider an action with higher benefit is preferred when two actions have the same utility. Similarly, there can be various tie breakers such as lower cost first, fewer number of non-zero weights first, excluding additional benefit and cost, or even random choice. At the end of Phase 3, the model returns the action with the best utility as the result of stateful decision-making process.

$$\text{Utility function } Utility(\alpha) = benefit_{\alpha} - cost_{\alpha} \tag{3.7}$$

$$\text{Selected action } \alpha_{best} = \arg\max_{\alpha} Utility(\alpha) \text{ for } \alpha \in A \tag{3.8}$$

3.1.3 Message-based Interaction Model

To solve the second limitation (i.e., no communication method), we attempt to introduce the message-based interaction model in addition to the stateful decision making model. To make a good decision, we need much information that will be used in analysis. In SoS simulation, we need to simulate interactions between SoS entities, which are the key source for one entity to get useful information. In agent-based modeling fields, Macal and North [25] describes the sociality of agents where interaction protocols include identification, spacial information, exchanged information and its influence, and other domain-specific mechanisms. Intelligent systems or agents, compared to traditional systems, are able to perform complex tasks on behalf of their users as independent from other systems [4, 14].

ABC modeling assumes that all possible interactions between SoS entities can be predictable and reflected in predetermination of benefits and costs. It means that all types of interactions and their participants and carried data is considered in advance of simulation. Since SoS has the uncertainty in SoS-level behaviors, which is characterized as emergence [31], resulting from autonomous and independent decision-makings of SoS entities, it is not easy to properly evaluate them all with the values of benefit and cost. Rather, it is more applicable and producing precise simulation results to model the interaction protocols and evaluate the models during the simulation repeatedly. ABC$^+$ modeling provides the message-based interaction model, where each message model describes identification of sender and receiver(s), purpose, data and other information specific to SoS simulation environment. It also provides the close relationship with the other models of ABC$^+$ such as Action-Benefit-Cost concepts and the stateful decision-making. The detailed description of the message-based interaction model is as follows.

Table 3.1: The elements of *Message* in ABC$^+$ interaction model

Element	Description	Example
Name	The short description of the message	Request to head to the building
Sender	The entity who sends it	Control Tower
Receiver	The one or more entities who receive it	Firefighter A
Location	The target location to which receivers are restricted	Anywhere
Purpose	The reason why it is sent to the receivers	Request (Action)
Data	The data carried by this message	*action*: Move to (1, 4)
Benefit	The value that a receiver gets as reward	5
Cost	The value with that a receiver is penalized	1
Timestamp	The moment when it is made and sent	14:02:30 24/Jan/2018
Trust	How the sender is trustful	High

Message Elements

In ABC$^+$ model, every communication between SoS entities is represented in the form of a message. The information is wrapped and delivered with related data via. SoS infrastructure. The basic elements are identification of people and information to be exchanged, that is sender, one or more receivers and data. To embody the SoS characteristics such as the collection of distributed, autonomous, independent but interoperable systems, it includes location, purpose, timestamp and trust elements in a message. Benefit and cost is as well included in order to support the intuitive and prompt evaluation on some messages like requests. The description for each element is listed in Table 3.1.

Suppose the MCI scenario similar to one in Section 2.3 as an example. The control tower, i.e., SoS-level manager, has collected information from the scene, and wants to guide Firefighter A to a certain building that is not yet searched. It sends out the message named with *Request to head to the building* to Firefighter A via. given SoS infrastructure. The message should identify the name of sender, the control tower and receiver, Firefighter A. Its purpose should be request, especially of action to the receiver. It provides the related data that is the *action* heading to the location of the target building, which is compatible to one of capable actions of the receiver. It specifies the inducement in the form of benefit and cost which will be given to the receiver when he accepts the request. Since the control tower is part of government, its trust is set as *High*.

Purposes

Every interaction has its purpose – we make requests such as asking something or asking someone to do something, make responses to the requests or simply gives information to others. Especially, in a certain situation, someone who has higher authority over another gives commands, which are compulsory, i.e., order. These five categories are listed in Table 3.2. They have different requirements on response

Table 3.2: Purposes of *Message*, R: requiring response, C: compulsory to the object entity

Purpose	Object Type	Description	R	C
Request	Information	Request some information at another entity	Y	N
	Action	Request to do a certain action at another entity	Y	N
Order	Action	Order another entity in lower authority to do a certain action	Y	Y
Response	Request	Respond to a request from the other entity	N	N
Delivery	Information	Deliver the information to another entity	N	N

Table 3.3: SoS Types and Interaction Patterns

SoS Type	Interaction Patterns
Virtual	No interaction
Collaborative	CS-CS
Acknowledged	CS-CS, CS-SoS manager (weak guidance)
Directed	CS-SoS manager (strong order)

and compulsion. While *request*s – Request Information and Request Action – requires responses but no compulsion on receivers, *Order* requires both responses and compulsion on receivers. *Response* and *Delivery* has no requirement on response and compulsion, but they are different in object type that each purpose type deals with. *Response* deals with a request and delivers the acceptance to the sender of the request, and *Delivery* deals with the information that is delivered from the sender to the receiver(s).

Interaction Patterns

As explained in Section 2.1, there are four types of SoS: virtual, collaborative, acknowledged, and directed. The relationship between entities is different according to SoS type. Virtual SoS assumes that each CS behaves independently while collaborative SoS assumes that CSs behave under the agreement on the basic level of collaboration by sharing information each other. Acknowledged and directed SoSs assumes that there exists SoS-level manager, who monitors the environment, collects information from entities and makes guidance or commands on entities. Especially in directed SoS, SoS-level manager has authority over other entities to order something to CSs. Table 3.3 shows the four SoS types and corresponding interaction patterns. Note that the interaction between CSs refers to the communication based on their states that is described in Section 3.1.2.

3.1.4 ABC$^+$ Model and Simulation

ABC$^+$ modeling is independent from specific simulation techniques. However, we assume agent-based discrete event simulation in this paper. For your understanding of how these models are integrated and utilized in the SoS simulation, we provide one way to execute these models using the simulation

procedure proposed by Kroiß [22]. Algorithm 1 shows the simulation procedure using the one in [22]. It is the base technology used in SIMVA-SoS [37] or SIMSoS [19]. Every time tick, the simulation engine evaluates decision-making of all entities, i.e., agents in [22]. To evaluate entities' decision-making properly, it simulates SoS infrastructure that delivers messages from one entity to other entiti(es) in the middle of evaluation. Then, it executes their decisions or actions and exogenous actions from the environment in random order.

The phased decision-making mechanism used in Line 5 of Algorithm 1 is depicted in Algorithm 2. To synchronize the phases of all entities during the simulation, we count the number of sent messages and repeat Phase 1 until all entities finish sending messages and consuming the incoming. This condition is checked in Line 6, 8-9 of Algorithm 1. Each entity's decision-making procedure follows the three phases in Section 3.1.2 in order. It observes the environment and receives and consumes messages incoming from the infrastructure. It sends out and publishes outgoing messages onto the infrastructure. The immediate actions is checked in Line 7 of Algorithm 1 and executed immediately during Phase 1. Then, when all entities in the SoS finish Phase 1, each performs Phase 2, the cost-benefit analysis, then finally selects the action with the best utility in Phase 3 and returns the action to the simulation engine.

Algorithm 1: Agent-based Discrete Time Simulation Procedure with ABC$^+$ model

world : World where the entities exist
entities: Entities like CS or SoS managers in the *world*

```
   // Entities update their states through observation and interactions
1  repeat
2      actions ← ∅
3      shuffle(entities)
4      foreach entity ∈ entities do
5          decision ← entity.makeDecision()
6          if decision ≠ null then
7              actions ← actions ∪ decision
8      infrastructure.deliver(publishedMessages)
9  until |entities| = |actions|
   // Expect environmental changes
10 exoActions ← environment.generateExogenousActions()
   // Simulation engine executes actions selected by entities
11 actions ← actions ∪ exoActions
12 shuffle(actions)
13 execute(actions)
```

Algorithm 2: Stateful Decision-Making Procedure

world: World that the entity belongs to

state : Its own state of the entity

```
// Phase 1: State Update
observe(world.environment)
if incomingMessages > 0 then
    consume(incomingMessages)
count ← publish(outgoingMessages, world.infrastructure)
if count > 0 then
    return null
// Phase 2: Cost-Benefit Analysis
foreach action ∈ actions do
    evaluate(action, state)
// Phase 3: Action Selection
sort(actions) according to utility
return the action with the best utility
```

3.2 Modeling Procedure

To evaluate the modeling method, we suppose the case scenario with the extension from the working example in Section 2.1 which is used for the experiment in Section 4. The detail of MCI-Response SoS is introduced as case scenario in the next section, and it is followed by specific steps how the stateful decision-making model and the message-based interaction model are modeled.

3.2.1 Case Scenario: MCI-Response SoS

Suppose that a big explosion has occurred and the system to respond the MCI situation is needed. We regard it as an SoS. Assuming that, the damage of the explosion is the biggest at the center of it and decreases according to the distance from the center. The buildings collapsed, whose severity differs by the damage they get. Inside the collapsed buildings people are trapped and waiting for firefighters to come and pull out them. The MCI-Response SoS consists of heterogeneous entities such as firefighters, ambulances, hospitals and control tower established by the government. The SoS-level goal is to rescue all trapped people from the MCI scene. SoS entities interact and collaborate in order to achieve the goal. At the same time, each CS has its own individual goal; firefighters try to pull out as many trapped people and as soon as possible, ambulances to transport the pulled-out patients to hospitals, and hospitals to cure the transported patients. The goal of control tower is the same as that of MCI-Response SoS but achieving the goal in a faster and efficient way.

We concretize the case scenario in order to specify a target SoS to model. We are going to use the resulting models in the experiment in Section 4. Figure 3.2 shows a setting of the MCI area. The city is represented as a grid, and each cell represents a building which is damaged. People are trapped

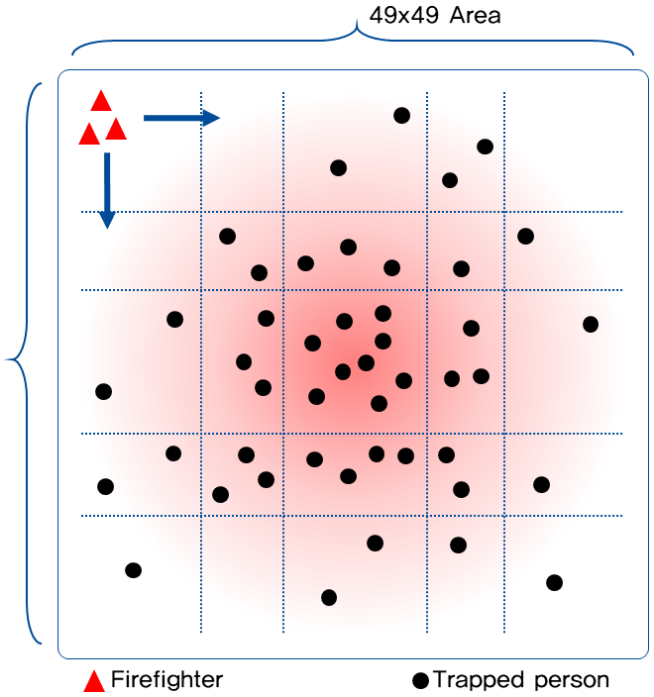


Figure 3.2: The MCI area with 49x49-sized grid in the case scenario. Each cell represents a building. A big explosion occurred at the center of the city and many people trapped in the buildings. Three firefighters try to search the trapped people and pull out them.

inside the damaged buildings. The explosion damage radiates outward, so that the number of trapped people is the largest at the center while it gradually decreases according to distances. To mimic the distribution of damage, we place them following the bi-dimensional normal distribution where each has a normal distribution with the mean as the central position and the standard deviation as a quarter of the axis length. To simplify the scenario, we consider only two types of SoS entities – firefighters and control tower. It still does not loose the generality because we are focusing on how each entity can be modeled with autonomous decision-making and interactions. Firefighters make decisions autonomously and interact with each other and control tower in acknowledged and directed SoS.

3.2.2 Modeling stateful Decision-Making

Our proposed decision-making model consists of three phases; state update, cost-benefit analysis and action selection. To make a proper model for one's decision-making model, we need to understand and include the state he focuses on and the mechanism how he analyze the information. Some steps may seem that it looses the abstraction level which is initially the statement of ABC modeling. However, not to loose the SoS characteristics, i.e., the lack of accessibility, note that we model the state and the mechanism only which is conjecturable from the view of SoS engineers. The following is the modeling steps for stateful decision-making model.

Step 1. **Collect the behavior history of the target entity and the situations** As an SoS engineer, modeling one's decision-making mechanism without authority requires many data to make ra-

tional guess on the target entity. We need as many historical data as possible, about externally observable behaviors of the entity and the situations where the behaviors are observed.

Step 2. **Identify externally observable actions of the entity** As sufficient data is collected, identify actions of the entity. Some behaviors can be categorized into discrete actions such as move northward or southward. On the other hand, there are some behaviors that cannot be categorized in a discrete manner. We may need to make a criterion under which behaviors should be categorized. The one thing to remember is that all external behaviors should be explained by at least one action.

Step 3. **Identify information available for the entity** This step identifies which information is related to the target SoS entity's decision. We call the item within the state as *information*, which is able to be defined as a simple data structure. We also describe type, value and evaluation functions for each information. An evaluation function should produce an integer value which is later used in benefit and/or cost. Evaluation of an *information* can be summation, average, counts, min/max or even complicated equations. including conditions and arguments.

Step 4. **Specify the weights according to behavior statistics** To produce the values for benefit and cost, we need to specify the weights with which each value from evaluation functions is considered. They mean how much positive and/or negative impact each state information has on actions. A value can be interpreted as benefit for one action and/or as cost for the other action at the same time.

Step 5. **Specify an utility function** Given the cost-benefit table, we need an utility function to finally select the best action as a final decision. It considers benefit and cost values. However, entities may have different utility function such as simple difference between benefit & cost or a pareto optimal choice by multi-objectives. Even tie-breaking functions may be needed additionally.

Step 6. **Validate the resulting model with the statistics** The final step is to validate the resulting model from the previous steps. A model is properly modeled when the simulation result is aligned to the statistics from the first step. If it does not follow the historical data, you need to repeat step 2 to 5.

Applying the steps to the case scenario, we can derive, for a firefighter CS, the state as Table 3.4 and the cost-benefit table as Table 3.5. A firefighter may have in his state three kinds of information. He searches the building at his location and counts (observe) the number of trapped people at this building, which is the first information or ι_1. The second information ι_2 is the given state of the distribution of trapped people. It is assumed to be given from the government and fixed throughout the simulation. ι_2 provides four evaluation functions, the description how the information can be evaluated. Given the location of a firefighter, each function summates expected numbers at the corresponding direction. ι_3 is the information about whether a building is searched or not. A firefighter keeps it updated by

Table 3.4: State example of Firefighter, NoTP: Number of trapped people

Information	ι_1	ι_2	ι_3
type	Integer	Integer matrix	Boolean matrix
value	NoTP in this building	Expected NoTPs	Whether buildings searched or not
$eval_1$	raw value	northSum(x, y)	northCount(x, y)
$eval_2$	-	southSum(x, y)	southCount(x, y)
$eval_3$	-	eastSum(x, y)	eastCount(x, y)
$eval_4$	-	westSum(x, y)	westCount(x, y)

Table 3.5: Cost-Benefit Table example of Firefighter, benefit and cost values are to be evaluated during the simulation

ABCItem	α_1	α_2	α_3	α_4	α_5
action	Pull-out	Move northward	Move southward	Move eastward	Move westward
$weight_{benefit}$	$\begin{bmatrix}10 & * & * & *\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & * & * & *\\ 4 & 0 & 0 & 0\\ 4 & 0 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & * & * & *\\ 0 & 4 & 0 & 0\\ 0 & 4 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & * & * & *\\ 0 & 0 & 4 & 0\\ 0 & 0 & 4 & 0\end{bmatrix}$	$\begin{bmatrix}0 & * & * & *\\ 0 & 0 & 0 & 4\\ 0 & 0 & 0 & 4\end{bmatrix}$
$weight_{cost}$	$\begin{bmatrix}2 & * & * & *\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}8 & * & * & *\\ 0 & 1 & 1 & 1\\ 0 & 1 & 1 & 1\end{bmatrix}$	$\begin{bmatrix}8 & * & * & *\\ 1 & 0 & 1 & 1\\ 1 & 0 & 1 & 1\end{bmatrix}$	$\begin{bmatrix}8 & * & * & *\\ 1 & 1 & 0 & 1\\ 1 & 1 & 0 & 1\end{bmatrix}$	$\begin{bmatrix}8 & * & * & *\\ 1 & 1 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$
duration	1	1	1	1	1

remembering his own or obtaining others' history from interactions. Similar to ι_2, it provides four evaluation functions, but each of which counts the unsearched buildings at corresponding direction.

In Phase 2, a firefighter evaluates his own action items according to the weights given in Table 3.5. Each firefighter has five capable action items; pull-out trapped people at this building, move northward, move southward, move eastward and move westward. They are all externally observable actions. In Table 3.5, α_1 has a big weight according to the number of trapped people, ι_1 in Table 3.4 for benefit. Other four movement actions have moderates weights according to expected numbers and unsearched buildings at corresponding direction. At the same time, they have small weights related to the numbers at other directions. It specially costs for the number of trapped people in this building with a big weight because pulling out people has the top priority over behaviors a firefighter can do. After the analysis, he calculates the utility of actions, following Equation 3.7.

3.2.3 Modeling Message-based Interaction

Our proposed interaction model focuses on the message-based representation of communication. To make a proper model for the interaction, we need to specify how interactions should take place in forms of messages. Identifying involved entities and message contents captures how they should make interactions. The following are the modeling steps for message-based interaction model.

Step 1. **Collect the interaction history within the target SoS** As the same as the case of decision-making model, SoS engineers need to collect many historical data in order to make a proper interaction model between entities. The information to be collected includes actual message contents, related communication and interaction protocols, response working protocols, etc.

Step 2. **Identify the sender** Every communication has a direction from one entity to another. The sender, who initiates communication, should be identified first. It defines purpose, receiver(s), and other elements such as data, authority, trust and timestamp.

Step 3. **Identify the receiver(s)** The number of receiver can be more than one while the sender is always identified as singular. The receiver passively participates in the interaction initiated by the sender. It can be identified as a specific entity or entity types. We may limit the receiver with the *location* element, so that restrict the receiver with a certain locational condition. Here we assume that all the entities consisting of the SoS are known each other.

Step 4. **Select a class of purpose** According to Table 3.2, we need to specify the purpose of the message. Purposes are relatively clear, but in the case of confusion, we consider three things; response requirement, compulsion to object entity and the object type it handles. Especially, *request* purposes may have additional benefit as reward and additional cost as penalty.

Step 5. **Describe the data and other elements** The next step is to describe the data and other elements such as trust, etc. The data is typed information which is represented as *information* of one's state. It is carried with specific value or requested by *null* value. Trust is related to the sender, but it is evaluated in the view of the receiver.

Step 6. **Validate the resulting model with the statistics** The final step is to validate the resulting model from the previous steps. A model is properly modeled when the simulation result is aligned to the statistics from the first step. If it does not follow the historical data, you need to repeat step 2 to 5.

Applying the steps to the case scenario, we can derive the interaction model as shown in Table 3.6. There are four types from virtual SoS to directed SoS, and each type has different interactions patterns as described in Table 3.3. Virtual SoS has no interactions between entities, so firefighters behave independently. In collaborative SoS, (1) firefighters share their state memory and try to search all areas as quick as possible. The shared state is their own search history, i.e., ι_3 in Table 3.4. Acknowledged SoS has the most highest number of interactions. Firefighters (1) share their state and (2) report to the control tower about its search progress. Control tower collects the information and make strategy to pull out people as soon as possible. It (3) requests firefighters to head to certain buildings not yet searched. Firefighters (5) answer the request with acceptance, considering its own goal, reward and penalty from the request. Directed SoS consists of CSs who are supposed to obey SoS manager's command. In that SoS, firefighters just report his search progress to the control tower and receives the command the control tower (4) orders.

Table 3.6: The messages according to the type of entity and SoS in ABC$^+$ model

SoS Type	Firefighter	Control Tower
Virtual	-	-
Collaborative	(1) Share my state to others	-
Acknowledged	(1) Share my state to others (2) Report my state to CT (5) Respond to the request	(3) Request firefighters to head to the unsearched
Directed	(2) Report my state to CT (5) Respond to the order	(4) Order firefighters to head to the unsearched

The case scenario does not exemplify all kinds of message purposes. However, the purposes are typical, so that it is not difficult to identify which purpose to the target message model. Sharing and reporting messages, i.e., (1) and (2), is defined with *delivery* purpose, request message, i.e., (3) from control tower with *request action*, order message, i.e., (4) of control tower with *order* and response messages, i.e., (5) of firefighters with *response* to request or order.

Chapter 4. Experiment

4.1 Research Questions

The goal of experiments is to quantitatively evaluate the expressiveness of both ABC model and ABC$^+$ model in simulation results and compare the simulation cost according to methods. The existing ABC model has limitations that it fails to express one's autonomous decision-making and interactions between SoS entities. It also fails to approximate real execution of target SoS and hinders SoS engineers and managers to analyze SoS-level behaviors. In our experiments, we have chosen the value and trends-in-progress of SoS-level goal achievement as the one way to examine SoS-level behavior. We consider the following research questions:

- RQ1: What impact do decision-making models have on SoS-level behaviors?
- RQ2: How does interaction model affect SoS-level behaviors?
- RQ3: How do ABC and ABC$^+$ models affect the simulation execution time?

The RQ1 is to evaluate the expressiveness of decision-making models of both ABC and ABC$^+$ models. ABC decision-making model has the limitation of fixed benefit & cost values and no memory to adjust those values in decision-making mechanism. It must have negative impact on behavior of each SoS entity and SoS itself as well. The simulation results are affected, then we cannot trust the analysis result based them. ABC$^+$ adopts state, a form of memory to solve the limitation. We examine the trend of SoS-level goal achievements over time since the limitation is expected to affect SoS-level goal achievement in the long term.

The RQ2 examines the expressiveness of the message-based interaction model in ABC$^+$ modeling. The existing ABC modeling has the assumption that all information is shared by all entities at the same time. There is a limit to reflect the information exchange in advance due to complexity, security or many other reasons. ABC model cannot approximate the realistic decision-making and behavior, which is based on interactions between SoS entities. ABC$^+$ modeling supports representing interactions with messages. It promotes to model the target SoS more accurate and analyze it easier. We examine the SoS-level goal achievement trends of different SoS types as time progresses as with RQ1.

The RQ3 is addressing the simulation cost of modeling methods. The simulation execution time is one of important factors in simulation-based SoS analysis. Many number of simulation trace samples are required because the uncertainty of SoS produces different execution traces for every repetition. ABC$^+$ models include additional information compared to the existing ABC models, so that it may increase the model size and the simulation complexity in terms of the execution time. We measure the execution time of the existing ABC models and ABC$^+$ models according to different interaction patterns of SoS types.

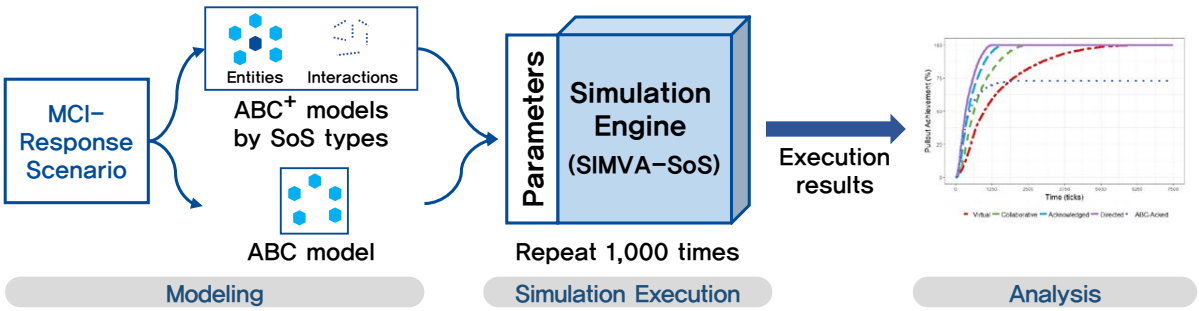


Figure 4.1: Overall experiment procedure and experimental setting

4.2 Experiment Setting

We have implemented MCI-Response case scenario in Section 3.2, using the derived stateful decision-making model and message-based interaction model. The experiment setting specifies 49-by-49 grid geographical map, distribution of trapped people with the mean of 24 and the standard deviation of 12. We varied two parameters, i.e., the number of total trapped people and the number of firefighters (CSs), to validate the implementation and see the variety of SoS-level goal achievement trends. The number of total trapped people is given with 50, 100, 150, 200, 250, and the number of firefighters is given with 2, 5, 10, 25, 50.

We made five versions; one for ABC model and the others for ABC$^+$ differing by SoS type. An ABC$^+$ model is split into two parts, which are stateful decision-making model and message-based interaction model. ABC$^+$ models have the same stateful decision-making process since the role and rule of SoS entity such as firefighters and control tower does not change at all. On the other hand, message-based interaction model changes by SoS types because different SoS types have different interaction patterns, including the existence of SoS manager, i.e., control tower.

The five models are put into simulation engine, which is the key component of SIMVA-SoS [37], the simulation-based statistical verification & analysis tool. It is firstly developed with the name of SIMSoS [19], which refers to SoS simulator, and its name has been changed recently in order to focus on statistical verification and analysis. It performs simulation in discrete event-based and agent-based manner. Its input model was ABC model by default, and we have paid our effort to support ABC$^+$ model in SIMVA-SoS.

Since we have 25 combinations in parameters coming from 5 values for trapped people and 5 values for firefighters, we finally have 125 different models for the whole experiment. As they produce non-deterministic results, we repeated 1,000 times for each model and averaged the measured values. In total, 125,000 simulation traces are used to answer the research questions. The Experiment process is summarized as Figure 4.1.

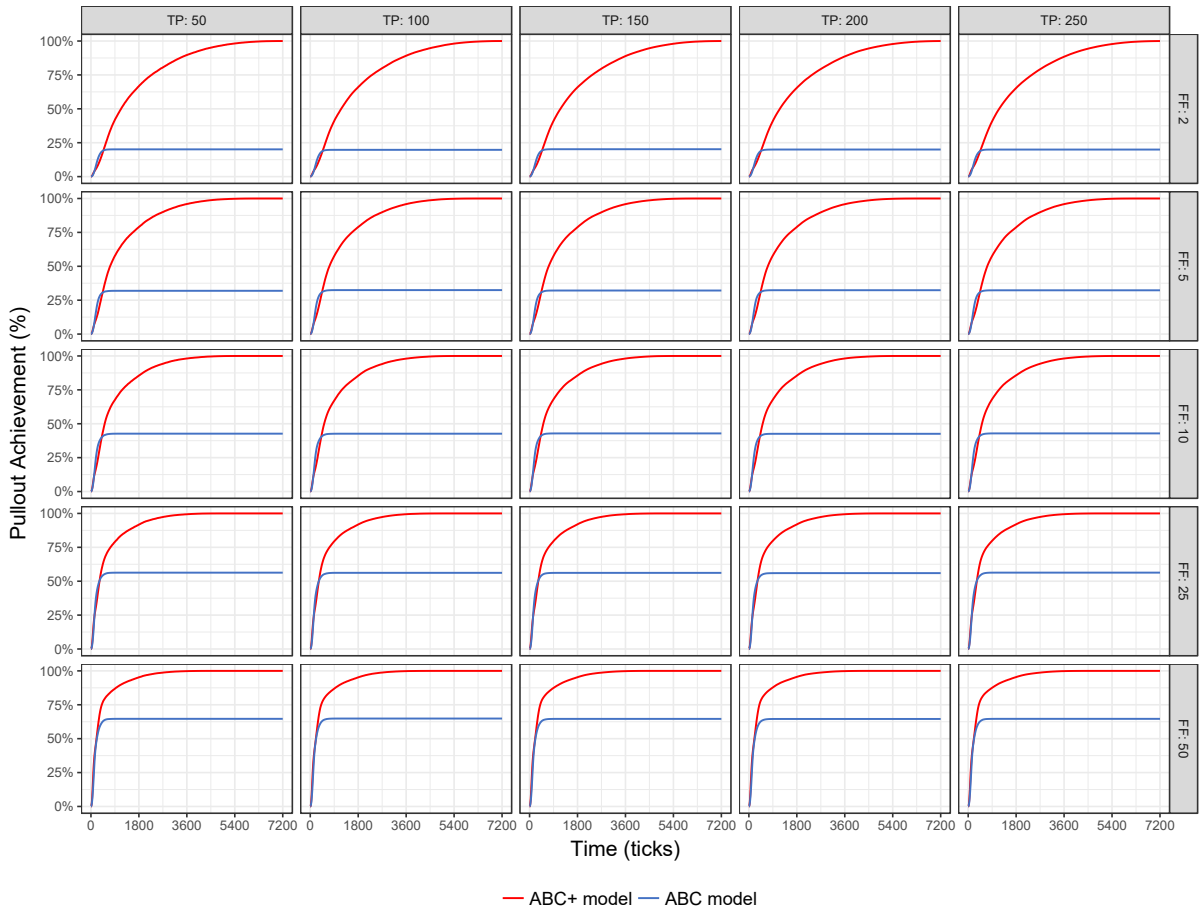


Figure 4.2: SoS-level goal achievement trends considering state, TP: total trapped people, FF: firefighters

4.3 Results and Analysis

4.3.1 RQ1: What impact do decision-making models have on SoS-level behaviors?

Before we examine the impact of stateful decision-making model to SoS-level behavior, we need to ask one question first – Does decision-making without some form of memory really make any problems in approximating real SoS execution? To answer the question, we made ABC model for the case scenario and watch the SoS-level goal achievement trends over time. As mentioned in Section 2.3, there must exist the evidence that results from the infinite loop, which is caused by fixed cost-benefit table. We assumed that trapped people does not get worse in their health to remove its effect to the SoS-level goal achievement over time. Figure 4.2 shows that the goal achievements of ABC model stop increasing at some moments. The moment differs by combinations of parameters. However, ABC model always has the loop in decisions, which stops increasing SoS-level goal achievement at a certain level. This infers that firefighters cannot help moving within limited areas because their decision-making mechanism has no power to escape a loop.

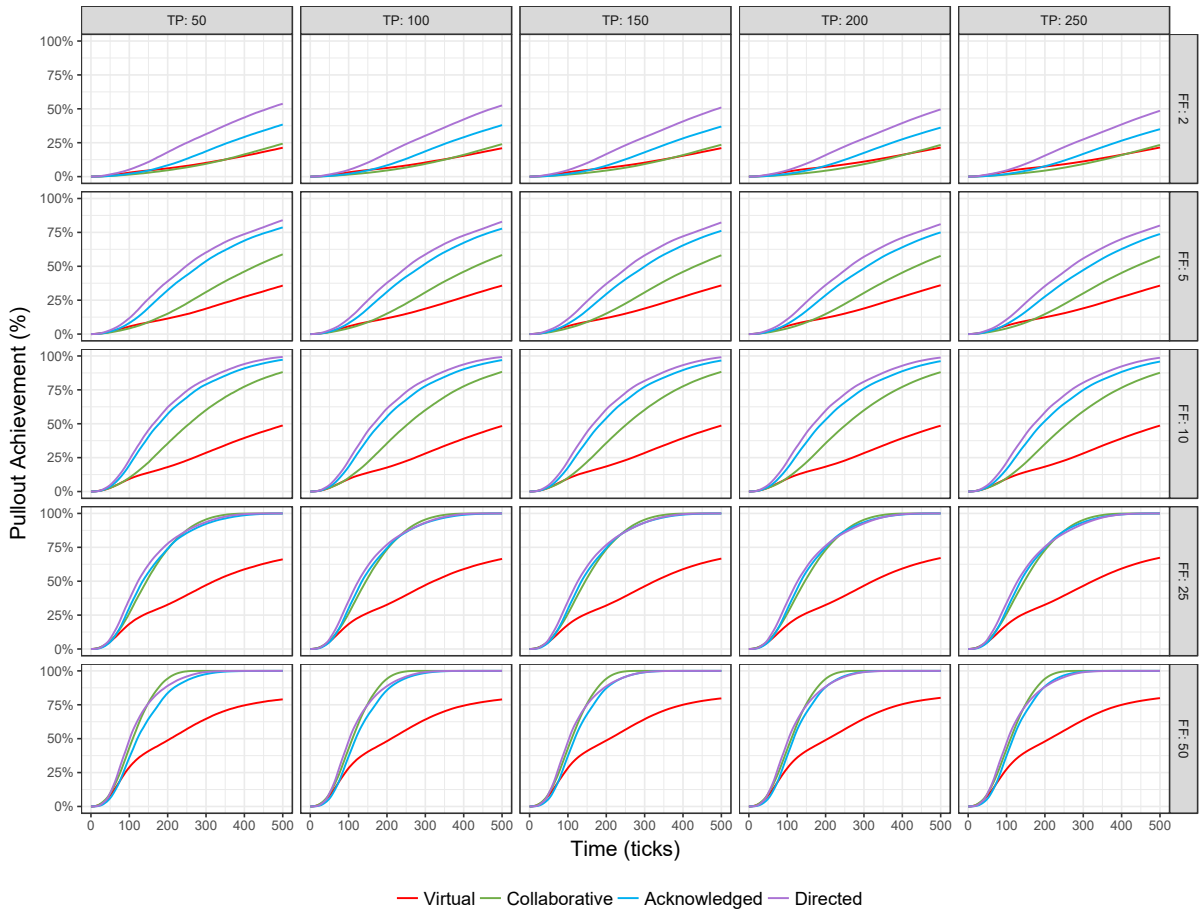
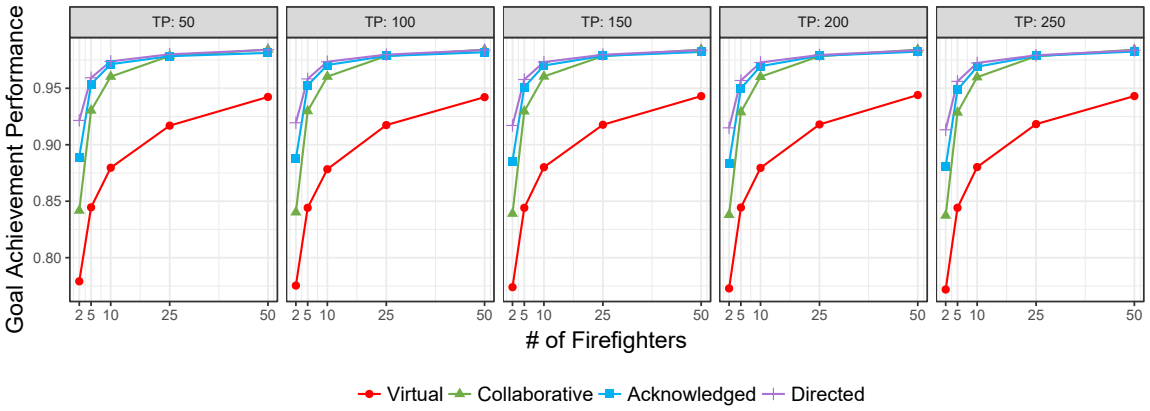


Figure 4.3: SoS-level goal achievement trends for different SoS Types, TP: total trapped people, FF: firefighters

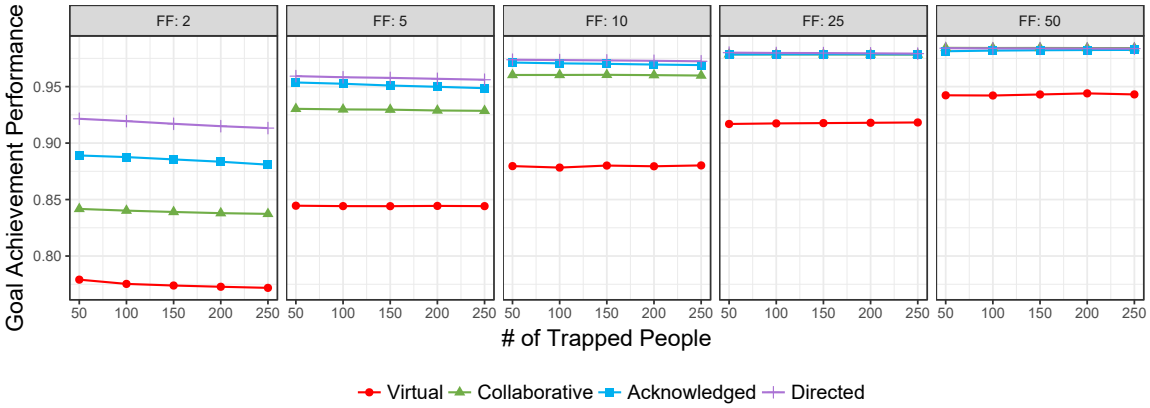
On the other hand, ABC$^+$ model shows the different trends where SoS-level goal achievements keep increasing until 100% pull-out. The ABC$^+$ model in this experiment is a virtual SoS model in order to remove the effect of interactions between SoS entities. The other types such as collaborative, acknowledged and directed SoS has interaction models between entities, which must affect SoS-level goal achievements. Firefighters make decisions solely by themselves, and it infers that the limit of ABC model is a matter neither of case scenario or interaction but of decision-making model. However, virtual ABC$^+$ model still achieves the goal eventually at some late moment, while ABC model does not. Only adding the stateful model to the existing ABC model, SoS simulation becomes working properly.

4.3.2 RQ2: How does interaction model affect SoS-level behaviors?

In addition that the stateful model is essential to simulate an SoS properly, we would like to check whether the interaction model plays an essential role. The second limitation in Section 2.3 mentions that the existing ABC model urges that every information is reflected in interactions, but we have addressed that it may be difficult or impossible. We even have to remake the whole model when different interactions between the same SoS entities has need of analysis. The message-based interaction model



(a) SoS-level goal achievement performance by SoS types and the number of CSs



(b) SoS-level goal achievement performance by SoS types and the number of total trapped people

Figure 4.4: SoS-level goal achievement performance by SoS types, TP: total trapped people, FF: firefighters

separates the interaction behaviors from one's decision-making model, and makes analysis on different interaction patterns easier than before.

Figure 4.3 shows the different trends of SoS-level goal achievement for parameter combinations. Five values for firefighters and four SoS types makes 20 different settings of MCI-Response SoS in five values for total trapped people. With ABC modeling, we need to build each model of 20 settings from scratch. However, we can make it with ABC$^+$ modeling by one decision-making model and four interactions models, i.e., 5 models in total. In addition, it is more easier and clearer to make proper models since ABC$^+$ models are more intuitive than ABC models.

As shown in Figure 4.4a, the number of firefighters shows dramatic differences in SoS-level goal achievement trends while the number of total trapped people appears to have no impact on their results in Figure 4.4b. The more firefighters work together, the gap between their goal achievement trends as SoS becomes narrower and shows similar SoS-level behaviors even in different types of SoS. Acknowledged and directed SoS meet first because they have a common thing – SoS manager's interference, then collaborative SoS is followed. Compared to other types, virtual SoS has only small improvement by the number of firefighters in SoS-level goal achievement. However, it is very natural, thinking of no

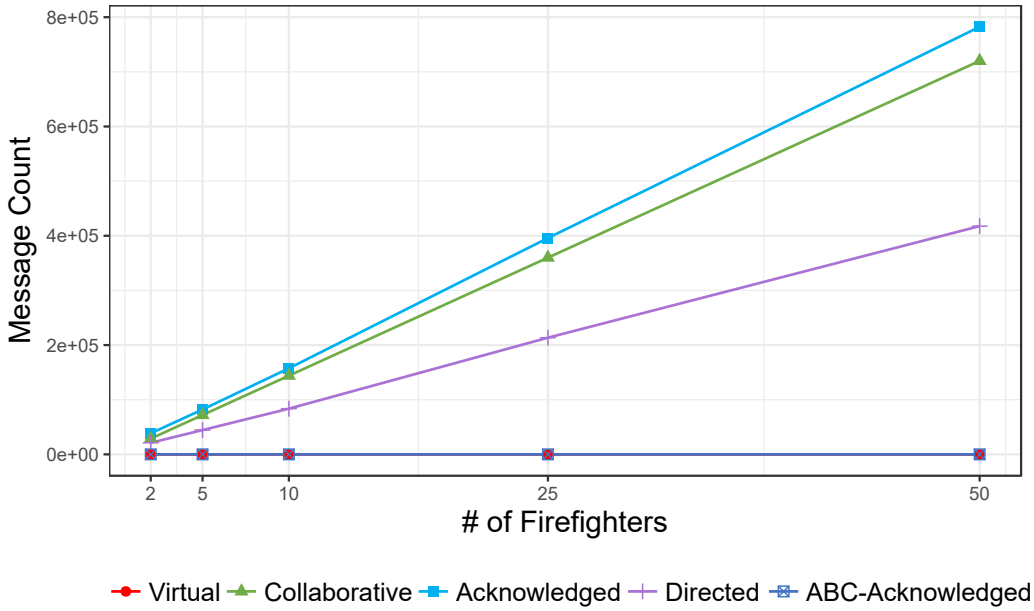


Figure 4.5: Message counts of different interaction and entity models

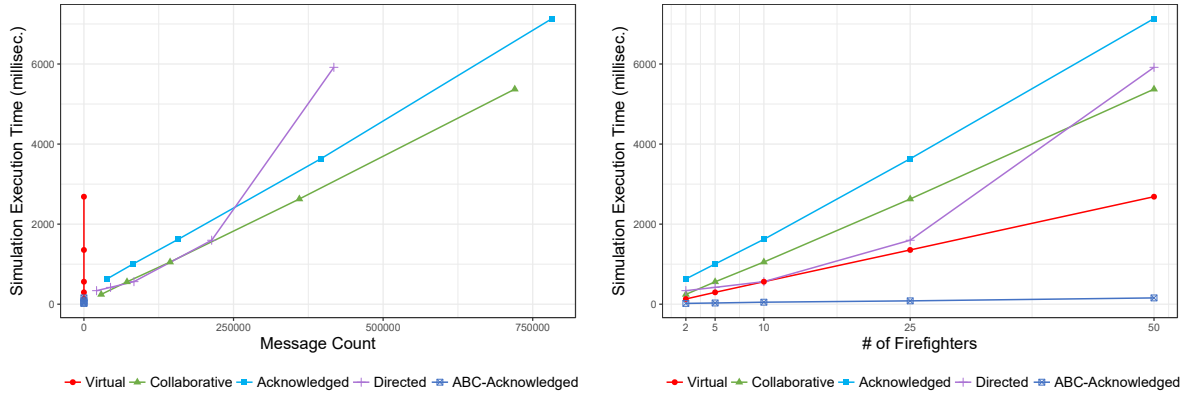
collaboration between entities.

Even surprising is that directed and acknowledged SoSs with SoS manager takes the top performance in the case of small number of firefighters, but collaborative SoS is more efficient than those when it comes to the case of large number of firefighters. This is an SoS-level emergent behavior which can be observed by ABC$^+$ interaction models. To quantitatively tell the difference, we measured the Goal Achievement Performance (GAP) with the Area Under Curve. The GAP values ranges from 0 to 1; the higher GAP refers to SoS goal achievement earlier in time. The GAP values for the case of 2 firefighters are virtual: 0.7753, collaborative: 0.8402, acknowledged: 0.8876 and directed: 0.9194. When it comes to the case of 50 firefighters, they are 0.9421, 0.9842, 0.9819 and 0.9839 in the same order.

4.3.3 RQ3: How do ABC and ABC$^+$ models affect the simulation execution time?

The larger size of simulation model leads to the higher complexity to execute the model. Analyzing SoS behavior requires many of simulation trace samples for SoS engineers to make a judgment because SoS has the uncertainty and the non-determinism in nature. Simulation execution time is the key factor to reducing the whole analysis process. Simulation-based statistical verification requires many of simulation trace samples to make the judgment whether a property is satisfied. Simulation execution time is the key factor to reducing the whole verification process. ABC$^+$ modeling defines the concept of state in one's decision-making mechanism and suggest how it would be utilized. It also propose the message-based interaction model in order to successfully represent various interactions between SoS entities. The size of ABC$^+$ model must be relatively larger than that of ABC model for the same scenario.

Figure 4.5 shows the average of message counts in five different models. There is no significant



(a) Simulation execution time by message counts

(b) Simulation execution time by the number of CSs

Figure 4.6: Simulation execution time for different SoS types. The simulation execution time varies with the existence of a SoS manager and the types of interaction patterns.

difference for the number of trapped people. It is the case of 100 total trapped people. Message counts increase in linear form according to the number of firefighters (CSs). The interaction models of the case scenario follows protocols that produce linear interactions between SoS entities. It means that the simulation execution time will increase linearly if the interaction is the key factor to it.

The first factor that may have impact on the simulation execution is naturally the message counts. Figure 4.6a shows the relationship between message counts and the simulation execution time. Virtual SoS and ABC models has no messages between SoS entities, so that it is shown as a vertical line in the graph. Collaborative and acknowledged SoS models follows linear trends as their message counts. However, directed SoS model shows an exponential increase in simulation execution time even though its message count does not. It infers that there is another factor that affects the simulation execution time. The second factor which may have impact on the simulation execution time is the size of models. The size of interaction model is fixed, but that of decision-making model can be increased according to the number of SoS entities involved. We have varied the number of firefighters, the key CS in the case scenario from 2 to 50. Figure 4.6b shows that directed SoS model still shows an exponential increase while other models do not.

The results of Figure 4.6 comes from the assumptions in the case scenario. All message-based interactions takes place via. SoS infrastructure, and it takes very short time to deliver a message from one entity to another. The decision-making models are also linear because adding one entity means that adding one message participants in the interaction model. It does change the trends of neither message counts nor the simulation execution time. The reason why the simulation execution time of directed SoS increases exponentially is that the centralization of decision-making processing. Directed SoS model has the SoS manager who collects all the information from CSs and processes overwhelming information. The process that SoS manager takes is designed in an exponential increase according to the scale of information it handles.

Figure 4.7 shows the average of simulation execution time for five different models; one is the ABC

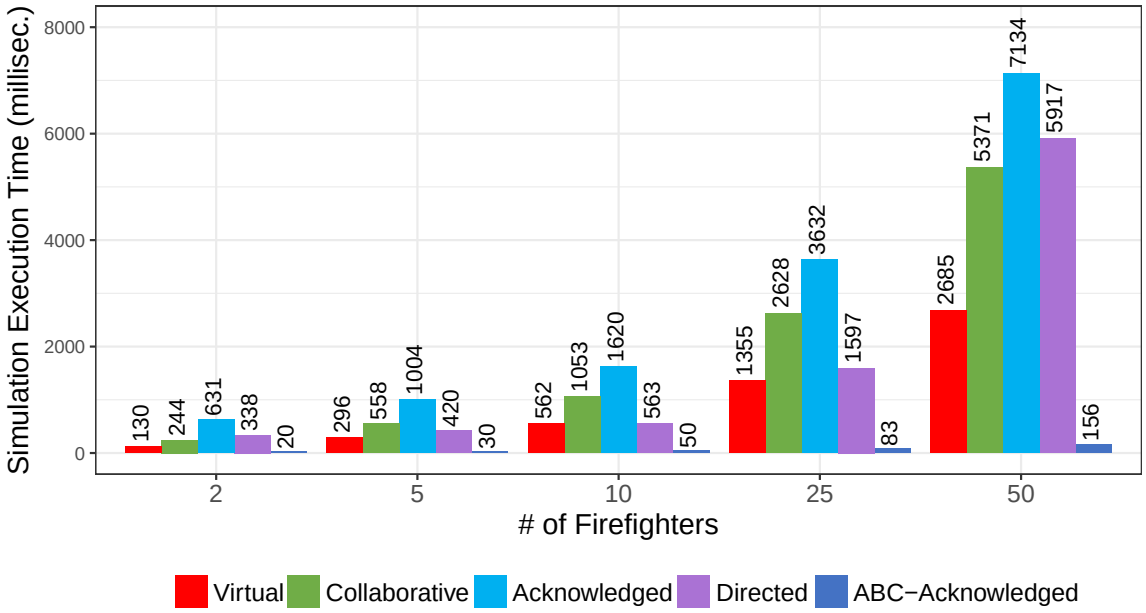


Figure 4.7: Simulation execution time for different models

model in RQ1 and the others are the ABC$^+$ models in RQ2. As expected, the execution of ABC$^+$ models takes longer than that of ABC model; acknowledged ABC$^+$ model takes 45.7 times longer at most than ABC model. Comparing four types of SoS in ABC$^+$ models, virtual SoS takes the shortest while acknowledged SoS takes the longest in their execution, following their message counts. Even though there is the wide gap between the execution time from ABC modeling and that of ABC$^+$ modeling, the execution time is very tightly related to the expressiveness of each modeling method. It is up to SoS engineers and modelers' decision to use which modeling method for SoS analysis. ABC models have the shorter execution time but the lesser expressiveness, so that they may lose the correctness of verification results. On the other hand, ABC$^+$ models sacrifice the execution time to attain the higher expressiveness and the reliability on analysis results.

4.4 Threats to Validity

The case scenario can be a threat to validity because it might be too simple to represent a real world SoS example. An SoS in real world can have a huge number of heterogeneous CSs and various complex environmental situations and resources. Even simplified model must be much bigger and more complicated than the case study models are. Our scenario is considering three firefighters and one control tower as SoS manager. However, this paper proposes a modeling approach and we focus on the impact of stateful decision-making and interactions on SoS-level behaviors. The experiment results show that we need to include state and interactions in our SoS simulation model in order to improve its expressiveness and to properly approximate real SoS execution. Figure 4.2 and Figure 4.3 shows that there is no difference in the ratio between the number of total trapped people and the number of firefighters. However, it does not harm the effectiveness and expressiveness of ABC$^+$ modeling, but it

just an assumption of the case scenario – firefighters have two categories of actions that pull-out and move but both take the same duration of time although their duration may be different in real world.

Another threat is in the evaluation methods. Our experimental results seems natural and plausible. The lack of state causes a loop in decision-making and hinders SoS-level goal achievement to increase after some moment. It also shows different trends in SoS-level goal achievement and the effects of different parameters such as the number of total trapped people and firefighters (CSs). However, we are just replacing an evidence to verify the model with quantitative results. We should qualitatively examine the simulation traces to find out SoS entity make decisions and interact as intended.

Modeling effort is also one of threats since ABC$^+$ modeling is apparent to require larger efforts to make models. We have tried to examine the modeling effort for ABC modeling and ABC$^+$ modeling, however, it can be done only by an indirect way because they have no concrete model representations. We measured the number of lines of code (LOC) of implementations in Java. ABC$^+$ models have longer length in LOC than ABC models. LOC of firefighter ABC model is 165. LOC of ABC$^+$ models are; common decision-making and interaction: 106/17, firefighter's: 230/65, control tower's: 67/59. However, ABC$^+$ models are much easier to implement because modeling parts are separated and structured.

Chapter 5. Related Work

In the modeling and simulation, there is no silver-bullet method for various purposes. Different methods can be used according to modelers' purposes. Baldwin et al. [2] compares Event-Based Modeling (EBM) and Agent-Based Modeling (ABM) approaches in the purpose of SoS analysis and verification. It models a journal collaboration scenario in both EBM and ABM, and finds out that ABM is more appropriate to see characteristics of SoS than EBM is, even though EBM has the strength in proving the simulation results. ABM modelers can easily produce entity models and see simulation results from rich and complex interactions between them. Combining ABM and EBM at a proper level of abstraction, easiness in modeling and validation can be obtained. Researches about the system where individuals are networked [6, 18] takes ABM as their modeling and simulation paradigm in order to produce interactive and emergent behaviors.

The one of prevailing paradigms to model the individual knowledge base is the Belief-Desire-Intention (BDI) agent modeling [34, 35]. It captures the human decision-making process to represent a series of three mental attitudes: belief, desire and intention. *Belief* is the informational state that perceives new information and update the belief along with the information and environment. *Desire* is the motivational state that generates the list of actions which are capable according to its status and goals. *Intention* is the deliberative state where the agent chooses among the list what is the best to achieve its goals and executes it. It is also defined by formal methods such as linear temporal logic (LTL) and computation tree logic (CTL). These researches allow modelers to generate a formal model for verification like model checking. However, we usually face the barriers in using them that practitioners are unfamiliar with specification processes, notations, etc. On the other hand, ABC$^+$ modeling provides a descriptive and intuitive way to model one's decision-making, focusing on the important things required in SoS simulation. It lowers the difficulty and makes translation easy from the resulting model itself to the implementation.

Another paradigm is MAPE-k feedback loop which is the autonomic computing architecture proposed by IBM [13, 17]. AMADEOS project [1] mentions MAPE-k as an appropriate control loop model for SoS. The architecture which includes *Monitor*, *Analyze*, *Plan* and *Execution* functions serves building the structure of self-configuring, self-healing, self-optimizing, self-protecting systems. The monitor function collects the details from the currently managed resources. The analyze function observes and analyzes situations to change some resources in the system. The plan function creates a procedure for desired changes. The execution function schedules and performs the necessary changes to the system. It requires detailed information to derive a proper model for one's decision-making. However, ABC$^+$ modeling takes the assumption that the detailed decision-making process is unknown because SoS engineers have no authority to access it. ABC$^+$ models try to make rational guesses from externally observable behaviors via. economic elements such as benefit and cost. It is one of ways to secure autonomy of SoS entities in SoS modeling and simulation.

Multi-Agent Systems (MAS) are distributed systems composed of entities called agents, which shares many with SoSs. Agent-based modeling & simulation methods developed in the field of MAS, can be adopted with few configuration efforts. The Foundation for Intelligent Physical Agents [33] (FIPA) has developed standards for heterogeneous and interacting agent-based systems. It describes the FIPA MAS models supporting information sharing with the form of messages through Agent Communication Language (ACL) [10]. A message consists of parameters including performative, content, language, encoding, ontology, protocol, etc [9]. One message can be described by its own grammar rules. Performative, which corresponds to purpose in ABC$^+$ interaction model, defines 20 of them. They are classified by more concrete and specific units than those of ABC$^+$ modeling.

In the information systems field, the Knowledge Query and Manipulation Language [8, 24] (KQML) is a de-facto ACL for communication between knowledge-based systems. It defines the concept of the message and protocol, which is used to interact between information systems or intelligent agents. It, as well as FIPA ACL, describes a message by its own BNF grammar rules. Many agent-based modeling frameworks such as Jade [3], Jack [5], Zeus [32], etc. are developed using FIPA standards and KQML. It assumes a non-zero delay in message transportation via. unidirectional communication links. KQML has a lot of performatives related to information and resource acquiring and response, including management on facilitation and networking itself.

Agent communication languages like FIPA ACL and KQML are mature in modeling message-based communication between entities. Their focus is to provide standards for actual implementation of MAS. ABC$^+$ modeling omits some categories like negotiation and error handling, which are realized by performatives of FIPA ACL. ABC$^+$ assumes a zero delay in message transportation via. SoS infrastructure. However, ACLs do not successfully consider SoS-specific details suggested by Kopetz [21] such as data and meaning, physicalism, neutrality, temporal aspects. SoS simulation is an abstract version of real SoS execution. We can safely omit some details for easiness in modeling and simulation but SoS-specific details needed to approximate the execution successfully.

Chapter 6. Conclusions

As various domain fields have demands on the large-scale and complex systems such as SoS, modeling and simulation methods which represent target systems in a proper level of abstraction are required for system engineers or managers. Simulation as approximation to real system execution, is a great tool to see the whole-level behaviors of a system. To deal with the demands of SoS engineers and managers, ABC modeling method is proposed, which reflects the lack of accessibility to detailed decision-making mechanism. It simplifies the model with economical terms such as Action, Benefit and Cost. However, it has limitations on expressiveness. ABC modeling does not consider autonomous and independent decision-making using one's own memory and various interactions between SoS entities. It blocks a proper approximation of real SoS execution and makes modeling and analysis difficult by reflecting all possible interactions in advance of simulation.

In this paper, we proposed ABC$^+$ with two supporting models. We introduced the concept of state and the phased decision-making mechanism and the message-based interaction model. Stateful decision-making allows an SoS entity to independently access its situation and adjust the cost-benefit table by itself, so that autonomous decision-making is represented by the model. Message-based interaction model gives an explicit way to describe the communication and collaboration among SoS entities. It allows SoS engineers to make SoS models in an intuitive, efficient and easier way. They all are able to SoS characteristics such as autonomy, managerial and operational independence, interoperability and emergence. With the detailed description of models and the modeling guidelines, it is expected for SoS engineers to model the structure of SoS and its SoS-level behaviors effectively and efficiently.

To demonstrate our proposed modeling method, we made the case study called MCI-Response SoS. The experiments shows the quantitative analysis on the limitations of ABC modeling and the effectiveness of ABC$^+$ modeling. A problematic example caused by fixed cost-benefit table is introduced with plots where SoS-level goal achievement stops increasing after a certain moment. ABC$^+$ models have no such a problem in their simulation results. In addition, ABC$^+$ models have split models into decision-making and interaction models, which explicitly shows the simulation results with clearer interpretations. Now we can control the effect of SoS entities and that of interaction patterns between them, with the aid of ABC$^+$ modeling approach. Speaking of simulation cost, ABC$^+$ models require more execution time than ABC models. However, we think the loss in expressiveness is a much more serious problem than simulation cost because it directly affects the reliability of analysis results.

ABC$^+$ modeling has various directions for further researches. Machine learning or game theory [29] can be used in the cost-benefit analysis phase in one's decision-making process. The most difficult step in modeling one's decision-making is to determine the weights with which how benefit and cost for actions are evaluated. There can exist many possible ways to model the same behaviors. The weights are the key factor to model a behavior that is only externally observable. SoS engineers may want to refine their SoS

entities' decision-making models, and machine learning or game theory will be a good tool for making better models. Another direction is about action selection. Many other utility functions can be used in the phase, which support coverage-based selection [38], multi-objective selection, or other sophisticated ways rather than a simple utility function. These functions may enlarge the extent of expressiveness in decision-making mechanism. The other direction is about other supporting models, which are compatible with the existing ABC$^+$ models. We may need models on ontology and environment, which are closely related to one's decision-making. They provide the worthy information to study which information and resources can be used and interpreted in some ways.

Bibliography

[1] AMADEOS Project. AMADEOS METHODOLOGY, MECHANISMS, AND IMPLEMENTATIONS. *Deliverable*, 3(2), 2016.

[2] W. C. Baldwin, B. Sauser, and R. Cloutier. Simulation approaches for system of systems: Events-based versus agent based modeling. *Procedia Computer Science*, 44:363–372, 2015.

[3] F. Bellifemine, A. Poggi, and G. Rimassa. Jade: A fipa-compliant agent framework. In *Proceedings of PAAM*, volume 99, page 33. London, 1999.

[4] J. M. Bradshaw. *Software agents*. MIT press, 1997.

[5] P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas. Jack intelligent agents-components for intelligent agents in java. *AgentLink News Letter*, 2(1):2–5, 1999.

[6] S. Cho, L. Knapen, T. Bellemans, D. Janssens, G. Wets, et al. A conceptual design of an agent-based interaction model for the carpooling application. *Procedia Computer Science*, 10:801–807, 2012.

[7] J. S. Dahmann and K. J. Baldwin. Understanding the current state of us defense systems of systems and the implications for systems engineering. In *Systems Conference, 2008 2nd Annual IEEE*, pages 1–7. IEEE, 2008.

[8] T. Finin, R. Fritzson, D. McKay, and R. McEntire. Kqml as an agent communication language. In *Proceedings of the third international conference on Information and knowledge management*, pages 456–463. ACM, 1994.

[9] A. Fipa. Fipa acl message structure specification. *Foundation for Intelligent Physical Agents, http://www. fipa. org/specs/fipa00061/SC00061G. html (30.6. 2004)*, 2002.

[10] F. Fipa. specification part 2: Agent communication language. Technical report, Technical report, FIPA-Foundation for Intelligent Physical Agents, 1997.

[11] N. Gilbert. Agent-based models, quantitative applications in the social science153. *London, Thousand Oaks*, 2008.

[12] V. Grimm and S. F. Railsback. Agent-based models in ecology: patterns and alternative theories of adaptive behaviour. *Agent-based computational modelling*, pages 139–152, 2006.

[13] P. Horn. Autonomic computing: Ibm\'s perspective on the state of information technology. 2001.

[14] M. N. Huhns and M. P. Singh. *Readings in agents*. Morgan Kaufmann, 1998.

[15] M. Jamshidi. *System of Systems Engineering: Innovations for the Twenty-First Century*. John Wiley & Sons, 2011.

[16] T. Kang and D. Mavris. A systems-of-systems approach for application to large-scale transportation problems. In *Proceedings of 15th Annual International Symposium of INCOSE, Rochester, NY, July*, pages 10–14, 2005.

[17] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[18] I. K. Kim, S. H. Lee, and H. J. Seo. Agent based interaction model for electronic health record system. In *Pacific Rim International Workshop on Multi-Agents*, pages 337–350. Springer, 2005.

[19] J. Kim. Model-based System of Systems Verification considering Selfishness. Master's thesis, Korea Advanced Institute of Science and Technology, Daejeon, Korea, 2017.

[20] J. Kim, D. Shin, and D. H. Bae. An Applicability Study of Action-Benefit-Cost Model and Statistical Model Checking for System of Systems Goal Achievement Verification. *KIISE Transactions on Computing Practices*, 23:256–261, 2017.

[21] H. Kopetz. A conceptual model for the information transfer in systems-of-systems. In *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2014 IEEE 17th International Symposium on*, pages 17–24. IEEE, 2014.

[22] C. Kroiß. Simulation and statistical model checking of logic-based multi-agent system models. In *Agent and Multi-Agent Systems: Technologies and Applications*, pages 151–160. Springer, 2014.

[23] A. J. Krygiel. Behind the wizard's curtain. an integration environment for a system of systems. Technical report, Office of the Assistant Secretary of Defense Washington DC Command and Control Research Program (CCRP), 1999.

[24] Y. Labrou and T. Finin. A proposal for a new kqml specification. Technical report, Technical Report Technical Report TR-CS-97-03, University of Maryland Baltimore County, 1997.

[25] C. Macal and M. North. Introductory tutorial: Agent-based modeling and simulation. In *Proceedings of the 2014 Winter Simulation Conference*, pages 6–20. IEEE Press, 2014.

[26] C. M. Macal and M. J. North. Tutorial on agent-based modeling and simulation. In *Simulation Conference, 2005 Proceedings of the Winter*, pages 14–pp. IEEE, 2005.

[27] D. MacGarty and D. M. Nott. *Disaster Medicine: A Case Based Approach*. Springer Science & Business Media, 2014.

[28] M. W. Maier. Architecting principles for systems-of-systems. In *INCOSE International Symposium*, volume 6, pages 565–573. Wiley Online Library, 1996.

[29] E. J. Mishan and E. Quah. *Cost-benefit analysis*. Routledge, 2007.

[30] I. Monarch and J. Wessel. Autonomy and interoperability in system of systems requirements development. In *16th IEEE International Symposium on Software Reliability Engineering, Chicago, IL*, 2005.

[31] 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. *ACM Computing Surveys (CSUR)*, 48(2):1–41, Sept. 2015.

[32] H. S. Nwana, D. T. Ndumu, L. C. Lee, and J. C. Collis. Zeus: a toolkit and approach for building distributed multi-agent systems. In *Proceedings of the third annual conference on Autonomous Agents*, pages 360–361. ACM, 1999.

[33] S. Poslad. Specifying protocols for multi-agent systems interaction. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 2(4):15, 2007.

[34] A. S. Rao and M. P. Georgeff. Modeling rational agents within a bdi-architecture. *KR*, 91:473–484, 1991.

[35] A. S. Rao, M. P. Georgeff, et al. Bdi agents: From theory to practice. In *ICMAS*, volume 95, pages 312–319, 1995.

[36] D. Seo, D. Shin, Y.-M. Baek, J. Song, W. Yun, J. Kim, E. Jee, and D.-H. Bae. Modeling and verification for different types of system of systems using prism. In *Software Engineering for Systems-of-Systems (SESoS), 2016 IEEE/ACM 4th International Workshop on*, pages 12–18. IEEE, 2016.

[37] SESoS. SIMVA-SoS. https://github.com/SESoS/SIMVA-SoS, 2017.

[38] M. Waters, L. Padgham, and S. Sardina. Evaluating coverage based intention selection. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 957–964. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

Acknowledgments in Korean

처음 서울을 떠나 홀로 낯선 대전에, 그리고 대학원이라는 생소한 곳에 왔습니다. 석사과정으로 2년을 보내며 이전에는 알지 못했던 많은 것들을 알게 되고 공부하고 배웠습니다. 학문과 연구라는 것에 동경을 갖고 당차게 시작했지만 생각보다 어렵고 힘든 길임을 깨닫기도 했습니다. 이 시간들을 보내면서 제가 많은 것을 배울 수 있었던 이유, 버틸 수 있었던 이유, 성장할 수 있었던 이유는 많은 분들의 애정어린 도움 덕분이 아닌가 생각합니다. 그분들께 마음을 담아 깊은 감사의 말씀을 전하고 싶습니다.

먼저 긴 기간 동안 지도해주시고 용기를 북돋아주신 배두환 교수님께 감사드립니다. 연구를 어떤 자세와 마음가짐으로 해야하는지, 어떻게 해야하는지뿐만 아니라 제 자신으로서 어떻게 살아야할지에 대해서도 나눈 이야기들과 해주신 말씀들 감사합니다. 바쁘신 가운데 시간 내어 심사위원을 맡아주시고 응원과 좋은 조언의 말씀 주신 고인영 교수님과 유신 교수님께도 감사드립니다. 연구적으로 이야기를 나눌 때마다 예리한 질문으로 제가 놓치는 부분을 짚어주신 지은경 박사님께도 감사드립니다.

두 해 동안 함께 생활 했던 연구실 선후배분들께도 감사드립니다. 자전거도 함께 타고 연구실의 어른 역할을 해주셨던 영택 선배님, 첫 석사 세미나 논문을 통해 논문은 어떻게 읽고 이해하고 전달해야하는지 가르쳐주신 지훈 선배에게 감사드립니다. 시크하면서도 엉뚱한 매력을 가진 원경 선배, 먼저 졸업한 선배이자 내 생의 첫 룸메인 준호에게도 고맙습니다. 선배처럼 친구처럼 1년을 함께 보내며 의지도 되고 덕분에 연구실 생활에 잘 적응할 수 있었던 것 같습니다. 짧지도 길지도 않은 1년 동안 연구실에 큰 임팩트를 남긴 치우 형에게도 고맙습니다.

가까이에서 연구실 생활과 연구에서 많은 부분 이끌어주시고 도와주신 선배분들께도 감사합니다. 같은 세부 연구뿐 아니라 뮤테이션과 졸업 연구까지 많은 부분에서 시간 할애하여 가르쳐주고 함께 고민해준 동환이 형, 멘탈이 흔들릴 때마다 고민 들어주시고 응원해주시고 박사 과정다운 크리티컬한 코멘트 주신 영민이형과 지영 누나에게 감사합니다. 비단 연구뿐 아니라 진로와 삶까지도 함께 치열하게 고민했던 시간들이 있었기에 여기까지 온 것 같습니다. 같이 졸업 준비를 하며 정신줄로 함께 줄넘기 한 유림 누나에게도 고맙습니다. 어느 한 사람이 고뇌와 우울에 빠졌을 때 서로 다독이고 응원한 시간들이 기억납니다. 여기까지 온 만큼, 이후의 삶도 유림 누나만의 방법으로 멋지게 살아낼 것이라 믿습니다. Thank you, Zele. I will remember your open and warm heart that I have felt for a half year.

학업뿐 아니라 대학원 생활의 소소한 일상들을 함께 보낸 동기들 병수 형, 하영이 형, 승현이 형, 태욱이 형, 상규, 재원이 연이에게도 고맙습니다. 졸업하는 사람, 여남은 공부를 더 하는 사람 등 각자의 길에서 또 멋지게 해낼 것이라 기대합니다. 타지에 아들을 보내고 묵묵히 기도해주시고 응원해주신 부모님께도 감사합니다. 평소에는 애 같아도 가끔은 저보다 어른 같기도 한 동생에게도 고맙습니다. 긴 기간 동안 실습하느라 너도 수고 많았다. 힘든 시간 함께 해준 민아에게도 고맙습니다. 미안하고 고마운 순간들이 참 많습니다. 힘들다 징징거리는 순간마다 알게 모르게 들어주시고, 크고 작게 힘내라 응원해주신 많은 분들께도 감사드립니다. 많은 사람들의 도움과 애정 기억하고 앞으로 저도 다른 이들에게 애정과 도움을 줄 수 있는 사람이 되겠습니다.

2017. 12, 진민규

Curriculum Vitae in Korean

이 름: 진 민 규

생 년 월 일: 1992년 08월 13일

학 력

2008. 3. – 2011. 2. 염광고등학교

2011. 3. – 2016. 2. 숭실대학교 컴퓨터학부 (학사)

2016. 3. – 2018. 2. 한국과학기술원 전산학부 (석사)

경 력

2016. 9. – 2017. 8. 한국과학기술원 전산학부 조교

연 구 업 적

1. **Mingyu Jin**, Donghwan Shin, and Doo-Hwan Bae. "ABC$^+$: Extended Action-Benefit-Cost Modeling with Knowledge-based Decision-Making and Interaction Model for System of Systems Verification." *Proceedings of the Symposium on Applied Computing*. ACM, 2018 (to be published).
2. Jiyoung Song, Young-Min Baek, **Mingyu Jin**, Eunkyoung Jee and Doo-Hwan Bae. "SoS GaP slicer: SoS Goal and PRISM Models for Change-Responsive Verification of SoS." *Software Engineering Conference (APSEC), 2016 23rd Asia-Pacific*. IEEE, 2017.
3. **진민규**, 신동환, 김준호, 배두환. 시스템 오브 시스템즈 수준의 목표 달성 검증 도구. 2017 한국정보과학회 학술발표논문집, pp. 552-554. June, 2017.