Policy-Guided Collaboration for Enhancing System of Systems Goal Achievement

Zelalem Mihret, Jakob Axelsson
School of Innovation Design and Engineering
Mälardalen University
Västerås, Sweden
(zelalem.mihret.belay, jakob.axelsson)@mdu.se

Eunkyoung Jee, Doo-Hwan Bae
School of Computing
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
(ekjee, bae)@se.kaist.ac.kr

***Abstract*—In its broadest sense, collaboration can be defined as a united effort to accomplish activities that ultimately lead to the achievement of shared objectives. This requires the sharing of information, planning, risks, and rewards between collaborating entities to a certain extent. In the case of systems of systems (SoS) where diverse and autonomous constituent systems (CS) interact to achieve shared goals, collaboration presents multifaceted challenges resulting from its distinct characteristics, including interconnectedness among the CS, heterogeneity, scalability concerns, dynamic environments, emergent behavior, stakeholder alignment challenges, and intricate decision-making processes. In order to enhance the achievement of the SoS goals, we propose a policy-guided collaboration approach. In this regard, we establish a learning-based policy generation process with the goal of guiding the decision-making behavior of CS. The practicality of the proposed approach is illustrated through a focused analysis of a high-rise building fire incident response system. Based on simulation results, the proposed approach performs better than the conventional approach in terms of SoS specific task completion time, performance with changes in simulation inputs, and efficiency. We also conducted a sensitivity analysis of task completion time by varying independent decision variables such as the number of CS instances and the size of collaborative tasks.**

***Index Terms*—Policy-guided collaboration, collaborative decision making, policy generation, system of systems engineering**

I. Introduction

The definitions and methods of a collaboration can vary across disciplines and even within the same field. Of the many possible reasons, the differences could be due to means-end point of views. Some focus on the process aspect of a collaboration, others on its effects and results. In the most general sense, a collaboration can be viewed as a unified effort to accomplish activities that eventually contribute to shared goals. This, to some extent, requires the sharing of information, planning, risks, and rewards between collaborating entities. Particularly, the systems of systems (SoS) paradigm, characterized by the integration of diverse and autonomous constituent systems (CS), presents multifaceted challenges to collaboration towards achieving shared goals.

In this paper, we focus on heterogeneous (in terms of their capabilities) systems' collaboration challenges and the solution approach towards achieving a shared goal. It has become a necessity to many organizations to recognize the needs and requirements of and work together with other organizations at different level. For example, the success of the air transport system is no longer dependent on only its parts. It is affected by the efficiency and effectiveness of subway systems, train ways, bus systems, taxi systems, booking systems, hotel services, financial services, airport systems, cargo and delivery systems, and many more. The ecosystem where an organization exists influence the effectiveness of its operations in different ways. We view such complex problem from the perspective of system of systems engineering, frame the solution as a guided collaboration to a dynamically changing operational environment, and propose a learning-based collaboration technique.

The concept of SoS does not fit neatly into a single common definition, but most often SoS is said to be composed of a number of CS working together to achieve certain shared goals [1]–[3]. The shared goals can only be achieved by a systematic collaboration. From SoS context, some CS involvement can be temporary and may only exist until the shared goals are achieved [4]. Guiding the decision making behavior of the CS in the direction that creates favorable global outcomes for the shared goal is one of the challenges that is still not adequately addressed in the system of systems engineering (SoSE) [5].

From the SoS context, collaboration is defined as the utilization of the capabilities inherent in CS to collectively achieve common objectives [6], [7]. These capabilities of the CS serve as the means through which the SoS mission is accomplished by executing tasks specific to collaboration, known as collaborative tasks. In essence, a CS within an SoS is generally perceived as an autonomous decision-making entity with the capability to execute and transform states related to collaborative tasks [1], [2], [8], [9]. Collaborations in SoS context have generally received limited attention, particularly concerning the development and utilization of techniques aimed at achieving shared goals. This research holds a degree of uniqueness in its focus on this particular aspect.

From a structural perspective, a policy contains a set of conditions, and the respective set of actions aiming to govern the behavior of entities [10]–[13]. In another dimension, an operational perspective; a policy has a specifier with specific

This research was partly supported by KKS, grant no. 2020-0230 and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2022R1I1A1A01072004).

goals which can be embedded in the set of conditions and a performer with capabilities to execute the set of actions. We use policies with emphasis on the latter perspective, and our policies aim at effective collaboration towards achieving the shared goals by dictating the decision-making behavior of the CS. This approach somehow addresses the *downward causation* challenge to some extent, at least within the collaboration time range to a specific goal [14]. This can be achieved by analysing decision making patterns of the CS and leveraging specific learning technique that incorporates each CS response behavior to policies.

The primary contributions of this study cover two key aspects. First, it introduces a policy-guided collaboration approach that improves the achievement of SoS goals. Second, it presents the implementation of dynamic policies to guide the decision-making behavior of the CS.

This paper is structured as follows. In Section II, we review the existing literature and state of the art for collaborations in the SoS context. The proposed approach is discussed in detail in Section III. Section IV presents a case study, namely the high-rise building fire incident response system, to demonstrate the practical utility of the proposed approach. In Section V, we discuss the challenges, opportunities and potential impact of policies in SoS collaborations. Finally, the conclusion and directions for future research will be presented in Section VI.

II. Related Work

Widely used models for cooperation in multi-agent systems (MAS)-based systems depend on a static framework and are hierarchical. Such approaches assume that agents are designed to fulfill common goals or that agents' behaviors are known and can be manipulated externally [15]–[18]. However, these prevalent models of cooperation exhibit limitations when applied to the dynamics of SoS. Their reliance on static frameworks and hierarchical structures is restrictive within the evolving and interconnected nature of SoS.

In the case of service oriented architecture (SOA), based on time of collaboration establishment and service discovery, there are three kinds of architecture: static, dynamic, and dynamic collaboration [19]. Typically, services do not have operational and managerial independence characteristics. This implies that services lack the ability to make autonomous decisions by interacting with or reacting to the environment in which they are situated. Services are designed to be part of a system, not a system itself. Hence, they are not operatable by themselves. They first need to be part of certain business logic to be realized, unlike CS.

Negotiation is a very essential concept in establishing a collaboration between autonomous systems. An incentive-based negotiation model is proposed to acquire CS for the commitment of SoS-level goals [20]. The main challenge in the incentive-based negotiation approach is the characteristics of the negotiation token, that is, the means of negotiating with various numbers and types of CS. Both assumptions, such as using a common negotiation token and a specific token for each particular case, have strong viability (practicality) challenges. Coordinated adaptation is the main focus of the research conducted by Watzoldt et al. [21]. Different coordination schemes that should be considered in the modeling language of collaborative systems with dedicated adaptation activities in the form of MAPE-K are identified. Collaboration, as used in Watzoldt et al. work, aims to fulfill each individual's goal and not to achieve a global goal that requires the individual's contributions.

With regard to using policies for collaboration purposes, Cunnington et al. proposed policies for guiding systems' behavior to adapt to new contexts [15]. It considers and uses policies as a means of self-adaptation to new environments. The intentions and decisions of other actors are not considered as important factors in generating the new policies. Omid's work addresses collaboration challenges between different level autonomic managers in policy-based management systems [22]. A hierarchical model is proposed that relies on message-based communication to organize autonomic managers. Collaboration focuses on self-optimization with the use of expectation policies. Each autonomic manager operates on the basis of a set of policies stored in a central repository. It does not take into account collaborative behaviors that change dynamically. It is also firmly bounded to a static architecture. With the objective of managing dynamically evolving SOA, Tsai et al. propose a policy generation technique to generate policies when collaboration protocols are just established, at run-time [19]. Collaboration policies are considered as the composition of local policies applicable for workflows generated at run-time. Such policies are limited and can only be used to enforce the common interests of participating parties.

III. Collaboration Approaches for SoS

This section focuses mainly on the collaboration approach proposed for SoS, presented in two subsections. The overview subsection discusses the existing generative policy architecture, and the subsequent subsection presents the proposed policy-guided collaboration approach.

A. Overview

The generic policy derivation (generation) architecture assumes two role-specific systems working in collaboration: a *management system* that produces initial policies and *managed devices* that generate policies based on the initial policies to cope with their environment [23].

The architecture focuses mainly on enabling *managed devices* to exercise autonomy in generating their own policies for their operational circumstances. Instead of the traditional policy-based management approach where the policy refining (PRF) and the policy decision point (PDP) reside in the *management system* and the only policy enforcement point (PEP) in the *managed devices*, the proposed architecture pushes the PDP and PRF further down to the *managed devices*. This enables *managed devices* to generate and apply policies at their discretion. *Managed devices* generate policies to operate in new environments where there are new expectations. However,

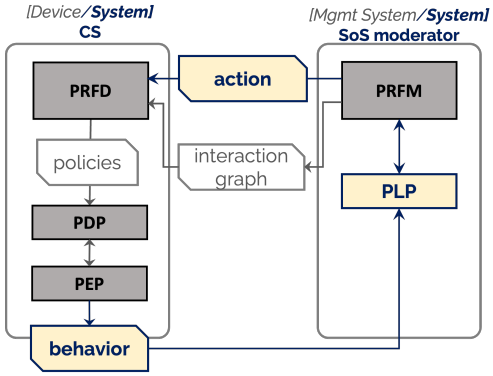

Fig. 1. Generative policy architecture (Extended) [23]. The extended architecture considers two separate systems for the policy generation and policy enforcement, namely the moderator and CSs respectively.

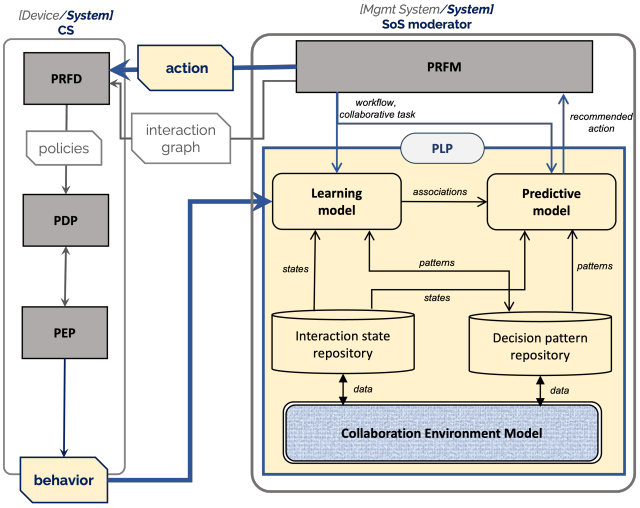

Fig. 2. PLP overall diagram for policy-guided collaboration.

it is not clear how this approach can be used for new requirements that necessitates collaboration between *managed devices* for a common goal.

The *managed device*, in the newly proposed architecture, has three components to process policy information, as shown in Fig. 1 in gray: (1) policy refinement (PRFD), policy decision point (PDP), and policy enforcement point (PEP). PRFD identifies entities and attributes that can be affected by the *managed device* actions based on the interaction graph the *management systems* prescribes. It analyzes the device state and generates a plan of actions along with a list of policies that will be processed further by PDP. PDP is responsible for making decisions on which policies to apply for which plan of actions based on analysis of historical data, contextual information, and interaction attributes. It generates policy decisions that the PEP shall consider. PEP consults PDP for actions, and enforces those actions that enable the device to exist and adapt in its environment by ensuring its priorities.

As a result of PEP, certain effects are induced into the environment, which can be observed as the behavior of the *managed device*. This could be decision to continue providing already started capabilities, initiating new services, halting interactions with some other specific *managed devices*, or denying access to local resources. The management system policy refinement component (PRFM) can be optimized by learning from the observed behavior. However, the existing generative policy architecture relies only on human direct input to make revision and improvement on PRFM.

We extend the generative policy architecture by introducing a policy learning point (PLP). PLP is a component where behaviors exhibited by the *managed device* system as a result of the recommended actions are accumulated, associated and learned to improve PRFM action recommendations.

PLP is an essential component to influence decision making behavior towards the shared goals in an environment where both the *managed devices* and *management systems* are autonomous systems, a typical case in SoS. In such cases, each system can have its own priorities, and there is less hierarchical authority to enforce the interest of one over the other. Instead, by learning the decision-making behavior, it is possible to influence how one behaves during collaboration.

B. Policy-Guided Collaboration Approach

From an operational perspective, a policy has a specifier with specific goals that can be embedded in the set of conditions, and a performer with the capabilities to execute the set of actions. Hence, policies can be considered as information mechanism between two independently managed systems, for example SoS moderator and CS.

As discussed in the related work section, the static policy specification, the listing of a fixed set of conditions, and the respective set of actions are inefficient to be used to achieve effective collaboration in the SoS context, discussed in detail in [24]. Towards to dynamic policy specification, we proposed integrating the PLP component into the generative policy architecture, as shown in Fig. 2. This approach offers a dual benefit: the ability to specify policies dynamically and enables collaboration guided by policies. This means that the conditions within the policy can be adjusted or entirely new conditions and actions can be constructed by learning from the CS responses. This learning process, combined with a predictive model, can also be used to enhance SoS goal achievement.

The learning model is where the moderator uses interaction experiences with the CS to improve its policy and value functions over time. It takes inputs from CS responses for recommended actions, workflow and collaborative tasks from PRFM, and interaction state and decision patterns from the collaboration environment. Different learning techniques, such as reinforcement learning, can be considered. The output of the learning model updates the decision pattern repository with new learned patterns and generates new or modified associations of conditions and actions of policies. The CS responses to policies depend on their decision-making behavior, especially when determining whether they should make their capabilities available for specific collaborative tasks or not. There is no

one-size-fits-all decision-making strategy, and there are no common metrics to measure how recommended actions will affect the CS decision-making behavior. Heuristic approaches can be used to satisfy such requirements for modeling and simulation purposes.

The predictive model uses historical data and statistical or machine learning techniques to make predictions about actions that a candidate CS is likely to enforce. It generates a recommended set of actions that the PRFM can communicate to CS based on the new associations learned. It considers interaction states and patterns from interaction states and decision pattern repository, respectively. Techniques such as stochastic predictive (based on occurrence frequency of similar decisions), case-based reasoning, drawing lessons from third-party's observations, or machine learning can be used to predict the recommended set of actions.

The collaboration environment model contains simulated SoS scenarios in the intended domain specific environment. The states of the interactions and the decision patterns from the simulated scenario serve as a data source for the learning and predictive models. It is continuously updated to reflect new or revised associations between states, decisions and policies.

For experimentation purposes, we use the reinforcement learning (RL) technique and employ the stochastic technique based on statistical analysis of decision patterns for the learning and prediction model, respectively. The RL views fit to describe the SoS moderator and CS interactions to achieve the common goals. The SoS moderator has a specific goal to achieve by utilizing the CS capabilities, and CS show their interest by approving/declining the collaboration requests. The state-action pairs – the building block of RL, can be explained as the collaboration tasks completion level and the actions the SoS moderator can take (or the actions the SoS moderator recommends via policies for the CS). These actions are finite. Any of the actions, if approved by any of the CS, will change the state of course of actions of the respective CS capability. In turn, the collaborative task state will be changed, and the SoS moderator can learn how good the action was by observing the state changes (evaluating how good the state changes are towards achieving the common goals).

One essential thing to note is when using RL for agents, as commonly called, such as robots, games (game playing), or self-driving cars, the notion of a policy is associated with the strategy the agent learns from the RL processes – that determines what action to take when the agent is in a specific state. In the SoS case, there are two aspects of the policy in the RL processes, (1) we consider the moderator actions as the policies recommended for the CS, while still (2) the moderator learns its internal policies from the RL processes. For the former case, unlike the other domains, the moderator actions did not directly affect the collaboration. Rather, it recommends policies to the CS. If the recommended policies are enforced by the CS, the collaboration state will change accordingly, and the SoS moderator can observe the state changes. Our focus is on these particular policies that the CS are expected to enforce. For the internal policies, the moderator learns simply by taking the maximum return from choosing the specific action on the current state.

IV. Emergency response SoS case study

We examine a case study and carry out a simulation-driven experiment to demonstrate the implementation of the proposed approach. The primary objective of this analysis is to ascertain the comparative benefits of the proposed approach in contrast to conventional methods, specifically in achieving collaborative objectives by leveraging diverse capabilities sourced from a range of CS. Additionally, this assessment aims to validate the practicality and feasibility of the proposed approach.

We develop a policy-guided collaboration simulator based on the Agent-based Modeling & Simulation (ABMS) methodology [25], using the Java agent development framework (JADE) [26]. ABMS can represent complex systems, such as SoS, with minimal knowledge of the parameter values or without fully knowing the optimal parameter states to describe the real-world environment. It can be used to model the CS decision-making with regard to planning, vulnerability analysis, interaction and collaboration facilities, and real-time response behavior. In a previous work, we demonstrated an action recommendation strategy taking into account SET-based properties using ABMS [27].

A. Emergency Response SoS

Emergency response systems, such as high-rise building fire incident response systems, use a number of heterogeneous systems to manage and control incidents that could be overwhelming to a single system. Often, the participant systems are not designed to operate in a hierarchical organization, *i.e.* manager-subordinate mode. Participant systems have the autonomy to decide whether to participate in collaborative tasks and contribute their capabilities or not.

The emergency response system is recognized as a collaborative SoS type [28], [29]. Each participant system has the freedom to join, leave or stay in the collaboration, therefore, they are fully independent in their managerial and operational aspects. They also readily exchange information concerning their capabilities and task accomplishment status – which satisfies geographic distribution characteristics. Effectiveness of the common goals, for example, rescuing as many victims at risk as possible, is determined based on emergent behaviors that result in the evolutionary development of interactions among constituting systems.

B. Simulation Design Overview

We model the SoS scenario based on ABMS. The SoS scenario tells the story of interactions between the CS and the moderator to achieve the shared goals. It aims to perform collaborative tasks using CS capabilities guided by policies. We abstract the following autonomous systems as agents in our simulation:

- CS - A CS is managerially and operationally independent system that has one or more capabilities to execute and transform certain collaborative tasks specified for a SoS

purpose. It has the freedom to choose when and how its capabilities should be activated, stopped, suspended, or resumed. For example, an ambulance is a CS in the case of emergency response collaborations. One of the capabilities of an ambulance is to transport victims from the incident scene to selected hospitals.

- Moderator - The moderator is a dedicated system that acts on behalf of the SoS interests. It facilitates the CS decisions towards achieving the SoS purpose. The moderator is in charge of generating policies, dispatching collaborative tasks and policies, monitoring collaborative task completion progresses, and observing and taking measure of selected properties of RTAM.
- Real-time Activity Monitor (RTAM) - RTAM embodies distinct simulation attributes and behaviors that describe and mimic collaboration in the real environment. Notably, specific properties define its uniqueness, such as, set of selected collaborative tasks and their completion levels, set of selected CS (constellation [4]) to provide capabilities, relationships between collaborative tasks, and the relationship between collaborative tasks and set of capabilities. The mechanisms governing and overseeing the behavior of these properties, as well as the monitoring of interactions with both the moderator and CS, are singularly tailored to each collaboration. In our simulation, we conceive of a reactive system that functions as an agent that dynamically represents a real-time collaboration, denoting it as the Real-time Activity Monitor.

The simulation design revolves around three primary agent classes: the CS, the moderator, and the RTAM. Within this design, the moderator takes on the responsibility of generating policies that dictate the collaborative course of action. These policies are then communicated to the CS, providing them with guidelines on how to allocate their respective capabilities towards the collaborative tasks at hand.

The decision-making process of a CS can be influenced by these policies, as they have partial and incomplete decision information about the collaboration. Importantly, these decisions made by the CS have a direct impact on the properties monitored by the RTAM. These monitored properties encapsulate various aspects such as the completion status of collaborative tasks, the involvement of CS capabilities, and the overall progression towards achieving the collaborative goal.

The moderator, in turn, observes and evaluates these effects as presented by the RTAM properties. This observation allows the moderator to gauge the effectiveness of the policies implemented, assessing how they influence and steer the collaborative efforts toward the ultimate goal. The intricate interactions among these agents and their consequential impact on the collaborative process are visually represented in the accompanying Fig. 3.

C. Policy-Guided Collaboration in Simulation

The simulation-based policy-guided collaboration has two phases. The first phase deals with generating policies. This involves systematic learning and prediction techniques, as outlined in the approach section. We use the RL technique for the moderator's decision-making strategy. Learning in RL occurs through trial and error [30]–[32]. The moderator evaluates the effectiveness of the policy based on the behavior exhibited by the CS. Regarding the prediction of policies, we employ the stochastic technique based on statistical analysis of decision patterns. The second phase is policy-guided collaboration. This phase uses policies generated from the first phase. When the moderator decides to guide the decision-making behavior of the CS, it recommends a set of actions that reinforce the probability of achieving the SoS goals. The two phases are strongly coupled and are codependent on each other.

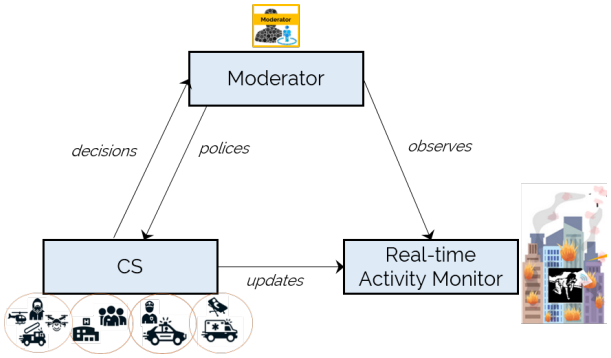

Fig. 3. ABMS-based simulation design for emergency response SoS.

We use public information about the Grenfell Tower fire incident to establish initial conditions for simulation purposes. The sources of the public information and data are various reports including inquires and hearings, police investigation reports and media outlets about the incident and the emergency response operations [33]. According to investigation reports, about 250 firefighters, 9 search drones, 30 emergency vehicles, and 6 hospitals participated in the response to the Grenfell tower fire incident. One of the shared goals that binds all the participating response systems was *saving as many trapped individuals as possible*. There were 80 deaths and 70 injuries, a total of 150 homes destroyed, burned about 60 hours [34]. We use the data to make estimates of values for the simulation scenarios, define interactions and data for the interactions, define CS and their capabilities, and define type of policies in the collaboration.

We consider first responders to the established hypothetical fire incident that includes police, fire fighters, rescue, ambulance, and hospitals to constitute and arrange for the SoS of emergency response. In addition, victims are also a CS. The common goal is evacuating and rescuing more number of individuals at risk with in a reduced time. The selected capabilities of each CS considered for the simulation purpose are shown in Table I.

We run several hundred simulation by varying scenario configurations. A configuration contains the number of instances of each CS, the capabilities that each CS commits to collaborative tasks, and a probabilistic estimate of the approval

TABLE I
CSs AND THEIR CAPABILITIES CONSIDERED IN THE SIMULATION

CSs	Capabilities
Police	Coordinating, Establishing cordons, Preserving order
Fire fighter	Extinguish fire, Search victims, Rescuing victims
Rescue service	Searching victims, Rescuing victims
Ambulance	First aids, Triaging, Transport
Hospital	Clinical treatment
Victims	Signal location and status

TABLE II
DIVERSE SCENARIO CONFIGURATIONS

	Police	Fire fighter	Rescue service	Ambulance	Hospital
Conf3	1	27	14	22	3
Conf2	1	39	20	27	3
Conf1	2	46	30	28	5

of the recommended actions by the moderator. Table II shows a handful scenario configurations only to illustrate and make a comparative analysis between the proposed and baseline approach.

We investigate effectiveness of the proposed approach in terms of achieving a reduced time for completing the collaborative tasks. For comparison purpose, we implemented the commonly practiced collaboration approach in distributed and cooperative environment, as described in [35]. We select this approach as a base line because it deals with autonomous agents (or managed systems) and assumes unpredictable availability of constituents' capabilities. The compiled simulation result is shown in Fig. 4 and Fig. 5.

D. Simulation Results

As shown in the summarized simulation results, Fig. 4 and Fig. 5, the first observation is that the completion time of the proposed approach is shorter compared to the baseline approach, especially as the number of victims increases. The case is not the same when the number of victims is smaller, for obvious reason that the collaborative tasks to rescue small number of victims got completed in much shorter time.

The red box shows the completion time of multiple simulations conducted for the same number of victims, varying the simulation configuration attributes, including the number of instances of heterogeneous CS. In the case of Fig. 4, the minimum is around 0.6 hours, and the maximum is 1.8 hours to complete the collaborative tasks for the victim number of thirty (30). The time range is largely distributed, compared to the case in Fig. 5 which is about 2.2 to 2.6 hours. This indicates that the proposed approach is more sensitive to the changes in the number of instances of the CS, despite it completes the collaborative task in a reduced time. The reason for this largely distributed completion time is due to more interactions with the moderator when there are more CS instances. PLP adds communication overhead and computational cost for knowledge association; therefore, the number of CS in the collaboration is expected to affect the performance of the proposed approach.

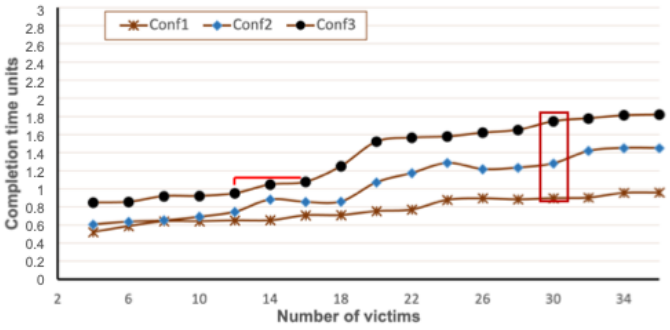

Fig. 4. Proposed approach completion time across diverse scenarios.

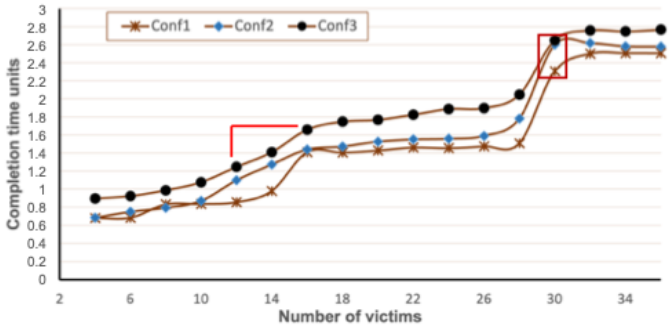

Fig. 5. Baseline approach completion time across diverse scenarios.

The red bar-line shows change in completion time when number of victims changes. For example, a change in victim number from 12 to 16 causes a small change in the maximum completion time in the case of Fig. 4, but a much larger change in the case of Fig. 5. This indicates that the baseline approach is much more sensitive to changes in task size compared to the proposed approach.

Based on the observation, as expected the baseline approach exhibits a linear increment which is closely the same as the sum of the increment in number of victims *i.e.* a 200% increment. The proposed approach still can manage the scaled phenomenon, only costing close to a 100% increase in the completion time of collaborative tasks. Again, the baseline approach is observed to show a drift increase in task completion time in some of the scenario configurations, which is about 4 times the proposed approach (more than a 400% increase).

As can be seen in the two simulation results, the proposed approach manages the scaled environment costing less in completion time. The substantial performance advantage of the proposed approach over the baseline approach can be attributed to the PLP component introduced to incorporate the decision-making behavior of the CS in the dynamic policy specification.

V. DISCUSSION

Effective and efficient collaboration in the SoS context is super complex for several reasons. For example, it deals with unpredictable CS decision-making behavior, involves orchestration of heterogeneous capabilities, exhibits evolutionary

development on different levels, and goal fulfilment is highly dependent on emergent behaviors. We use policies to enforce guiding constraints on collaborating CS to ensure the execution of services, workflows, and applications that enhance the achievement of SoS purpose. In the following subsections, we present the discussion points.

A. Limitations of Static Policies for SoS

The limitations of static policy specification for SoS collaboration are obvious. Essentially, a static policy primarily functions as a regulator, overseeing and directing the behavior of the system to operate optimally in alignment with its predefined design objectives. It operates by outlining a set of conditions and their corresponding actions, serving as rigid boundaries that confine the behavior of the system. Anything beyond these predefined parameters is typically considered undesired system behavior. This static approach can directly contradict the essence of a SoS, as emergent behaviors play a pivotal role in fulfilling its purpose.

The static nature of these policies, where conditions and corresponding actions are pre-established, proves restrictive as emergent behaviors, crucial for SoS functionality, might not be encompassed within these predefined boundaries. In the case of SoS, the conditions and actions are selectively determined based on available information and the specific mission's nature.

Hence, there arises a need to explore the creation of global policies capable of enforcing diverse global constraints or properties, considering the newly established interaction behaviors within the SoS.

B. Dynamic Policies and Policy-Guided Collaboration

One way to address the limitations of the static policy, as presented in this paper, involves integrating the response behavior of the CS through learning models. By incorporating learning and prediction models that forecast future collaborative behaviors, there exists an opportunity to bridge the gap between static policies and the evolving dynamics of collaborative interactions within the SoS.

Directing the CS decision-making behavior to effectively contribute to the overarching purpose of a SoS poses significant challenges, especially when faced with two specific conditions. Firstly, the necessity to recognize a CS autonomy is one of the essential characteristics of a SoS-based analysis. Autonomy is associated with the level of managerial and operational independence *i.e.*, the authority that the system has to manage its resources, and the ability to decide when and how to perform its functions [1], [8]. A global policy can affect a CS decision-making behavior only if the CS has consent to. Our approach tries to ensure the consent of CS to the recommended policies through learning and systematic prediction. With this regard, we expand the policy-based decision-making components of the management system (as described in the policy architecture [23]) by including PLP that learn and associate new knowledge for PRFM.

Secondly, achieving effective policy specification becomes challenging when intricate knowledge about the decision-making factors influencing CS behavior is lacking. The absence of precise insight into these decision-making factors hampers the ability to craft policies that accurately guide CS towards the SoS purpose. This limitation impacts the quality and effectiveness of the policies designed to steer CS behavior, potentially leading to less effective contributions to the overall SoS objectives.

These conditions, the need to maintain the autonomy of the CS, and the absence of detailed knowledge about the decision-making factors of the CS collectively influence the formulation and impact of policy specifications in the context of SoS. Balancing these factors is crucial to devise policies that guide the decision-making behavior of CS effectively without compromising their autonomy while navigating the complex landscape of an evolving SoS.

C. ABMS-Based Evaluation

We use ABMS to verify the efficacy of the proposed approach and perform a comparative analysis with the effectiveness of the baseline approach. ABMS is useful in two ways: On the one hand, it enables investigations of how decisions of the CS change the SoS behavior from the bottom-up perspective. On the other hand, a moderator can alter scenario configurations to improve the behavior of a system from a top-down perspective. One of the application strengths of ABMS that helps to make a SoS analysis rigorous is that it most likely reveals unanticipated emergent behavior due to its bottom-up approach. The emergent behavior observed is clearly dependent upon the implementation of the CS as agent; therefore, determining the correct level of fidelity and identifying which system characteristics to model within each CS is key and is, in itself, a complex activity. ABMS also provides highly scalable modeling and analysis that can be be constructed with simplified agent behaviors; and modularity of agents enables extension of behavioral complexity if required [36].

D. Dynamic Policy Challenges

A dynamic policy is shown to be more effective compared to a static policy in achieving SoS collaboration goals. However, it is also less predictable for CS, as it may already have allocated resources when policy modifications are made or new ones are introduced. This increases the risks for CS and could make them more hesitant to join or exit a constellation due to possible policy changes. A topic such as this merits further research, for example, how to maintain CS engagement in SoS collaboration during a period of policy changes.

VI. Conclusion

The underlying motivations for the work presented in this paper are the unique characteristics of SoS, and the inherent challenges associated with the characteristics in the design and development of SoS systems. A SoS aims to achieve its goals by harnessing CS capabilities. We explored the use of

policies for guiding the CS decision-making behavior in order to achieve the SoS purpose. The policies in the proposed approach are intended to be used in the real world system. Hence, the development of the proposed approach, with due considerations of the timing of events within the real-world system will result in more accurate and valid policies. As part of the future work, it would be interesting to investigate the virtual prototyping design method for policy-guided SoS collaboration.

REFERENCES

[1] J. Boardman and B. Sauser, "System of systems-the meaning of of," in *2006 IEEE/SMC international conference on system of systems engineering*. IEEE, 2006, pp. 6–pp.

[2] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering: The Journal of the International Council on Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998.

[3] Y. Tian, H. Liu, and J. Huang, "Design space exploration in aircraft conceptual design phase based on system-of-systems simulation," *International Journal of Aeronautical and Space Sciences*, vol. 16, no. 4, pp. 624–635, 2015.

[4] P. Svenson and J. Axelsson, "Should i stay or should i go? how constituent systems decide to join or leave constellations in collaborative sos," in *2021 16th International Conference of System of Systems Engineering (SoSE)*. IEEE, 2021, pp. 179–184.

[5] J. Dahmann, "System of systems pain points," in *INCOSE International Symposium*, vol. 24, no. 1. Wiley Online Library, 2014, pp. 108–121.

[6] H. R. Darabi and M. Mansouri, "The role of competition and collaboration in influencing the level of autonomy and belonging in system of systems," *IEEE Systems Journal*, vol. 7, no. 4, pp. 520–527, 2013.

[7] C. Savur, S. Kumar, S. Arora, T. Hazbar, and F. Sahin, "Hrc-sos: Human robot collaboration experimentation platform as system of systems," in *2019 14th Annual Conference System of Systems Engineering (SoSE)*. IEEE, 2019, pp. 206–211.

[8] 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)*, vol. 48, no. 2, p. 18, 2015.

[9] 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*. IEEE, 2008, pp. 1–7.

[10] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The ponder policy specification language," in *International Workshop on Policies for Distributed Systems and Networks*. Springer, 2001, pp. 18–38.

[11] Z. Ma, Y. Yang, and Y. Wang, "A security policy description language for distributed policy self-management," in *2015 4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering*. Atlantis Press, 2015.

[12] C. S. Shankar, V. Talwar, S. Iyer, Y. Chen, D. S. Milojicic, and R. H. Campbell, "Specification-enhanced policies for automated management of changes in it systems." in *LISA*, vol. 6, 2006, pp. 73–84.

[13] N. Dulay, N. Damianou, E. Lupu, and M. Sloman, "A policy language for the management of distributed agents," in *International Workshop on Agent-Oriented Software Engineering*. Springer, 2001, pp. 84–100.

[14] J. Axelsson, "What systems engineers should know about emergence," in *INCOSE International Symposium*, vol. 32, no. 1. Wiley Online Library, 2022, pp. 1070–1084.

[15] D. Cunnington, I. Manotas, M. Law, G. de Mel, S. Calo, E. Bertino, and A. Russo, "A generative policy model for connected and autonomous vehicles," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 1558–1565.

[16] W. Alshabi, S. Ramaswamy, M. Itmi, and H. Abdulrab, "Coordination, cooperation and conflict resolution in multi-agent systems," in *Innovations and advanced techniques in computer and information sciences and engineering*. Springer, 2007, pp. 495–500.

[17] C. Zhang and D. A. Bell, "Hecodes: a framework for heterogeneous cooperative distributed expert systems," *Data & knowledge engineering*, vol. 6, no. 3, pp. 251–273, 1991.

[18] A. Nasir, Y. Salam, and Y. Saleem, "Multi-level decision making in hierarchical multi-agent robotic search teams," *The Journal of Engineering*, vol. 2016, no. 11, pp. 378–385, 2016.

[19] W.-T. Tsai, Q. Huang, B. Xiao, Y. Chen, and X. Zhou, "Collaboration policy generation in dynamic collaborative soa," in *Eighth International Symposium on Autonomous Decentralized Systems (ISADS'07)*. IEEE, 2007, pp. 33–42.

[20] N. Kilicay-Ergin and C. Dagli, "Incentive-based negotiation model for system of systems acquisition," *Systems Engineering*, vol. 18, no. 3, pp. 310–321, 2015.

[21] S. Wätzoldt and H. Giese, "Modeling collaborations in adaptive systems of systems," in *Proceedings of the 2015 European Conference on Software Architecture Workshops*, 2015, pp. 1–8.

[22] O. Mola, "Collaborative policy-based autonomic management in iaas clouds," 2013.

[23] D. Verma, S. Calo, S. Chakraborty, E. Bertino, C. Williams, J. Tucker, and B. Rivera, "Generative policy model for autonomic management." IEEE, 2017, pp. 1–6.

[24] B. Z. Mihret, E. Jee, Y.-M. Baek, and D.-H. Bae, "A collaboration policy model for system of systems," in *2018 13th Annual Conference on System of Systems Engineering (SoSE)*. IEEE, 2018, pp. 1–8.

[25] G. I. Hawe, G. Coates, D. T. Wilson, and R. S. Crouch, "Agent-based simulation for large-scale emergency response: A survey of usage and implementation," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, pp. 1–51, 2012.

[26] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*. John Wiley & Sons, 2007.

[27] Z. Mihret, E. Jee, and D.-H. Bae, "Simulation-based recommendation generation for heterogeneous systems participating in a collaborative work," in *2022 17th Annual System of Systems Engineering Conference (SOSE)*. IEEE, 2022, pp. 333–338.

[28] S. Perry, J. Holt, R. Payne, C. Ingram, A. Miyazawa, F. O. Hansen, L. D. Couto, S. Hallerstede, A. K. Malmos, J. Iyoda *et al.*, "Report on modelling patterns for sos architectures," *COMPASS Deliverable D*, vol. 32, 2013.

[29] R. J. Payne and J. Bryans, *Modelling the major incident procedure manual: A system of systems case study*. Computer Science, Newcastle University, 2012.

[30] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2042–2062, 2017.

[31] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE circuits and systems magazine*, vol. 9, no. 3, pp. 32–50, 2009.

[32] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[33] Grenfell-Tower, "Grenfell tower-fire safety investigation: The fire protection measures in place on the night of the fire, and conclusions as to: the extent to which they failed to control the spread of fire and smoke; the extent to which they contributed to the speed at which the fire spread." *Fire Safety Engineering*, 2018.

[34] Grenfell-Tower-Inquiry. (2018) Dr barbara lane's expert report. [Online]. Available: https://www.grenfelltowerinquiry.org.uk/evidence/dr-barbara-lanes-expert-report

[35] M. J. DiMario, J. T. Boardman, and B. J. Sauser, "System of systems collaborative formation," *IEEE Systems Journal*, vol. 3, no. 3, pp. 360–368, 2009.

[36] A. Kinder, M. Henshaw, and C. Siemieniuch, "System of systems modelling and simulation–an outlook and open issues," *International Journal of System of Systems Engineering*, vol. 5, no. 2, pp. 150–192, 2014.