THE HOLON PROGRAMMING MODEL FOR HETEROGENEOUS AND ADAPTIVE SYSTEM OF SYSTEMS *

Muhammad Ashfaq
University of Jyväskylä, Finland
muhammad.m.ashfaq@jyu.fi

Ahmed R. Sadik
Honda Research Institute Europe
ahmed.sadik@honda-ri.de

Tommi Mikkonen
University of Jyväskylä, Finland
tommi.j.mikkonen@jyu.fi

Muhammad Waseem
University of Jyväskylä, Finland
muhammad.m.waseem@jyu.fi

Niko Mäkitalo
University of Jyväskylä, Finland
niko.k.makitalo@jyu.fi

ABSTRACT

As digital ecosystems evolve into increasingly complex networks, harnessing their collective potential becomes paramount. This article focuses on developing software for smart ecosystems by programming their underlying System-of-Systems (SoS)—a domain ripe with opportunities and challenges. We propose the Holon Programming Model (HPM), which adds the programmability aspect to the holonic architecture of SoS and will pave the way for more integrated and adaptable systems. Demonstrating the HPM's utility, we show how developers can program diverse behaviours and execute operations within the disaster management ecosystem, aiming to inspire and motivate software developers to explore the potential of SoS-level applications, which are fundamental for smart ecosystems.

Keywords System of Systems · Holonic Architecture · Smart Ecosystems · Systems Programming

1 Introduction

Smart ecosystems represent the next frontier in technology, characterized by interconnected systems communicating and collaborating within intelligent environments. Such ecosystems are distributed, adaptive, and open sociotechnical systems endowed with self-organization, scalability, and sustainability [1]. Moreover, smart ecosystems are dynamic, influenced by external and internal factors, susceptible to periodic disturbances, and their constituents can communicate outside their domain [2].

At the core of these ecosystems lies a complex arrangement known as a System of Systems (SoS), which combines the strengths of their Constituent Systems (CSs) to surpass their capabilities [3]. The SoS in smart ecosystems integrates diverse CSs from various organizations, forming a complex and dynamic entity [4] that presents significant management challenges, especially from the software engineering perspective. The CS architecture within SoS is increasingly converging to facilitate API-like access to different resources without necessitating in-depth knowledge of their internal workings. This trend closely resembles the evolving architectures within the Internet of Things (IoT) and Industry 4.0, which are characterized by the integration of devices (such as sensors and actuators), their capabilities, gateways, and cloud services [5].

The *holonic architecture* [6] provides a robust framework for understanding and modelling the SoS's overall behaviour. This architecture models the heterogeneous CSs as *holons* [7], both autonomous and integrally connected entities, to demonstrate their independent functions and contributions to a more extensive system. The holonic architecture enables CS discoverability [8], dynamic SoS composition [9], and ad-hoc scalability [10]. However, the holonic architecture needs to provide the decision-making capabilities and adaptability to adapt to the changes in the surrounding ecosystem effectively [11]. Furthermore, current software development tools and methods must provide intuitive ways for

* *Preprint submitted to: IEEE Software's Special Issue on Digitalization of Smart Ecosystems*

holistically programming interactions among all the smart ecosystem participants, including systems, devices, robots, and humans. Integrating software on top of the holon composition can address these shortcomings, facilitating the programming of SoS behaviours and enhancing the interaction within the ecosystem.

In this article, we propose the *Holon Programming Model* (HPM), designed to bridge these gaps. HPM adds programmability to the holonic architecture of SoS, providing the necessary tools for seamless integration and cooperation within smart ecosystems.

2 Towards Programmable System of Systems

As the holonic architecture is making CSs increasingly interoperable, the next step should be the creation of programming capabilities specifically designed for managing complex real-world ecosystems. This vision is inspired by the "programmable world" concept [12] that led to the development of programming models for the IoT devices [13]. While these models excel at device-level programming, they lack the scalability necessary to program CSs of SoS.

Building on the insights from the literature, we identify three core programmability challenges of the holonic architecture in SoS:

1) Heterogeneous CSs: The diverse nature of CSs complicates traditional programming approaches, making it challenging for software developers to manage the complexity.

2) System Dynamics: Smart ecosystems are highly dynamic, with CS constantly joining and leaving the SoS.

3) Environmental Interaction: The SoS must be able to take inputs from the environment and respond adaptively. This implies that the holon composition must be capable of sensing environmental changes and initiating appropriate responses.

Addressing these challenges necessitates a programming model tailored to the unique demands of SoS, including multi-system programming, accommodation of heterogeneity and diversity, and an emphasis on software's distributed, highly dynamic, and potentially migratory nature. Furthermore, this model must be reactive to the system dynamics and the surrounding environment by triggering actionable responses. Unfortunately, the current landscape in smart ecosystems lacks such a model, which would enable software developers to create SoS-level applications effortlessly. This gap necessitates a paradigm shift from traditional, sequential, and static development approaches to the ones that define and manage collective interactions among CSs.

The proposed HPM is designed to develop coordinated, proactively and pervasively initiated, multi-system programs efficiently. These programs use collective SoS behaviours as their fundamental components. The HPM enables software developers to specify which changes the systems should jointly observe in the surrounding environment and write joint, system-level behaviours for an observed change.

3 The Holon Programming Model for Developing SoS-level Applications

Figure 1 presents the proposed model for programming the system-to-system and system-to-device level behaviours with holons. It consists of four distinct layers. 1) The **Foundation Layer** consists of holons that model the CS of an ecosystem. 2) These holons merge to form a holon composition at the **Composition Layer**. 3) The **Collaboration Layer** encapsulates the collaborative efforts of holons to achieve specific objectives. 4) The **behavioural Layer** outlines the orchestrated behaviours emerging from these collaborations. The remainder of this section provides a detailed description of each layer.

3.1 Foundation Layer

This layer consists of *holons* that encapsulate the systems and devices within the ecosystem. Each holon offers its encapsulated entity's services or resources as *capabilities* for use by other holons. For example, a holon might possess an NLP capability to translate voice instructions from another holon into service calls. Moreover, each holon maintains a *holon state* that keeps up-to-date information about the status of its resources and services.

A holon also utilizes its capabilities to interact with and produce output to the *external world*. An input from the external digital or physical world is called a *sensation*. Holons observe these sensations and broadcast them to holon collaborations. These observing holons reside in the Foundation Layer or the Composition Layer. For example, a holon might generate a sensation to indicate its desire to initiate or join a holon composition. Similarly, a holon sensing a temperature change is another example.

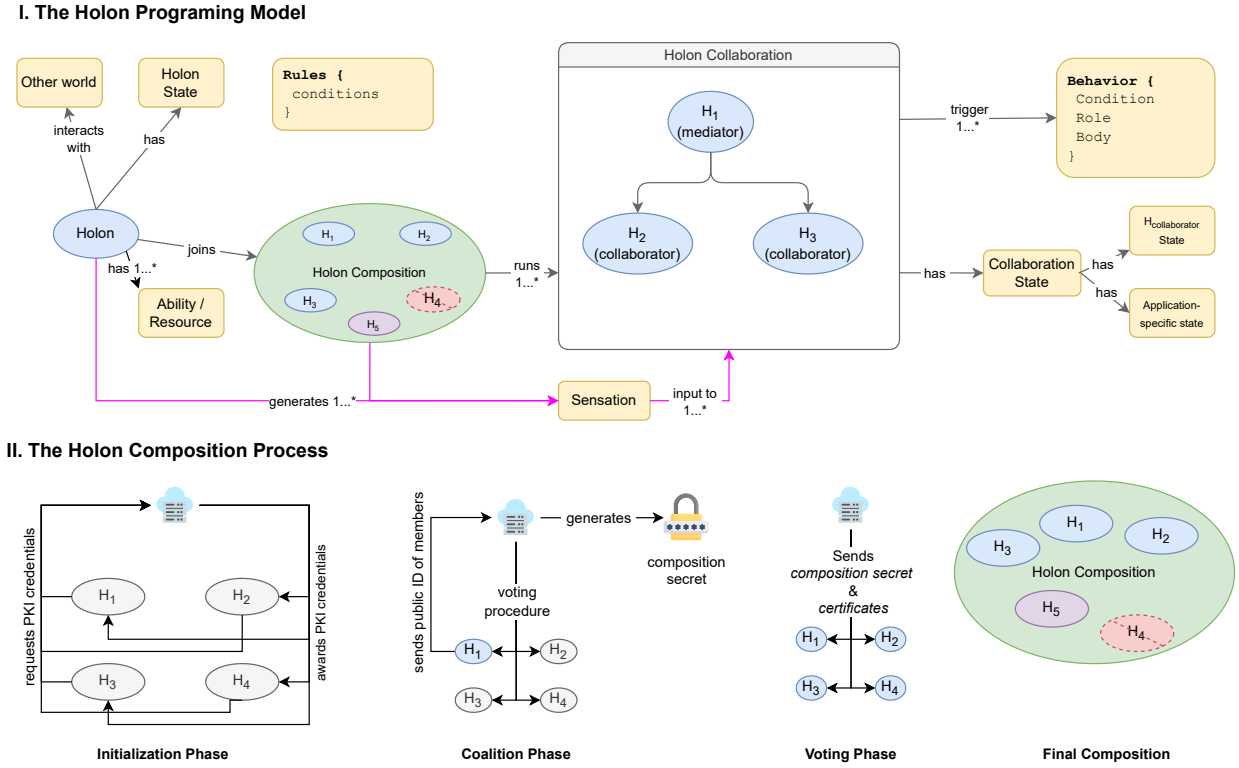

Figure 1: **I)** An illustration of the Holon Programming Model. Each CS is represented through a holon, which makes the holon composition with other holons (SoS). The holons in this composition collaborate to trigger behaviours in response to sensations coming from the external environment. **II)** Detailed process of the Holon Composition.

3.2 Composition Layer

The core principle of holonic architecture is that the holons compose with each other to form higher-order holons. We propose *Holon Composition Framework* (HCFW) to support this composition. In HCFW, holons compose with each other and make a *holon composition* where they collaborate to achieve high-level goals. A holon composition is a holon itself—acting both as a self-sufficient whole in achieving collective objectives and as a part when merging with other compositions. When two or more holon compositions merge, they form a new composition, wherein the holons and behaviours represent the union of the contributing compositions.

The HCFW is conceptually similar to our previously proposed Trusted Device Coalition Framework (TDCFW) [13], which enables secure communication among IoT devices. On the other hand, HCFW supports both system-to-system and system-to-device communication.

Decentralized collaborative ecosystems need rules, mechanisms and processes that the constituent systems must follow. In the HCFW, developers can program such *rules* to specify how the holons should be composed and what *conditions* cause the holon composition to start discovering new holons. The conditions can be related to any physical or digital world processes. For example, the conditions can be related to some physical world measurable quantity (e.g., physical proximity between devices) or an event broadcast by a specific holon (e.g., an emergency signal sent by an autonomous car or a system failure signal sent by a network).

In the HCFW, a secure composition of holons is established by engaging holons in trust negotiations. Part II of Figure 1 details the formation process of this composition. The process consists of three phases: 1) The *initialization phase* requires stable connectivity to a trusted entity (e.g., a cloud service). In this phase, the holons receive certificates from the trusted entity with the corresponding secret and public keys. This ensures direct and secure connectivity with each relevant holon. 2) The *coalition phase* is initialized when a holon wants to compose with other holons. It sends the public identifiers of the potential composition members to the trusted entity. The composition secret is then generated and split among the composition partners. 3) The *voting phase* is initialized by the trusted entity to confirm if the

potential partner holons agree to the composition. After confirmation, the trusted entity delivers the composition secret and the certificate to the holons.

This process leads to the formation of the holon composition, which can operate autonomously from this point forward. The established composition can now dynamically include or exclude holons through voting to effectively manage holon collaborations. For example, if k out of n holons agree on a specific decision, it is implemented in the entire composition. The developer can customize the voting procedure to suit specific requirements.

The holon composition layer ensures that the holons can merge and form holon compositions where they can trust each other. The HCFW ensures the holon composition's reliability across various network conditions, including full connectivity, unreliable connectivity, and complete isolation [13].

3.3 Collaboration Layer

The holon collaborations are an organized subset of holons from the composition. In a collaboration, one holon assumes the role of *mediator* to facilitate interaction. In contrast, the others serve as *collaborators*, engaging in information exchange. A holon collaboration has its *state* that maintains both the individual states of participating holons and the collective state of the collaboration. This shared state is synchronized across all the holons in the collaboration.

The *mediator holon* captures sensations and gathers holons that can fulfil the *roles* of the *behaviours* linked to these sensations. Following this, it executes the code for scheduling the behaviours and manages the collaboration state. The mediator holon is selected based on criteria that can be superior connectivity, computational power, battery life, diverse capabilities, or other criteria determined by the developer. The collaborator holons cooperate with the mediator to implement behaviours.

When two compositions are merged, their collaboration layers start sharing the state information. Both compositions dispatch sensations to the selected mediator holon, forwarding them to all collaborators.

The holon collaborations are the extension of *holarchies* [7] as they coordinate closely with their environment with a software layer for behaviour management and scheduling, blending structural integrity with functional agility.

3.4 Behavioural Layer

A *behaviour* is a collective response of the holons to a sensation triggered by their collaboration. It is a modular unit that determines how holons of collective intelligence interact over a certain period. It consists of three parts: 1) The *triggering condition* must be met for the behaviour to initiate. This condition also verifies the availability of the necessary holon resources. 2) The *responsibility* specifies which holons are eligible to partake in executing the behaviour and maps the required responsibilities to the capabilities of the collaborating holons. This ensures that only holons with the requisite and available capabilities are selected for the task. 3) The *body* contains the programming logic specified by the developer to define the cooperative interactions among holons, leveraging their capabilities.

Behaviours are generally transient, with holon capabilities utilized exclusively during the execution phase. If a holon's capability is unavailable, the behaviour execution is halted and rescheduled once it is available again.

4 Case Study: Disaster Management Ecosystem

To demonstrate the HPM, we apply it to a disaster management ecosystem. This shows how various CSs react to environmental stimuli and achieve goals by collaborating strategically and executing behaviours.

Disaster management involves a coordinated response to crises, incorporating several entities with the necessary expertise and resources. These entities possess heterogeneous cyber-physical systems vital for maintaining situational awareness and enabling coordination among organizations and residents in disaster scenarios [14]. However, this process faces various challenges, including effective communication, coordination, resource allocation, and timely information management in disasters [15].

Part I of Figure 2 presents a stripped-down disaster management scenario that involves various entities and their assets, including Command and Control (C2), government departments, and procedures. In this scenario, an SOS call is initiated from the disaster site. This call usually includes the nature of the disaster (i.e., earthquake, wildfire, stranded or missing person) and its location (last known position, exact or approximate). C2 receives the SOS call and has assets, including a search plane, a rescue helicopter, and a Micro Aerial Vehicle (MAV). C2 communicates with other departments, including fire, police, and health services. Responding to the call, C2 assesses the disaster's nature and initiates appropriate operations. The figure illustrates two operations: Public Awareness and Search and Rescue (SAR).

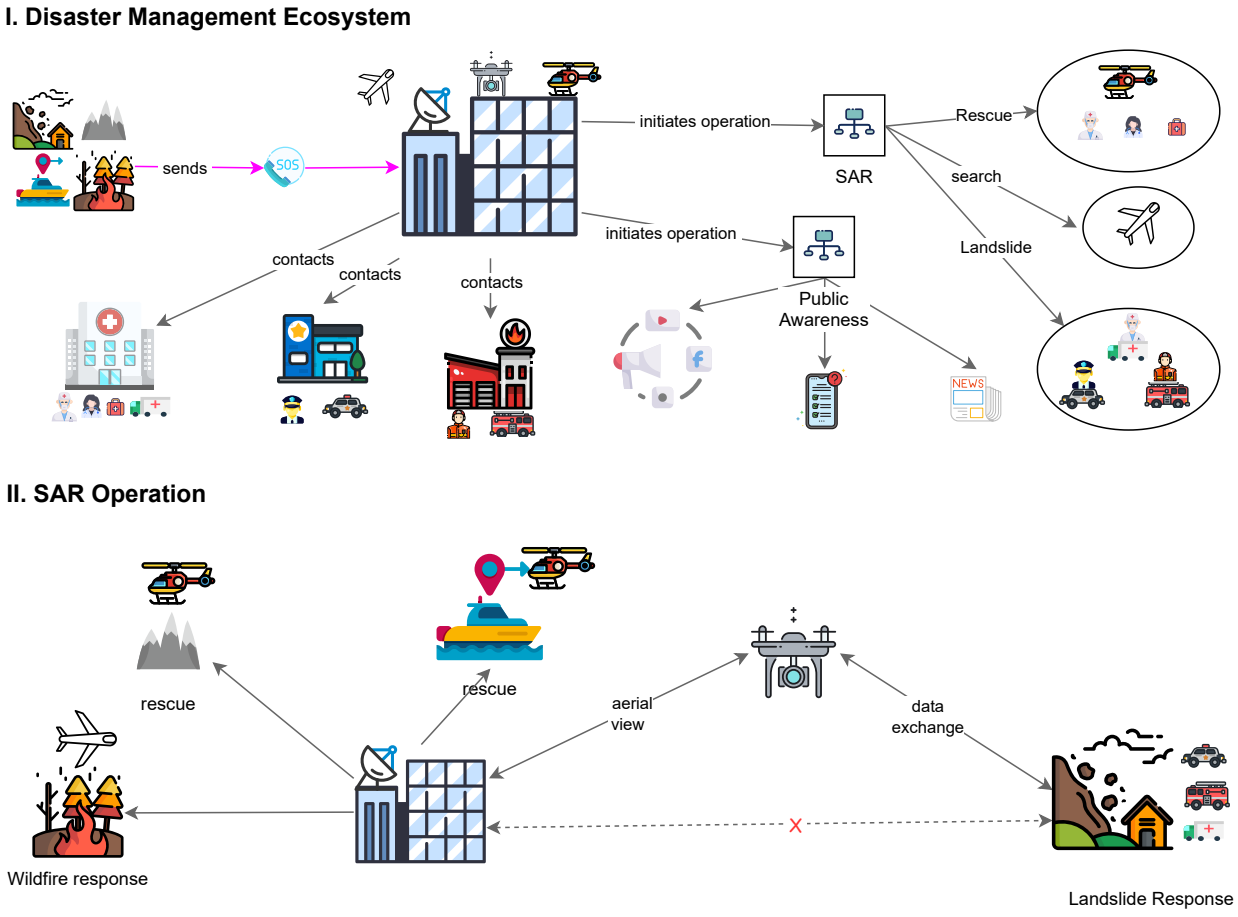

Figure 2: **I)** The Disaster Management Ecosystem. **II)** The details of Search-and-Rescue Operation. **III)** The ecosystem is programmed using HPM.

Part II of Figure 2 details the SAR operation, illustrating how C2 assembles and deploys response teams composed of the necessary personnel and assets. If the call concerns a missing person, C2 deploys a search plane to pinpoint the person's location. Following a successful search, C2 evaluates whether the person is stranded (e.g., in the ocean or mountains) and dispatches a rescue helicopter equipped with a medical team. The C2 mobilizes a search plane outfitted with water-spraying gear for wildfire incidents.

For a landslide, the C2 must coordinate with other departments due to the complex nature of the response, requiring a broader range of capabilities. The landslide site requires paramedic staff and ambulances from the Health Department to provide first aid to those injured and transport critically injured individuals to the hospital. A fire engine is also needed due to the fire risk from short circuits. Furthermore, police presence is essential to safeguard the property of the affected individuals. C2 coordinates with the relevant departments to mobilize these resources.

Upon arrival at the site, the assets begin their activities by forming a team. Throughout the operation, the team communicates constantly with C2 to provide updates about casualties, injuries, and the operation status and receive instructions from C2. However, due to the remote location, the communication link between SAR and C2 is prone to disruptions. To mitigate this, C2 deploys an MAV to act as a communication relay, ensuring a stable connection and seamless coordination for the operation.

For the public awareness operation, C2 disseminates information about the disaster through official communication channels, including social media or local news outlets.

4.1 SoS Perspective and Challenges

In this ecosystem, each entity, such as C2 and various departments, functions as a CS in its own right. These heterogeneous CSs must dynamically compose into SoS at runtime to execute specific operations, including SAR and Public Awareness. This runtime-composed SoS needs to adaptively respond to the environmental inputs by triggering appropriate behaviours essential for efficient operation within the ecosystem. Similar to IoT programming, software developers must be able to program these behaviours, ensuring the SoS can act cohesively and effectively within the ecosystem.

Programming such an SoS presents numerous challenges that traditional programming paradigms often need to be revised to address. These challenges include ensuring cooperation among the C2, rescue teams, and MAVs under unreliable network conditions and executing various SAR operation tasks such as team deployment, search and rescue, and ad-hoc bridging of MAVs. This requires software developers to craft behaviours that allow the dynamic composition of teams, harnessing the diverse capabilities of CSs to respond adaptively to environmental stimuli and operational demands in the disaster management system.

4.2 Realization of the scenario in HPM

Part III of Figure 2 demonstrates the scenario modelling using the HPM. In this model, the entities *C2*, *HealthDpt*, *PoliceDpt*, and *FireDpt* are depicted as holons, each possessing specific resources. C2's resources include a search plane, a rescue helicopter, and an MAV. The *FireDpt* holon has firefighters and fire engines at its disposal. The *PoliceDpt* holon is outfitted with police officers and police cars. Finally, the *HealthDpt* holon has doctors, nurses, and ambulances as its resources.

The C2 holon interacts with the external world from where it receives an SOS sensation. The sensation includes the disaster's nature and location. Upon receiving this sensation, the C2 evaluates its content. It initiates the formation of a composition that includes *HealthDpt*, *PoliceDpt*, and *FireDpt* holons. This composition then collaborates to execute specific behaviours to respond to the disaster. In the figure, the composition illustrates search and rescue (SAR) collaboration. In the SAR collaboration, the C2 is the mediator, while the other departments are collaborators.

The pseudocode for SAR collaboration is provided in Listing 1. The collaboration maintains a state object `sosCall`, which is initially set to `null`. This object is populated once C2 detects an SOS sensation, with the sensation's location being assigned to `sosCall.loc` and the disaster nature to `sosCall.type`, which could be a landslide, wildfire, stranded, or missing person. The collaboration has several behaviours triggered based on `sosCall.type`.

The `wildfireResp` behaviour has a *condition* that the `sosCall.type` must be `"landslide"` and the `waterCarrier` *role* must be fulfilled by a resource from a holon participating in the collaboration. Furthermore, the resource must possess a full water tank and sprayer. C2 Holon's search planes are preferred for this role over helicopters due to their lower fuel consumption, long flight times, and no need for landing at the site. The `waterCarrier` proceeds to `sosCall.loc`, sprays water and returns to base either upon depleting its tank or upon receiving a command from C2.

```
Holon Collaboration SAR {
sos: sosCall := null

behaviour wildfireResp(waterCarrier):
  if sosCall.type is "wildfire" and
     waterCarrier has waterTank and
     waterCarrier has waterSprinkler and
     waterCarrier.waterLevel is full:
  do
    move(waterCarrier, sosCall.loc)
    sprinkleWater(waterCarrier)
    if waterCarrier.tank is empty or
       C2.sensation is "comeback":
      returnToBase(waterCarrier)

behaviour search(searcher):
  if sosCall.type is "missing person":
  do
    move(searcher, sosCall.loc)
    search()
    if search.successful:
      returnToBase(searcher)
      set sosCall.type == "rescue"

behaviour rescue(helicopter, medic):
  if sosCall.type in ["rescue", "stranded"]:
    move(helicopter, sosCall.loc)
    # provide first aid if needed
    returnToBase(helicopter)

behaviour landslideResp(entities[]):
  if sosCall is "landslide":
    async for e in entities:
      alert(e)
      await e.status == "on_site"
      coordinate(C2, e)
    if commLink is "weak":
      deployMAV(sosCall.loc)

}
```

Listing 1: Pseudocode for SAR collaboration

The `search` behaviour is triggered if the *condition* `sosCall.type == "missing person"` is met. This behaviour needs a `searcher`, which C2 holon's search plane can fulfil. Operationally, the `searcher` moves to the `sosCall.loc` and performs the search. Upon completing the search, it returns to the base *and* sets the `sosCall.type` to `"rescue"`, thereby triggering the rescue behaviour.

The `rescue` behaviour can also be triggered if `sosCall.type` is `"stranded"`. In addition to the rescue helicopter, the *role* of paramedic staff is required, which can be fulfilled by the resources of the healthDpt holon. The helicopter transports the paramedic staff to the site, where they provide first aid if needed, retrieve the victim, and then return to base.

The `landslideResp` behaviour is triggered when `sosCall.type` is `"landslide"`. All the holons in the composition (e.g., C2, HealthDpt, RescueDpt, FireDpt, PoliceDpt) are needed to fulfil the entity roles. C2 alerts all entities and ensures their teams are dispatched to the site to carry out their specific duties. The resources from FireDpt, HealthDpt, and PoliceDpt are all crucial for mitigating fire hazards, providing medical attention, and securing property during emergencies. C2 communicates with these teams and exchanges data with them. Should the communication link between C2 and the on-site teams weaken, C2 deploys the MAV to establish a communication bridge. This MAV enhances C2's communication capabilities and situational awareness by live-streaming video from the disaster site.

5 Discussion and Conclusions

In this article, we proposed the HPM and demonstrated that it aligns closely with SoS properties and characteristics [3]. Notably, the HPM facilitates the emergence of novel situations and the evolutionary development of collaborative behaviours. Moreover, new holons might join or leave the holon composition based on their availability and need in the situational context. Accomplishing this with programming-level abstractions inside different devices and sub-systems would be complicated, as these typically focus on devices' abilities rather than on the possible roles the device might have in a bigger whole.

In addition to composing holon-based systems, the HPM also introduces additional benefits, which still need to be investigated. In particular, focusing on holon composition will introduce new ways to consider security-related aspects when designing the system, as these are natural candidates to refer to when engineering security and safety-related features for SoS in practice. Focusing on the collaboration and the different roles of the participating sub-systems then allows the developers to consider the big picture instead of examining the capabilities of individual devices in isolation.

References

[1] Gerard Briscoe, Suzanne Sadedin, and Philippe De Wilde. Digital ecosystems: Ecosystem-oriented architectures. *Natural Computing*, 10:1143–1194, 2011.

[2] Jakob Jul Jensen. Applying a "smart ecosystem" mindset to rethink your products. *Computer*, 53(12):98–101, 2020.

[3] Mark W. Maier. Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284, 1998.

[4] Rasmus Adler, Frank Elberzhager, Rodrigo Falcao, Julien Siebert, Eduard Christiaan Groen, Jana Heinrich, Florian Balduf, and Peter Liggesmeyer. A research roadmap for trustworthy dynamic systems of systems – motivation challenges and research directions. *Fraunhofer IESE, Germany, Tech. Rep*, 1, 2023.

[5] Antero Taivalsaari and Tommi Mikkonen. A roadmap to the programmable world: Software challenges in the IoT era. *IEEE software*, 34(1):72–80, 2017.

[6] Gordon Blair, Yérom-David Bromberg, Geoff Coulson, Yehia Elkhatib, Laurent Réveillère, Heverson B. Ribeiro, Etienne Rivière, and François Taïani. Holons: Towards a systematic approach to composing systems of systems. In *Proceedings of the 14th International Workshop on Adaptive and Reflective Middleware*, pages 1–6, December 2015.

[7] Arthur Koestler. *The ghost in the machine*. The Ghost in the Machine. Macmillan, 1968.

[8] Vatsala Nundloll, Yehia Elkhatib, Abdessalam Elhabbash, and Gordon S Blair. An ontological framework for opportunistic composition of IoT systems. In *2020 IEEE international conference on informatics, IoT, and enabling technologies (ICIoT)*, pages 614–621, 2020.

[9] Abdessalam Elhabbash, Vatsala Nundloll, Yehia Elkhatib, Gordon S. Blair, and Vicent Sanz Marco. An ontological architecture for principled and automated system of systems composition. In *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 85–95, June 2020.

[10] Ahmed R Sadik, Bram Bolder, and Pero Subasic. A self-adaptive system of systems architecture to enable its ad-hoc scalability: Unmanned vehicle fleet-mission control center case study. In *Proceedings of the 2023 7th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, pages 111–118, 2023.

[11] Khalid Halba, Edward Griffor, Ahmed Lbath, and Anton Dahbura. A framework for the composition of IoT and CPS capabilities. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1265–1272, 2021.

[12] Bill Wasik. In the Programmable World, All Our Objects Will Act as One | WIRED. `https://www.wired.com/2013/05/internet-of-things-2/`, 2013.

[13] Niko Mäkitalo, Timo Aaltonen, Mikko Raatikainen, Aleksandr Ometov, Sergey Andreev, Yevgeni Koucheryavy, and Tommi Mikkonen. Action-oriented programming model: Collective executions and interactions in the fog. *Journal of Systems and Software*, 157:110391, 2019.

[14] Chao Fan and Ali Mostafavi. Establishing a framework for disaster management system-of-systems. In *2018 Annual IEEE International Systems Conference (SysCon)*, pages 1–7. IEEE, 2018.

[15] Saad Mazhar Khan, Imran Shafi, Wasi Haider Butt, Isabel de la Torre Diez, Miguel Angel López Flores, Juan Castanedo Galán, and Imran Ashraf. A systematic review of disaster management systems: approaches, challenges, and future directions. *Land*, 12(8):1514, 2023.