A self-adaptive system of systems architecture to enable its ad-hoc scalability - (Unmanned Vehicle Fleet - Mission Control Center Case study)

Ahmed Sadik, Bram Bolder

2023

Preprint:

A self-adaptive system of systems architecture to enable its ad-hoc scalability

Unmanned Vehicle Fleet - Mission Control Center Case study

AHMED R. SADIK, Honda Research Institute Europe, 63073, Germany

BRAM BOLDER, Honda Research Institute Europe, 63073, Germany

PERO SUBASIC, Honda Research Institute USA, CA 95134, United States

The concept of System of Systems (SoS) refers to a collection of Constituent Systems (CSs) that interact to deliver an emergent behavior that cannot be achieved by any individual CS on its own. The focus of this research is on the ad-hoc scalability of SoS, meaning that the size of the system can change during operation by adding or removing a CS or changing the size of existing CSs. The Unmanned Vehicle Fleet (UVF) is selected as a practical example to showcase the challenge and solution of ad-hoc scalability. UVF has various applications in fields such as search and rescue, intelligent transportation and mobility, but it operates in a dynamic environment that is prone to uncertainties like changing missions, increasing range and capacity, UV failures, and battery requirements. The proposed solution to overcome these uncertainties is a self-adaptive system that can dynamically change the UVF architecture in real-time, allowing for upscaling or downscaling the size of the UVF. The Mission Control Center (MCC) can control this change either through fully-automatic mode based on performance criteria like battery utilization and communication traffic, or through manual mode where the operator makes the decision. A multi-agent environment and rule management engine are implemented to simulate the UVF behavior and validate the proposed solution in both automatic and manual modes.

CCS CONCEPTS • System of Systems • Unmanned Vehicle Fleet • Self-adaptive Architecture • Ad-Hoc Scalability

Additional Keywords and Phrases: Holonic Architecture, Multi-agent Simulation, Swarm Robotics

ACM Reference Format:

1 INTRODUCTION

The definition of System of Systems (SoS) evolved over time, rather than being established immediately [16]. One early definition of SoS was presented in [3] as "an array of Constituent Systems (CSs) functioning together to achieve a common goal". SoS is a result of the integration of multiple CSs to serve a higher purpose, and evolves over time as tasks are added, removed, or modified during operation [15]. It is also emergent [6], meaning its behaviors are not fully understood until it is integrated, and the overall SoS value cannot be simply summed up from the values of its constituent systems. As a result, some behaviors may be desirable, while others may not. Swarm robotics is an ideal example of an SoS where multiple robots work together to achieve a common goal. Unmanned Vehicles (UV), such as Unmanned Aerial Vehicles (UAV), Unmanned Ground Vehicles (UGV), Unmanned Surface Vehicles (USV), or Unmanned Underwater Vehicles (UUA), offer different capabilities and can be controlled by a remote operator or a pre-programmed system [7]. The use of UVs offers many benefits, including cost efficiency, versatility, and ease of deployment [9]. However, they also come with limitations such as limited mission capacity, complexity, range, battery life, cybersecurity risks, and the potential for malfunctions [2]. Using a fleet of UVs can overcome these limitations and make up for the individual limitations of each unit [4].

This research defines the Unmanned Vehicle Fleet (UVF) as an SoS that integrates multiple UV swarms to achieve complex missions by forming collective structures and emerging behaviors that enhance their overall capabilities [22]. The ad-hoc scalability of the UVF is crucial for its evolution and emergence [13]. Ad-hoc scalability refers to the system's ability to adapt to changes that occur during operation by adding or removing resources to meet new demands [10]. The research presents an approach to design and simulate a Mission Control Center (MCC) that can control the UVF's ad-hoc scalability by dynamically adapting the SoS architecture. The next section outlines the research problem. Section 3 highlights the central, hierarchical, and holonic system architectures as the foundation of the proposed solution. The implementation of the proposed solution is described in section 4, using a multi-agent environment that models the MCC, the operator, and the UVs as software agents. Section 5 discusses the simulation case study and its results. Finally, section 6 summarizes and discusses the research and highlights future work.

2 PROBLEM

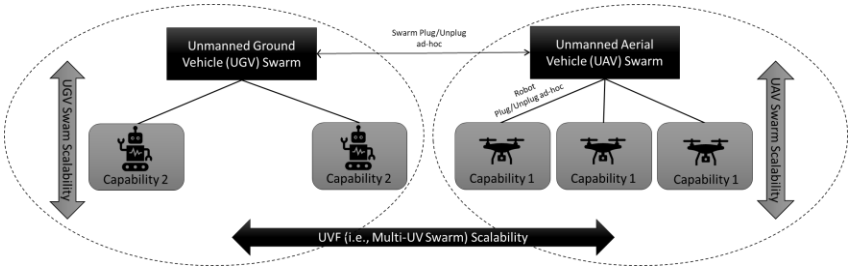

Figure 1: Ad-hoc Scalability in the UVF

The UVF can be seen as a realization of the SoS concept with multiple UV swarms. Swarm robotics focuses on designing simple physical agents to achieve desired collective behavior through local interaction among the agents and with the environment [8]. This definition aligns with the SoS definition, highlighting scalability as the key to its emergence and evolution [1]. As shown in Figure 1, the UVF involves two types of ad-hoc

scalability. The first type occurs within the UV swarm, where adding or removing UV entities impacts the swarm's capacity to execute tasks that match the UV capabilities. The second type occurs within the UVF, where including or excluding a UV swarm affects the SoS existing capabilities and overall plan. UVF scalability is limited by the static design of its system architecture pattern, that is often implemented by the MCC is the fundamental challenge that has been highlighted. The main research question that has been addressed by this research is "*How to achieve the ad-hoc scalability concept in a UVF by adapting its architecture based on its performance criteria such as battery utilization and communication traffic*"

3 SOLUTION

The system architecture pattern is the foundation for implementing an application design at the highest level of abstraction. It defines the overall system structure and behaviour, providing a blueprint for its design. The proposed research solution incorporates a dynamic MCC design that enables scalable UVF architecture patterns to meet changing mission requirements. With this design, the UVF can be scaled in real-time without disruption, allowing it to easily adapt to new requirements. The MCC includes a decision-making mechanism that operates in either a fully automatic mode or a supported-manual mode. In fully automatic mode, the MCC automatically selects the best UVF architecture pattern to meet current mission needs [14]. In supported manual mode, the operator chooses the architecture pattern, and the MCC validates the choice and provides a viable solution. Figure 2 illustrates the solution concept used in designing the MCC, showcasing three architecture patterns (Central, Hierarchal, Holonic) as examples. However, the concept can be applied to other system patterns as well.

The proposed central architecture pattern can be seen on the left side of Figure 2. Two structure layers are proposed in this pattern. The first layer is the supervision layer, where the MCC locates. The second layer is the operational layer, where the UVs exist. All the communication links between the MCC and the UVs are based on master-slave protocol, where in this case the MCC takes the role of the master node while the UVs take the role of slave nodes. In master-slave protocol, one device acts as a master node that manages the timing and the data flow. The slave nodes cannot initiate the communication or communicate with one another Therefore, all the UVs waits the MCC command and report back their feedback. Would the UVs like to cooperate, they must communicate through the MCC as a hub. The central pattern has simple structure and behavior, however it suffers from the Single Point of Failure (SPoF) which reduce its reliability. Furthermore, the scalability and flexibility of the central architecture is very limited by the number of UVs that can be connected and controlled by one MCC simultaneously [23].

The proposed hierarchical architecture is shown in the middle of Figure 2. This architecture improves upon the central pattern by increasing scalability and flexibility [24]. It has three structure layers: the execution layer, where UVs receive direct commands from the MCC in the supervision layer; the operational layer, where the leader UVs, in the execution layer, lead a group of follower UVs via master-slave communication links. Global cooperation occurs through the MCC and leader UVs, resulting in higher delay but local cooperation occurs through the leader UVs. This structure provides a more responsive and reliable performance compared to the central architecture, as the SPoF is distributed across multiple locations [5]. However, the system is not fault-tolerant as the follower UVs are dependent on their leaders, which negatively affects its robustness. Additionally, scalability can only be achieved at the bottom of the hierarchy.

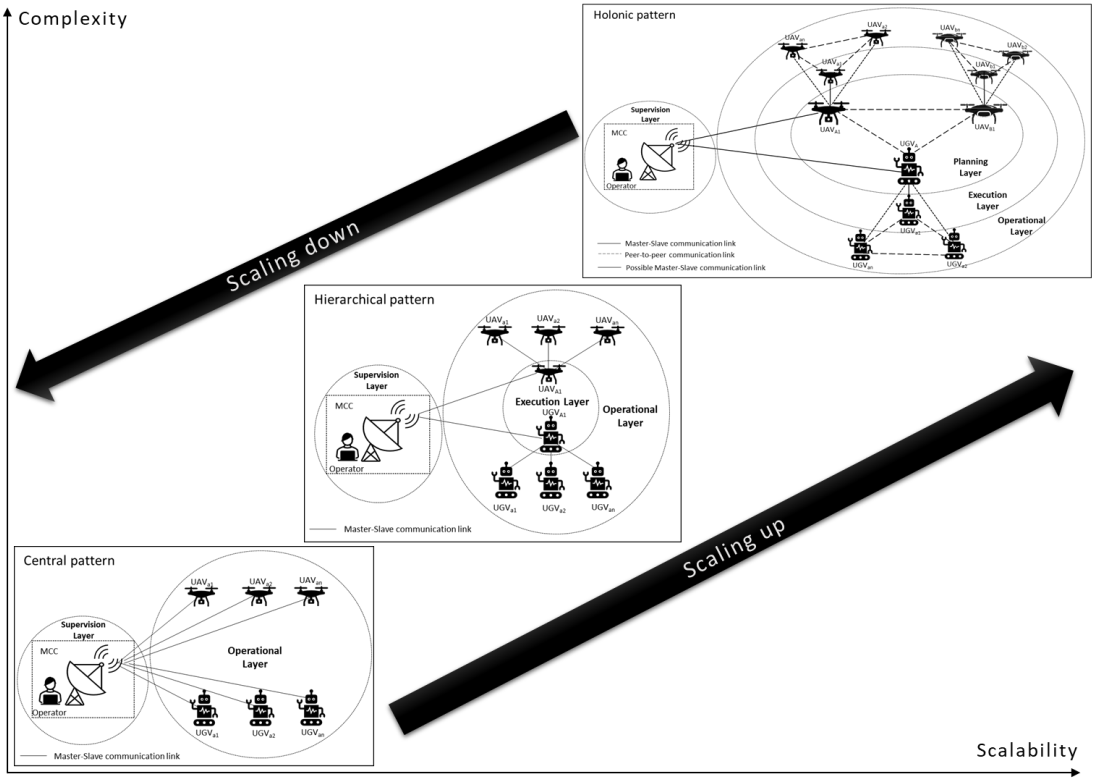

Figure 2: solution concept – scaling the UVF via adapting its pattern architecture by the MCC

The holonic architecture pattern is depicted on the right side of Figure 2. This pattern features four structural layers, with UVs in the planning layer receiving commands directly from the MCC in the supervision layer via master-slave links [21]. The UVs in the planning layer communicate with each other using peer-to-peer links, allowing for any node to initiate communication when an event occurs. One UV in the planning layer has a permanent link with a UV in the execution layer, which carries out the plan with the help of other UVs in the operational layer. In the event of failure, the UV in the planning layer can promote another UV in the operational layer to the execution layer to establish a new master-slave link. The holonic pattern offers high scalability and flexibility, capable of integrating a large number of UVs, executing complex plans and having a robust architecture with no single point of failure [20].

4 IMPLEMENTATION

The similarities between Multi-Agent Systems (MAS) and UVF are clear. An MAS is a group of artificially intelligent agents that work together in a flexible network to solve problems beyond the capabilities of a single agent [14]. Meanwhile, a UVF is a group of UV swarms, or control systems, that interact to produce a collective behavior that can't be achieved by a single swarm [12]. This comparison has led to the implementation of the proposed solution using a MAS environment called Java Agent DEvelopment (JADE), as shown in Figure 3-a.

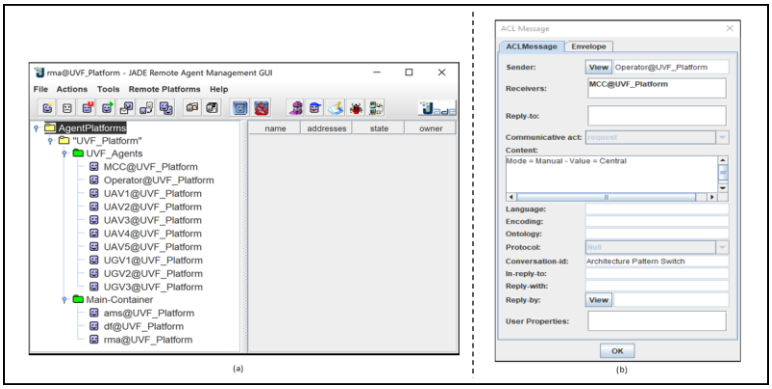

Figure 3: (a) UVF implement as a MAS in JADE (b) ACL message communication

Figure 3-a shows JADE implementation of UVF case study under UVF platform. The UVF platform is composed of two containers, where three software agents are living. The main JADE container contains two essential agents, which are the Agent Management System (AMS) and a Directory Facilitator (DF) [19]. AMS provides a unique ID for every agent to be used as an agent communication address. While the DF announces the services which each agent can afford. The UVF-agents container is bundling three different categories of agents. The first category is the humans which is represent as the operator agent, the second category is controller which is represented as the MCC agent, the third category is the autonomous machines which are represented as the UVs.

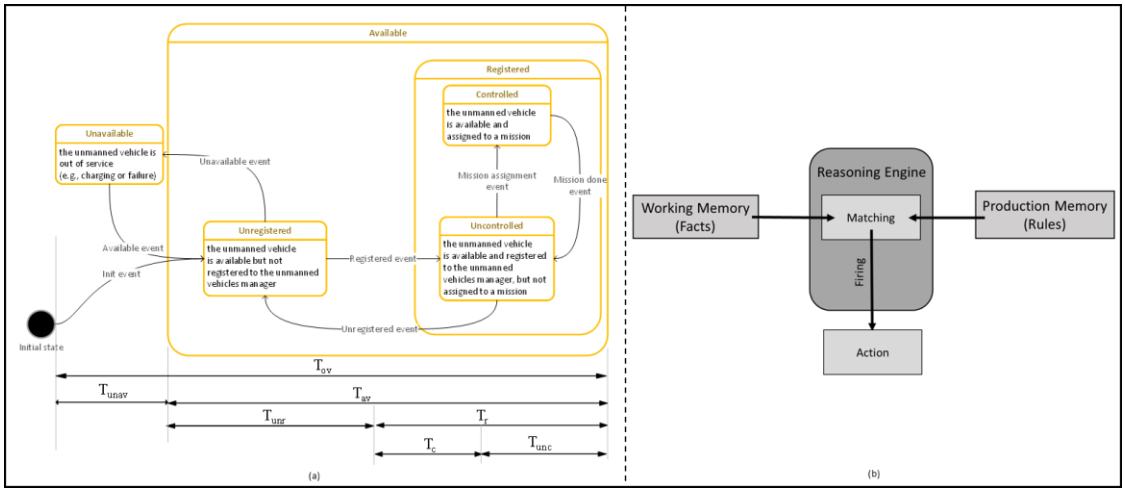

Figure 4: (a) UV behavioral state machine (b) Drools rule engine

JADE agents use the Foundation for Intelligent Physical Agent - Agent Communication Language (FIPA-ACL) to exchange messages [11]. One example of FIPA-ACL message exchange can be seen in Figure 3-b. The ACL message that is shown in Figure 3-b explains how the operator agent can override the control mode from automatic to manual, as the message is sent from the operator agent to the MCC agent. The message

content field contains the assigned new value of the required architecture pattern by the operator, which in this case is central. The same message exchange mechanism is followed by the other agents within the UVF platform. The only difference that the operator agent decision making, and behavior is based on direct interaction with a human operator. However, the decision making in case of the MCC, and the UVs agents is done autonomous, based on a set of rules that are applied by the MCC and the UV behavioral state machine.

Figure 4-a shows the state machine that is used by a UV agent to model a UV behavior. The state machine diagram is a dynamic behavioral diagram from SysML standard language, which shows the sequences of states that a UV go through during its operation in response to events that may trigger an action [18]. The following UV states are defined:

- Initial: a simple state, where UV initially becomes ready to operate
- Available: a composite state, where the UV is either registered or unregistered. T_{av} is the time which the UV is available, and either registered or unregistered.
- Unavailable: a simple state, where the UV cannot be registered, as it is out service due to a failure or battery charging. T_{unav} is the time which the UV is unavailable.
- Unregistered: a simple state, where the UV is available but not registered yet, as it is configuring its parameters. T_{unr} is the time which the UV is available but unregistered.
- Registered: a composite state, where the UV is either controlled or uncontrolled. T_r is the time which the UV is registered either controlled or uncontrolled
- Uncontrolled: a simple state, where the UV is registered but not assigned to any mission yet. T_{unc} is the time which the UV is registered and uncontrolled
- Controlled: a simple state, where the UV is registered and assigned to a mission. T_c is the time which the UV is registered and controlled

Initially, the UV becomes available and unregistered after it receives an init event. If the UV succussed to configure itself, it will send a registered event to UVs manager. When the MCC allows its registration, it will be registered but not controlled, till the MCC assigns a mission to it, then the UV in controlled state. When the UV finishes its mission, it goes back to uncontrolled state. If the UV needs to be reconfigured, it must return to the unregistered state. If the UV has a failure or requires a battery charge, it must return to the unavailable state. Based on this state machine diagram, the MCC can monitor the UVs status and hence it calculates their Utilization. The UV utilization (U_{uv}) is the ratio between T_c to T_{unc}. Furthermore, the MCC calculates the UV communication traffic (Tr_{uv}), which is the data rate communicated through the UV. U_{uv} and Tr_{uv} are two important performance criteria that are used by the MCC to balance the UVF usage while adapting it is architecture pattern, as it will be illustrated in detail in the next section.

Figure 4-b shows Drools rule engine that has been impeded into the MCC agent to contain the operation rule. Drools is a deliberative software agent that codify the knowledge base and the reasoning into facts, rules, and actions [17]. The working memory holds the facts which present the domain knowledge, while the production memory holds the rules which are presented in form of conditional statements. The reasoning engine is a problem solver which solves a given problem by matching the present facts with the existing rules. Drools reasoning engine can apply a hybrid chaining reasoning. A hybrid chaining reasoning is a mix between the forward and the backward chaining which can be more efficient in some cases than both.

5 CASE STUDY

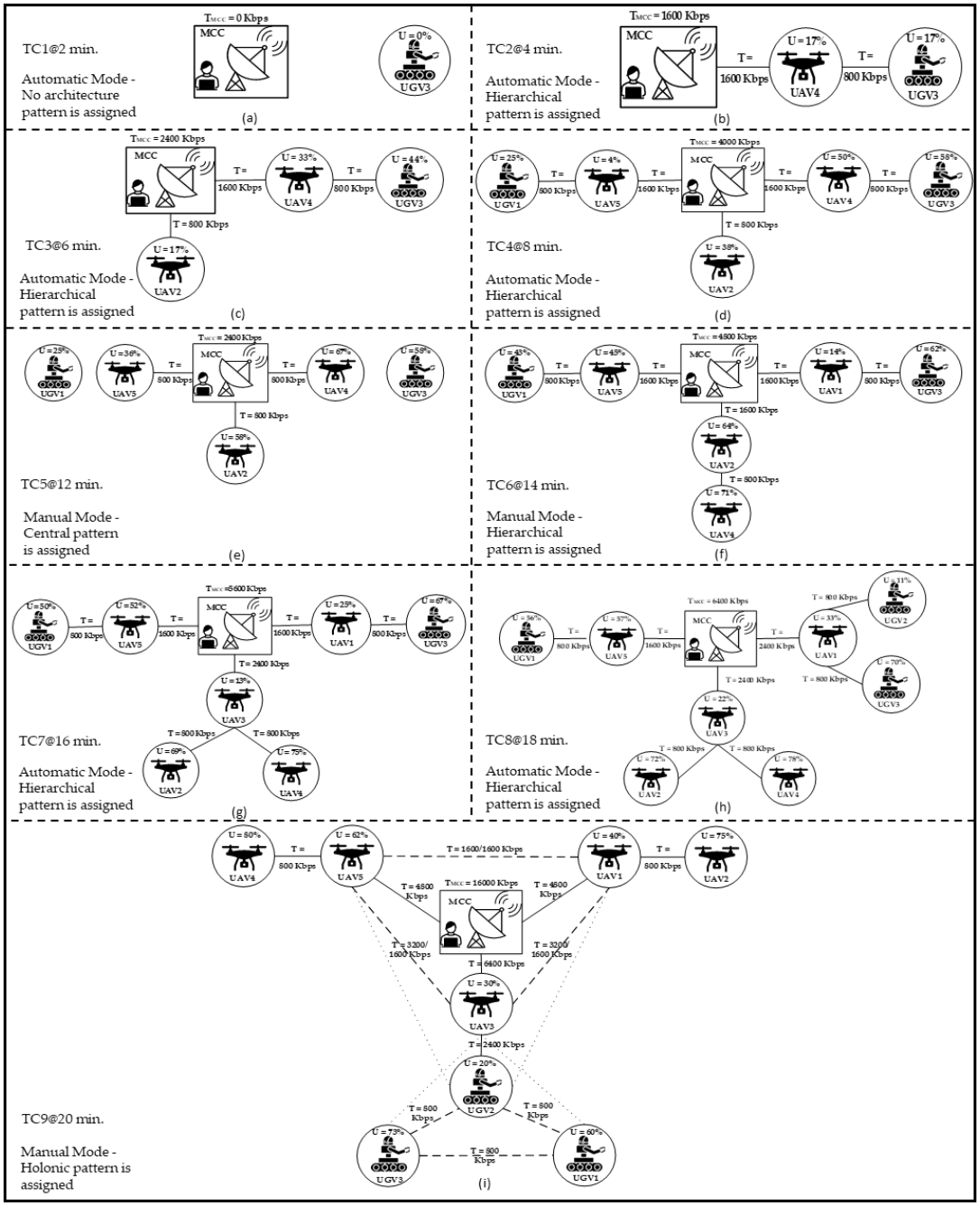

Figure 5: different architectural structure based on the test cases

Figure 5 shows the 9 test cases used in our approach and deployed in JADE for a 20-minute simulation scenario. The simulation involves a group of UAVs and UGVs randomly changing their state according to set constraints. The MCC provides either automatic or manual control based on the operator's choice and adapts the UVs to one of the architecture patterns based on utilization and traffic monitoring.

5.1 Constraints

- **C1**: the maximum number of available UAVs is five.
- **C2**: the maximum number of available UGVs is three.
- **C3**: all the available UAVs are within the MCC control range.
- **C4**: all the available UGVs are out of the MCC control range.
- **C5**: all the UAVs have similar capabilities, thus any of them can operate as a leader UAV in a cluster.
- **C6**: all the UGVs have similar capabilities, thus any of them can operate as a leader UGV in a cluster.
- **C7**: each of the connected UVs transmits fixed data rate of 800 Kbps.
- **C8**: maximum number of communication links between the MCC and the UVs is three.
- **C9**: maximum number of communication links between a leader UV and the follower UVs is two.

5.2 Rules

- **R1**: in central pattern, one layer (i.e., operational) of UVs that all in the MCC control range can exist.
- **R2**: in hierarchical pattern, two layers (i.e., operational, and execution) of UVs can exist.
- **R3**: in holonic pattern, three layers (i.e., operational, execution, and planning) of UVs can exist.
- **R4**: a UV out of the MCC range is connected either in the execution layer or in the planning layer, via a UV in the operational layer.
- **R5**: in automatic operational mode, the UVs are filling the operational layer at first, then the execution layer, then the planning layer.
- **R6**: the UVs with minimum utilization have higher priority to be directly connected to the MCC (i.e., operational layer must be formed from the UVs with minimum utilization).
- **R7**: connecting a follower UV to a leader UV is based on balancing the traffic on the current leader UVs.
- **R8**: connecting a follower UV to a leader UV is based on balancing the utilization of the current leader UVs, if all the leader UVs have the same traffic.
- **R9**: the third layer of the holonic pattern is composed of clusters from the same UV types.

5.3 Test cases

- **TC1**: at 2 min simulation time, the operation mode is set to automatic. As seen in Figure 5-a, UGV_3 has registered but is outside the MCC's control range and cannot be controlled. Since no UVs are under control, no architecture pattern has been assigned.
- **TC2:** at 4 min simulation time, the operation mode is set to automatic. In Figure 5-b, UAV_4 has registered and is now under MCC control. As a result, UAV_4 establishes a communication link with the MCC. R4 is applied, allowing UGV_3 to also connect to the MCC through UAV_4. With two communication layers present, the MCC structures the UVs into a hierarchical pattern as specified by R2.
- **TC3**: at 6 min simulation time, the operation mode is set to automatic. In Figure 5-c, UAV_2 is registered and in communication with the MCC, as shown by R5. The hierarchal pattern is maintained through R2.

- **TC4**: at 8 min simulation time, the operation mode is automatic. In Figure 5-d, UAV_5 and UGV_1 are registered, so the MCC establishes communication links with them via R5 and R4, respectively. The existing UVs are structured into a hierarchal pattern, as per R2, with two layers of communication.
- **TC5**: at 12 min simulation time, the operation mode is changed to manual and the assigned pattern is central. The MCC implements R1 to structure the UVs in a centralized pattern as seen in Figure 5-e.
- **TC6**: at 14 min simulation time, the operator switches operation mode to manual and sets the pattern to hierarchical. In Figure 5-f UAV_1 is newly registered with the MCC, while UGV_1 and UGV_3 are already registered but not controlled. The MCC uses R2, R4, R5, R6, and R7 to form a hierarchical pattern structure. R5 and R6 dictate that operational layer connections to the MCC must be established using UVs with lowest utilization. UAV_1, UAV_2, UAV_4, and UAV_5 can directly connect to the MCC. According to Figure 6, UAV_1, UAV_2, UAV_4, and UAV_5 utilization rates are 0%, 58%, 67%, and 36% respectively. C8 states that UAV_1, UAV_2, and UAV_5 form the operational layer, while UAV_4, UGV_1, and UGV_4 form the execution layer. The leader-follower connections are balanced with three leader UVs (UAV_1, UAV_2, UAV_5) connected to three follower UVs (UAV_4, UGV_1, UGV_4). The follower UV selection is random, as all options meet MCC rules.
- **TC7**: at 16 min simulation time, the operation mode is set to automatic by the operator. UAV_3 registers with the MCC in Figure 5-g. To optimize utilization of the operational layer, the MCC applies R6. According to Figure 6, the utilization rates for UAV_1, UAV_2, UAV_3, UAV_4, and UAV_5 are 14%, 64%, 0%, 71%, and 45% respectively. Thus, the first UV layer is formed by UAV_1, UAV_3, and UAV_5. The MCC then applies R2, R4, R7, and R8 to structure the execution layer. R7 reveals an imbalance in the traffic on the leader UVs (UAV_1, UAV_3, UAV_5) that can be seen in Table 1, with one UV required to handle 1600 Kbit. To resolve this, R8 is applied to identify the leader UV with minimum utilization (UAV_3 = 0%). UAV_3 is selected to handle the highest traffic in the execution layer (1600 Kbit). The MCC forms a two-layer structure, resulting in a hierarchical pattern, according to R2.
- **TC8**: at 18 min simulation time, the operation mode remains automatic. A new registration of UGV_2 is shown in Figure 5-h. Due to the high traffic of UAV_3 (1600 Kbit) as indicated in Table 1, UGV_2 cannot be led by UAV_3 as per R7. The MCC uses R8 to choose between UAV_1 or UAV_5 to lead UGV_2. Based on Figure 6, utilization of UAV_1 and UAV_5 are 25% and 52% respectively. Therefore, UAV_1 is selected to lead UGV_2. The MCC establishes the connection between all registered UVs in a hierarchical pattern, as indicated by R2.
- **TC9**: at 20 min simulation time, the operator switches the operation mode to manual and the assigned pattern is holonic. In Figure 5-i, the MCC uses R5 and R6 to form the operational layer using UVs with minimum utilization. Based on Figure 6, the utilization levels of UA_1, UA_2, UA_3, UA_4, and UA_5 are 33%, 72%, 22%, 78%, and 57% respectively. Hence, based on C8, UAV_1, UAV_3, and UAV_5 form the operational layer. Using R9, UGV_1, UGV_2, and UGV_3 form a cluster with higher traffic compared to connecting UAV_4 and UAV_2. The cluster is connected to the UAV with the lowest utilization, UAV_3 (UA_3 = 22%). With UG_1, UG_2, and UG_3 utilization levels of 56%, 11%, and 70% respectively, UGV_2 becomes a master-slave link with UAV_3 and peer-to-peer links with UGV_1 and UGV_3 based on R8 and C9. As per R7, UAV_1 can only lead one follower UV, and UAV_5 can lead another, so UAV_2 is assigned to be led by UAV_1, and UAV_4 to be led by UAV_5.

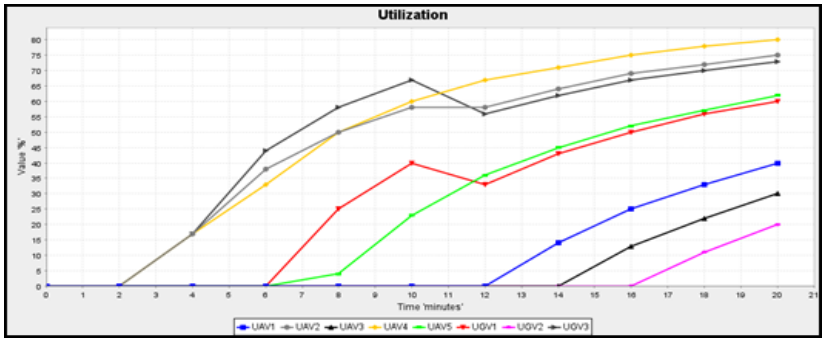

Figure 6: UVs Utilization in min

Table 1: UVs Traffic in Kbit

Test case	Time (min.)	Tr_{A1} (Kbit)	Tr_{A2} (Kbit)	Tr_{A3} (Kbit)	Tr_{A4} (Kbit)	Tr_{A5} (Kbit)	Tr_{G1} (Kbit)	Tr_{G2} (Kbit)	Tr_{G3} (Kbit)	Tr_{MCC} (Kbit)
1	2	0	0	0	0	0	0	0	0	0
2	4	0	0	0	800	0	0	0	0	1600
3	6	0	0	0	800	0	0	0	0	2400
4	8	0	0	0	800	800	0	0	0	4000
5	12	0	0	0	0	0	0	0	0	2400
6	14	800	800	0	0	800	0	0	0	4800
7	16	800	0	1600	0	800	0	0	0	5600
8	18	800	0	1600	0	800	0	0	0	6400
9	20	4000	0	5600	0	4000	1600	1600	1600	16000

6 DISCUSSION

The research highlighted the challenge of the ad-hoc scalability within a UVF from the SoS prospective. As the main ad-hoc scalability limitation is resulted from the static design of the UVF system architecture, the research proposed a dynamic system architecture that adapts its pattern (central, hierarchal, or holonic) based on the UVF demand. A multi-agent simulation has been introduced based on an analogy between the MAS and the UVF characteristics, under the SoS umbrella. The MAS simulation defined three software agent categories, which are UV agent, operator agent, and MCC agent.

The UV agent models its autonomous behavior by executing its state machine. This technique provided a realistic simulation leverage, as it enabled simulating the randomness in the UVs states. For example, during TC1, UGV_5 was available. However, upon C4 and R4, UGV_5 needed another UV that in the MCC range to be able to communicate with the MCC, which has been fulfilled in TC2. The operator agent enables the human decision making that dramatically influences the flow of events among the other agents. This has been illustrated in TC5, TC6, and TC9, when the operator commands the MCC to assign a specific architecture pattern. The operator decision making enables the flexibility and intelligence that cannot be afforded by algorithms, when dealing with uncertain situation such as cyber-attacks or partial system failure, that they are not designed to solve. The MCC agent has been implementation using Drools rule engine. Drools is particularly advantageous in applying forward and backward reasoning simultaneously (i.e., hybrid reasoning). Hybrid reasoning is especially important in automatic operation mode, as the MCC applies the forward

reasoning to construct an architecture based on the existing rules and contains. Then after constructing the architecture, it uses the backward reasoning to conclude which architecture pattern is assigned. For example, in TC8, the MCC concludes from R2 that the current pattern is hierarchal.

To verify the solution, test driven development technique has been followed. Accordingly, the illustrated test cases have covered all the rules and constrains. Ultimately, it is shown in Figure 6 that the UVs utilization is converging, as the rules tend to balance the overall utilization over time. Furthermore, it has been shown in Table 1 that the traffic is dramatically increases when switching between TC8 and TC9 from hierarchal to holonic pattern, that is of course due to the increase of the architecture layers and the peer-to-peer communication among the UVs. It is also noticed that switching to central pattern results in zero traffic, as shown in TC5. This means that the cost of a scalable, cooperative, and reliable UVF SoS, will be reflected on the overall traffic, utilization, and ultimately the UVs battery lifetime. Additionally, this is explaining why the existence of the human operator is very crucial to compromise all the UVF aspects and coupe with uncertain situations. During the future work, the utilization and traffic will be factorized and expressed as a function of the battery life. Thus, the rules can use both the utilization and the traffic simultaneously to optimize the UVF architecture pattern.

References

[1] Christian Albrecht, Meike Reimann, Ursula Rauschecker, Nicky Athanassopoulou, Simon Ford, Philippe Liatard, Stefan Eckert, Dolores Ordoñez, Maite Irigoyen, and Eusebio Bernabeu. Roadmaps and Recommendations for Strategic Action in the field of Systems of Systems in Europe Bitte Titelbild liefern.

[2] Godwin Asaamoning, Paulo Mendes, Denis Rosário, and Eduardo Cerqueira. 2021. Drone swarms as networked control systems by integration of networking and computing. Sensors 21. DOI:https://doi.org/10.3390/s21082642

[3] W. Clifton Baldwin and Brian Sauser. 2009. Modeling the characteristics of system of systems. In 2009 IEEE International Conference on System of Systems Engineering, SoSE 2009.

[4] Kathleen Giles. 2016. A Framework for Integrating the Development of Swarm Unmanned Aerial System Doctrine and Design. In Swarm-Centric Solution for Intelligent Sensor Networks, STO-MP-SET-222.

[5] Christian Goerick. 2010. Towards an understanding of hierarchical architectures. IEEE Trans. Auton. Ment. Dev. 3, 1 (2010), 54–63.

[6] Michael Henshaw, Carys Siemieniuch, Murray Sinclair, Vishal Barot, Sharon Henson, Cornelius Ncube, Soo Ling Lim, Huseyin Dogan, Mo Jamshidi, and Dan Delaurentis. 2013. The Systems of Systems Engineering Strategic Research Agenda Systems of Systems Engineering. 8th Int. Conf. Syst. Syst. Eng. Maui, Hawaii, USA - June 2-6, 2013 2 (2013).

[7] Hui Min Huang, Kerry Pavek, Brian Novak, James Albus, and Elena Messina. 2005. A framework for Autonomy Levels for Unmanned Systems (ALFUS). In AUVSI's Unmanned Systems North America 2005 - Proceedings.

[8] Praveen Kalla, Ramji K Ramj, and P Ravindranath. 2017. Swarm home robots. In 2017 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), 139–144.

[9] Jinkyu Kim, Teruhisa Misu, Yi-Ting Chen, Ashish Tawari, and John Canny. 2019. Grounding Human-To-Vehicle Advice for Self-Driving Vehicles. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

[10] Martin Kleppmann. 2019. Designing Data-Intensive Applications.

[11] Sujeet Kumar and Utkarsh Kumar. 2014. Java Agent Development Framework. International Journal of Research 1.

[12] Jérôme Leudet, François Christophe, Tommi Mikkonen, and Tomi Männistö. 2019. AILiveSim: An Extensible Virtual Environment for Training Autonomous Vehicles. In 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), 479–488. DOI:https://doi.org/10.1109/COMPSAC.2019.00074

[13] Shou-Chih Lo, Yi-Jen Lin, and Jhih-Siao Gao. 2013. A Multi-Head Clustering Algorithm in Vehicular Ad Hoc Networks. Int. J. Comput. Theory Eng. (2013). DOI:https://doi.org/10.7763/ijcte.2013.v5.686

[14] Hengbo Ma, Yaofeng Sun, Jiachen Li, Masayoshi Tomizuka, and Chiho Choi. 2021. Continual Multi-Agent Interaction Behavior Prediction With Conditional Generative Memory. IEEE Robot. Autom. Lett. 6, 4 (2021), 8410–8417. DOI:https://doi.org/10.1109/LRA.2021.3104334

[15] Mark W. Maier. 1998. Architecting principles for systems-of-systems. Syst. Eng. 1, 4 (1998). DOI:https://doi.org/10.1002/(SICI)1520-6858(1998)1:4%3C267::AID-SYS3%3E3.0.CO;2-D

[16] Irene Martin Rubio and Diego Andina. 2018. Smart Manufacturing in a SoSE Perspective. In Advances in Renewable Energies and Power Technologies. DOI:https://doi.org/10.1016/B978-0-12-813185-5.00015-2

[17] Mark Proctor. 2012. Drools: A Rule Engine for Complex Event Processing. In International symposium on applications of graph transformations with industrial relevance, 2–2. DOI:https://doi.org/10.1007/978-3-642-34176-2_2

[18] Ahmed R. Sadik and Christian Goerick. 2021. Multi-Robot System Architecture Design in SysML and BPMN. Adv. Sci. Technol. Eng. Syst. J. 6, 4 (2021). DOI:https://doi.org/10.25046/aj060421

[19] Ahmed R. Sadik, Andrei Taramov, and Bodo Urban. 2017. Optimization of tasks scheduling in cooperative robotics manufacturing via Johnson's algorithm case-study: One collaborative robot in cooperation with two workers. In Proceedings - 2017 IEEE Conference on Systems, Process and Control, ICSPC 2017. DOI:https://doi.org/10.1109/SPC.2017.8313018

[20] Ahmed R Sadik and Bodo Urban. 2016. A Novel Implementation Approach for Resource Holons in Reconfigurable Product Manufacturing Cell. In Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics, 130–139.

[21] Sharmila Sankar and V. Sankaranarayanan. 2012. A Predictive Route Maintenance Protocol Based on Signal Strength for Dense Ad Hoc Networks. Int. J. Comput. Theory Eng. (2012). DOI:https://doi.org/10.7763/ijcte.2012.v4.534

[22] Melanie Schranz, Martina Umlauft, Micha Sende, and Wilfried Elmenreich. 2020. Swarm Robotic Behaviors and Current Applications. Frontiers in Robotics and AI 7. DOI:https://doi.org/10.3389/frobt.2020.00036

[23] Jeffrey Smith, Jayashree Harikumar, and Brian Ruth. 2011. An Army-Centric System of Systems Analysis (SoSA) Definition. (October 2011), 38.

[24] Yahya M. Tashtoush and Omar A. Darwish. 2012. A Novel Multipath Load Balancing Approach Using Fibonacci Series for Mobile Ad Hoc Networks. Int. J. Comput. Theory Eng. (2012). DOI:https://doi.org/10.7763/ijcte.2012.v4.455