Received: 25 May 2018 | Revised: 23 November 2018 | Accepted: 3 January 2019

DOI: 10.1002/sys.21477

REGULAR PAPER

WILEY

A mission-based architecture for swarm unmanned systems

Kathleen Giles | Kristin Giammarco

Systems Engineering, Naval Postgraduate School, Monterey, California, USA

Correspondence
Kathleen Giles, Ph.D., Systems Engineering, Naval Postgraduate School, Monterey, CA 93943, USA.
Email: kbgiles@nps.edu

Abstract

This research applies a mission engineering approach with model-based systems engineering foundations to formalize a swarm unmanned system design methodology and architecture. The architectural framework and methodology herein presented extend current swarm system design methods, which are primarily bottom-up approaches focused on the behavior of individual agents. We introduce a top-down, hierarchical approach with an overarching mission decomposed into phases, tactics, plays, and algorithms. Three unmanned aerial vehicle swarm case studies (one of which is discussed in this paper) are used to demonstrate the approach and its effectiveness in formalizing a mission-based framework of common patterns in swarm missions that promote architecture reusability.

KEYWORDS

mission engineering, model-based systems engineering, swarm

1 | INTRODUCTION

Unmanned systems are developing rapidly in the government and private sectors, leading to a surge of interest in swarm system technology. Swarm systems are "large numbers of relatively simple, physically embodied agents [that] can be designed such that a desired collective behavior emerges from the local interactions among agents and between the agents and the environment."[1] Swarm agents may be unintelligent and ineffective as individuals, but interactions among agents have the potential to produce beneficial emergent behavior enabling system-level robustness, flexibility, and scalability.[1]

Swarm systems present a considerable design challenge from an operational perspective. Most researchers designing swarm systems do not come from a systems engineering background; rather their expertise lies in computer science. The general focus is often at the detailed design level without an overarching design architecture to provide a mission context. It is essential to connect the mission with the detailed design in order to produce an operationally suitable product. In Ref. 2, Scharre asserts that "the biggest challenge in adopting multivehicle control is not technical, but rather understanding the cognitive demands placed on the human operator and how many vehicles can be effectively controlled." Due to the inherent autonomous properties of swarm systems, a swarm commander will be required to manage and control swarm systems at a higher level of abstraction than is allowed for by current designs. To enable the swarm commander to focus on the mission, incorporating autonomy into both multivehicle control and tactical decision making is necessary.[2] An architectural framework that incorporates multiple levels of autonomy, particularly for "decision selection" and "action implementation" in Parasuraman's model[3] enables element reuse across missions.

Considering the scope of control a swarm commander exercises over the swarm system and the mission, we apply the concept of mission engineering to swarm system missions. Mission engineering is the "deliberate planning, analyzing, organizing, and integrating of current and emerging operational and system capabilities to achieve desired warfighting mission effects."[4] Mission engineering applies system integration methods to define a mission capability in terms of mission functions and treats the *mission* as the system.[4,5] By focusing on the swarm system from a mission perspective, naval doctrine is instituted as a primary design factor for architecting the system. Doctrine provides a standardized conceptual framework for connecting strategy, operations, and tactics, and is influenced by technology, organizational structure, the adversary's capabilities, and physical environment.[6] Model-based systems engineering (MBSE) with life-cycle modeling language (LML) models is used to formally describe the swarm's mission and apply a mission engineering approach to swarm architecture design. Defining mission functions in a modular, composable architecture allows repeating behavior logic patterns to be incorporated and reused throughout the design, facilitating subsequent automation in implementation. Furthermore, offering a mission-focused swarm architecture at a higher level of abstraction facilitates operator engagement in the system architecting process. Experiments by Refs. 7 and 8 suggest that a playbook-type architecture provides operators with the flexibility to balance manual control and cognitive workload when handling unexpected conditions. With this research, we aim to contribute a first step toward

the long-term capability that will enable one individual (the swarm commander) to control multiple vehicles with far less cognitive load than that employed using current systems engineering design methods.

This research was inspired by the field experimentation conducted by the Naval Postgraduate School (NPS)'s Advanced Robotics Systems Engineering Laboratory (ARSENL) team. In 2015, ARSENL advanced unmanned aerial system field experimentation by autonomously launching, flying, and landing 50 unmanned aerial vehicles (UAVs).[9]

2 | RELATED WORK

2.1 | Bottom-up swarm system design

Agent-based modeling, finite state machine (FSM), and agent behavior based are bottom-up modeling methods frequently used in swarm system design.[10–12] Bottom-up modeling approaches focus on assembling subcomponents of systems to build more complex systems. This approach is advantageous from a modularity and composability perspective, but risks failure to meet higher level system requirements if the design process precedes high-level system architecture development. For that reason, combining bottom-up and top-down models is an established software development heuristic.[13]

Modeling agent-level interactions can provide valuable information about the emergent behavior inherent to swarm systems. Agent-based modeling is commonly used to model a swarm as a group of autonomous agents making decisions individually based on their assessment of the environment and according to a rule set.[14,15] McCune et al.[16] used agent-based modeling to investigate command and control of UAV swarms, Bonabeau simulated human systems using agent-based modeling methods, and Munoz[17] studied defensive UAV swarm employment using agent-based modeling. A swarm system's emergent behavior is a key design attribute to consider when developing a swarm system, creating swarm tactics, or devising an assessment methodology for a swarm system. While agent-based modeling can provide useful information regarding interactions within a system, the variability in agent definition and limited standardization in approaches make model verification difficult.[15]

FSMs (or finite state automata) are a common approach for modeling multi-vehicle autonomous system architectures.[18–21] Within an FSM architecture, each agent operates in one of several states at a given time. Trigger events, generated by on board sensors or environmental conditions, cause the agent to transition between states. This approach is applicable in developing military swarm systems as the states and triggers can be defined deterministically, which is necessary for high-risk mission events such as target strikes. Conversely, there may be other mission events, such as covert searching, in which some bounded degree of unpredictability is desired. In those cases, a probabilistic FSM can be used to permit different behaviors within a state or allow multiple transitions between states based on fixed or varying probabilities.[20,22] Task allocation and aggregation behaviors have also been accomplished using probabilistic FSM approaches.[18]

Behavior-based design, in which the individual agent's behavior is developed iteratively until the desired swarm behavior is acquired, is a typical swarm system design method. Behaviors may apply to individual agents, their environment, and groups of agents (often called *collective behaviors*).[22] Behaviors may also be categorized as higher level abstract behaviors and lower level primitive behaviors, or simply *primitives*.[23] The term "primitives" is borrowed from the computer science discipline and functions similarly in robotics literature; they act as building blocks for programming higher level functions.

A formative behavior-based design is Brooks' subsumption architecture, which uses a layered approach for controlling systems, and incorporates augmented FSM processors for managing inputs and outputs.[24] Brooks' key contribution was decomposing the robot control problem into behaviors rather than into functional modules, allowing higher levels of behaviors to subsume lower layers of less complex behavior. Nicolescu and Matarić presented a behavior-based hierarchical architecture for robots in which reusable primitive behaviors support a library of abstract behaviors.[23] Brambilla et al. categorized collective behaviors into spatial organization, navigation, collective decision making, and other collective behaviors such as fault detection and human-swarm interaction.[22] Spatially organizing and distributing agents within their environment is a critical function for avoiding collisions and enabling efficient use of sensors. The most fundamental swarm organizing behavior is frequently called aggregation in swarm robotics literature. From aggregation, more complex patterns and formations such as flocking can be assembled. *Consensus* is a typical method for achieving a collective decision among the swarm.[10,25,26]

2.2 | Top-down swarm system design

Top-down design models, in which the high-level functional elements are specified before decomposing lower level functions, are common in systems engineering. Activity, sequence, state machine, and use case diagrams are traditional systems engineering behavior modeling diagrams.[13] Top-down design approaches are less prominent in swarm system design. Most swarm systems have been developed using bottom-up development methods in which an individual agent's behavior is iteratively fine-tuned until the desired collective behavior is achieved, frequently called "code and fix." DeLoach et al.[27] developed the multi-agent systems engineering methodology for analyzing, designing, and producing heterogeneous multi-agent systems. Their method uses graph-based models to define the agents and interfaces. Brambilla et al.[10] proposed a property-driven, top-down design method that formally describes the features of the system the designer wants to realize. Their method includes four phases: formally stating system requirements by specifying the intended properties, creating an abstract macroscopic model and model checker to verify the properties, using the macroscopic model as a guide for implementing the system (macroscopic to microscopic transition), and testing the system using real robots. Many of the traditional systems engineering architecting techniques[28–30] are appropriate for swarm system design; however, they have not been widely applied to this relatively new problem space.

2.3 | Toward a mission-based composable architecture—playbooks

A key advantage of homogeneous swarm technology is that a swarm is designed to be composed of simple, modular, identical components.[1] These agents are not programmed for a specific role and do not operate under a centralized coordinating agent. Accordingly, the loss of an individual does not cause a significant decrease in system performance because another agent or agents can assume its duties. Thus, a homogeneous swarm can be adaptable, expendable, robust, and scalable.[1,31]

The idea of using a *playbook*, or collection of predefined action plans for multi-robot unmanned systems, is not new, and has been used in both UAV[32] and unmanned ground vehicles.[33] A playbook can be described as a "library of plans of action that are available for the operator to instantiate at various levels of detail, allowing various levels of autonomy for the agents."[34] Rather than controlling each sequence of actions, the catalog of predefined behaviors simplifies operation, reduces necessary communications, and synchronizes agent tasking.[34] Sheard et al. emphasize that "[complex system] structure cannot be described at a single level or with a single view; multi-scale descriptions are needed to understand complex systems."[35] Playbooks offer a means to organize these complex systems into more tractable components. The playbook approach used in this research may be considered as a specific type of the pattern-based approach described under the guided modeling section in Ref. 36.

Playbook-type architectures have shown potential for improving cognitive workload for operators controlling multivehicle systems. Squire et al. studied the effects of various multivehicle control architectures on human performance.[8] Experiments by Parasuraman et al. used a simplified version of the "Playbook" interface in the RoboFlag multivehicle simulation environment that incorporated a hierarchical task model to task robots to perform various functions under temporal and conditional constraints.[7] The authors evaluated the effects of three levels of automation on human performance under defensive, offensive, and mixed conditions using subjective measures to rate mental workload and situational awareness, and performance metrics to capture mission completion time and mission success rate.[8] The results of the study suggest that a playbook design provides operators with the necessary flexibility for balancing manual control and cognitive workload when managing unexpected conditions.[8] Furthermore, the playbook system architecture appears promising for overcoming the "single robot parenting" paradigm.

Researchers have used the playbook approach to develop modular, composable architectures. McLurkin's library of behaviors for swarm robots is designed to be used as building blocks for more complex tasks.[37] His approach focuses on behaviors at the group level to facilitate distributed system programming by developing scalable, reusable behaviors that generate predictable outcomes. His architecture, influenced by Brooks' subsumption architecture[24], consists of a hierarchy of behaviors ranging from highest to lowest level: demos, group, pair, and primitive behaviors.[37] Although McLurkin's work focuses on ground robots and behaviors related to navigation, dispersion, and clustering, the philosophy is applicable to our research.

The "STP" (skills, tactics, and plays) multi-robot architecture developed by Browning et al. is particularly applicable to this research because it was designed to control a team of autonomous robots in an *adversarial* environment: RoboCup robot soccer.[38] Though our proposed architecture terms sound similar to Ref. 38, they are different in terms of hierarchy and function, and STP's elements are categorized temporally. Browning et al. define *skills* as "encoded low-level single-robot control algorithms for executing a complex behavior to achieve a short-term objective" (300 ms-5 s time period), *tactics* delineate which skills the individual robot should execute to achieve a specific goal (1-30 s), while *plays* determine "how the team of robots should coordinate their execution of tactics in order to achieve the teams overall goals" (5-30 s).[38] Tactics, which include a set of acceptable parameters that are dependent upon the play, determine which state machine a robot will execute. In the STP architecture, tactics and skills can be decoupled from plays to support operating individual robots that perform different roles (ie, a heterogeneous system).

In 2017, Defense Advanced Projects Research Agency (DARPA) initiated a program called Offensive Swarm-Enabled Technology (OFFSET) to advance swarm technology by focusing on human-swarm teaming and swarm autonomy within a realistic gaming environment.[39] Their method also takes a hierarchical, composable approach to the swarm framework which is composed of a mission, tactics, primitives, and supporting algorithms.[39] This program focuses exclusively on the urban operational environment, with a goal of building a playbook of tactics to support uncooperative urban missions. DARPA's research is particularly relevant as it aims to bridge the gap between the operational level of control and the programming solution level.

3 | PROPOSED ARCHITECTURE

The architecture proposed in this research called mission-based architecture for swarm composability (MASC) is designed to be a composable model abstract enough to be applied across a range of missions. The architecture intends to integrate mission doctrine, utilize composable elements, demonstrate modularity across missions, and be intuitive to the human operator. These attributes are important for promoting reusability across missions in a conceptual-level architecture. For the purpose of this research, a swarm is defined as a group of 20 or greater individual, self-organized, homogeneous UAVs that perform a mission through local interactions under a decentralized control architecture.[31,40]

3.1 | Taxonomy

MASC describes a swarm mission in terms of reusable, modular templates of tactics and plays. The following UAV swarm mission taxonomy is used to describe the mission architecture, and includes the terms:

- A *swarm mission* describes the overall objective assigned to the UAV swarm. Example swarm missions include: maritime interdiction operations (MIO), search and rescue, intelligence, surveillance,

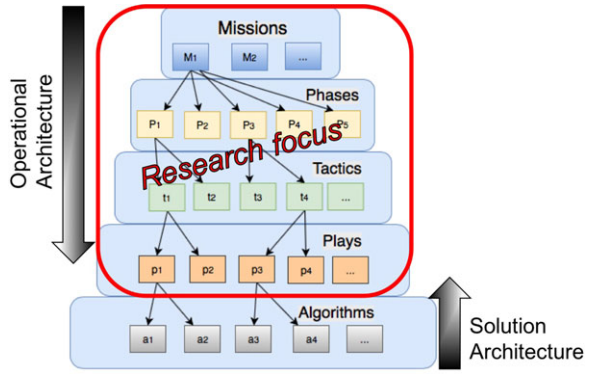

FIGURE 1 MASC framework. MASC is a many-to-many framework of elements starting with missions at the highest level. Each mission is composed of phases, tactics, plays, and algorithms at the lowest level

reconnaissance, and humanitarian assistance and disaster relief (HADR) . Each swarm mission includes five mission phases.

- A *swarm mission phase* defines a discrete time period within the mission. There are five phases in a swarm mission: Preflight, Ingress, OnStation, Egress, and Postflight. The three phases that cover the in-flight portion of the mission—Ingress, OnStation, and Egress—are the focus of this research. A swarm mission phase is composed of one or more swarm tactics.
- A *swarm tactic* is the employment and arrangement of agents (UAV) in relation to one another for the purpose of performing an assigned task.[41] Examples of swarm tactics include `Efficient search`, `Evade`, `Track`, and `Attack`. A swarm tactic is composed of one or more swarm plays and is designed to be used in multiple missions.
- A *swarm play* describes the lower level behaviors and maneuvers of the swarm as a group of agents.[41] The robotics community uses the term "behavior" to describe "a regularity in the interaction dynamics between the agent and the environment."[42] Swarm plays, the building blocks of swarm tactics, can be defined as behaviors with distinct triggers and temporal constraints. Swarm play examples include `Launch`, `Transit to waypoint`, `Orbit`, `Split`, and `Join`. Swarm play parameters are tunable attributes of a play that can be modified based on the mission or rules of engagement. A swarm play is designed to be used in multiple swarm tactics and is composed of one or more swarm algorithms.
- *Swarm algorithms* are the step-by-step procedures used by the controlling software to solve a recurrent task such as path planning, sorting, or foraging. Swarm algorithms are the building blocks of swarm plays. Swarm algorithms use data from the individual UAV such as heading, velocity, attitude, position, altitude, health status, and state.[41,43]

These solution-neutral behavior descriptions make up a modular framework for composing requirement specifications for missions employing different combinations of the same elements, and improved requirements for the algorithms needed to implement the mission. Figure 1 illustrates the MASC framework. A mission is decomposed into phases, which are composed of a library of tactics. Similarly, tactics are decomposed into plays, and plays are composed of algorithms. Within the context of the MASC framework, the swarm algorithms reside at the boundary wherein the operational architecture ends and the solution architecture begins. Swarm robotics research up to this point has focused on the algorithm and play (*behavior* or *primitive* in common parlance) levels. This research focuses on the operational part of the architecture—the missions, phases, tactics, and plays.

Figure 2 illustrates a generic swarm mission at the phase level. Each swarm mission is composed of the same five mission phases (Preflight, Ingress, OnStation, Egress, and Postflight). This research is primarily focused on the in-flight phases: Ingress, OnStation, and Egress. The Ingress and Egress phases are constructed similarly for each mission, with minor variations due to mission-specific rules of engagement. The major compositional differences occur during the OnStation phase, a result of the variety in mission objectives and required level of human involvement. Each phase decomposes into its corresponding tactics at the next lower level in the model, as indicated by the "decomposed" notation at the bottom of each box. Figure 2 represents most swarm missions at the phase level. Exceptions to this standard mission flow are missions in which the UAVs are deemed expendable and recovery is not planned. For such cases, the Egress and Postflight phases are absent.

3.2 | Methodology

The MASC elements described in the previous section are applied to three mission scenarios: MIO, HADR, and swarm v. swarm. This research focuses on the MIO example using the following methodology. The two other mission scenarios are detailed in Ref. 44. Figure 3 depicts the overall flow of the methodology.

1. Develop a high-level design reference mission that describes the swarm's role in the mission, includes the rules of engagement, the expected interactions of the key players, and the appropriate level of swarm automation for the scenario. Write a detailed narrative describing the key players' actions during each swarm phase from the temporal perspective of swarm.
2. Create a tactics level FSM depicting the desired transitions between swarm tactics.
3. Create an activity model simulation for the mission, beginning at the phase level (see Figure 4 for example). Then, decompose each phase into tactics from the swarm's temporal perspective using the model heuristics, such as allocating each activity to a performer, as further described in Ref. 44. Next, select the applicable plays from the options available, and modify the play parameters as required.
4. Run the activity model simulation to check for logic errors. Use Monterey Phoenix (MP) modeling (Section 4.3) to focus on specific interactions and to identify any potential undesired behaviors.
5. Review the implementation with the stakeholder.
6. Modify the tactics, plays, and activity model as necessary.
7. Incorporate the lessons learned from the methodology into future swarm doctrine and swarm system requirements.

FIGURE 2 Overall swarm mission flow. Each swarm mission is composed of five operational mission phases (generated using Innoslate)

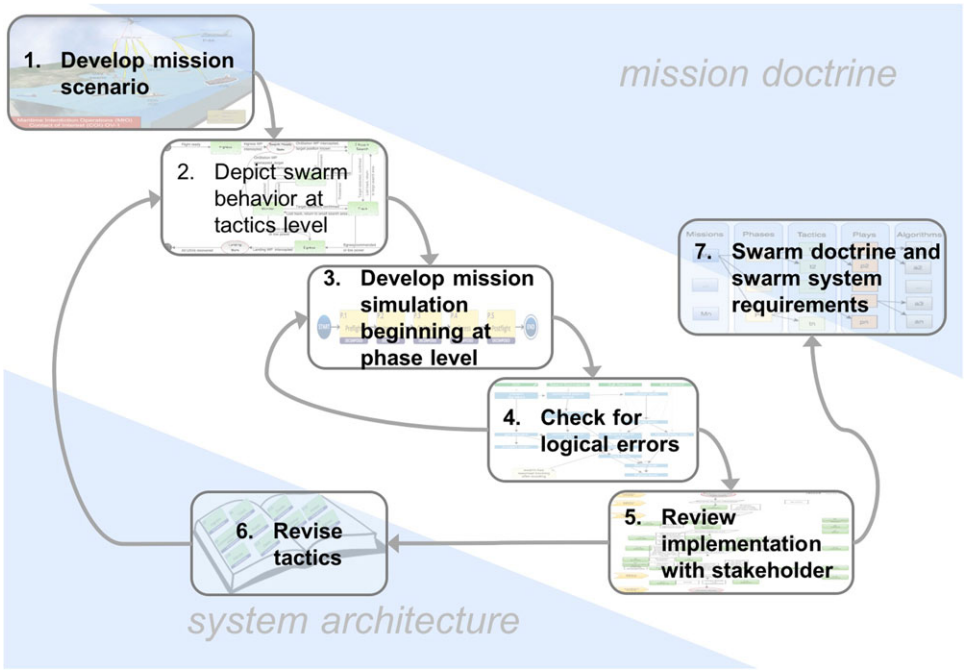

FIGURE 3 MASC is applied to a mission using a seven-step iterative process

4 | MASC APPLICATION TO A MISSION

In this fictional MIO scenario, the U.S. Navy is called upon to support the Indonesian Navy in combatting illegal shipping in the Sunda Strait. MIO is a U.S. Navy mission, typically executed with maritime air support that involves surveillance and interception of private or commercial vessels, boarding and searching of suspect vessels, and detaining, diverting, or seizing vessels found in violation of United Nations sanctions or other international laws.[45] The homogeneous UAV swarm consists of identical UAVs launched from a U.S. Navy destroyer, a ground control station, a launch system, and a recovery system. The notional swarm is capable of providing: streaming infrared and electro-optic video for detecting, classifying, and identifying targets during wide-area search and an air-to-ground radar for all-weather detection of surface vessels. Simultaneous voice relay and data-link communication are available over ultra high frequency (UHF), very high frequency (VHF), and commercial and military satellite systems. The swarm's primary mission is to assist with finding survivors and assessing infrastructure damage by providing real-time remote sensing data.

4.1 | MIO mission depicted as an FSM

The swarm's role in the MIO mission can be described using a tactics-level FSM. Figure 5 describes Subswarm 1's operations within the MIO scenario using the MASC framework. The swarm's initial state (s_0) is "on deck and preflighted" and the final state (F) is "on deck and recovered." The set of states (S) includes the `Ingress`, `Efficient search`, `Track`, `Monitor`, `Evade`, and `Egress` tactics (represented by green boxes), and the Swarm Ready and Landing readiness states. The inputs (Σ) are depicted in natural language describing the transition that occurs between the states. Finally, the transition functions ($\delta : S x \Sigma \Rightarrow S$) are the mappings of inputs to original states which result in a subsequent state change. For example, δ*(Efficient Search, Target detected)* = *Track* describes the automatic transition from `Efficient search` to `Track` tactic if a potential target of interest is detected. The `Evade` tactic is enacted automatically if the swarm senses it is being attacked by the adversary. For this scenario, the swarm operates at a low level of automation, such that a target must be confirmed by the Swarm Commander as "of interest" for it to be monitored.

4.2 | MIO mission simulation

The swarm's activities during the OnStation phase are depicted as an LML action diagram in Figure 6. Two main parallel branches show the swarm's tactics employment, as designated by a swarm commander during mission planning, in the top branch and the swarm's power consumption in the bottom branch. Within the swarm's tactics branch, two parallel activity branches depict the swarm dividing into two subswarms and executing their respective tactics. Subswarm 1 performs the searching, tracking, and monitoring functions, while subswarm 2 maintains contact with the other participating units via the `Communication relay` tactic. Figure 6 is a simple example of how to employ a

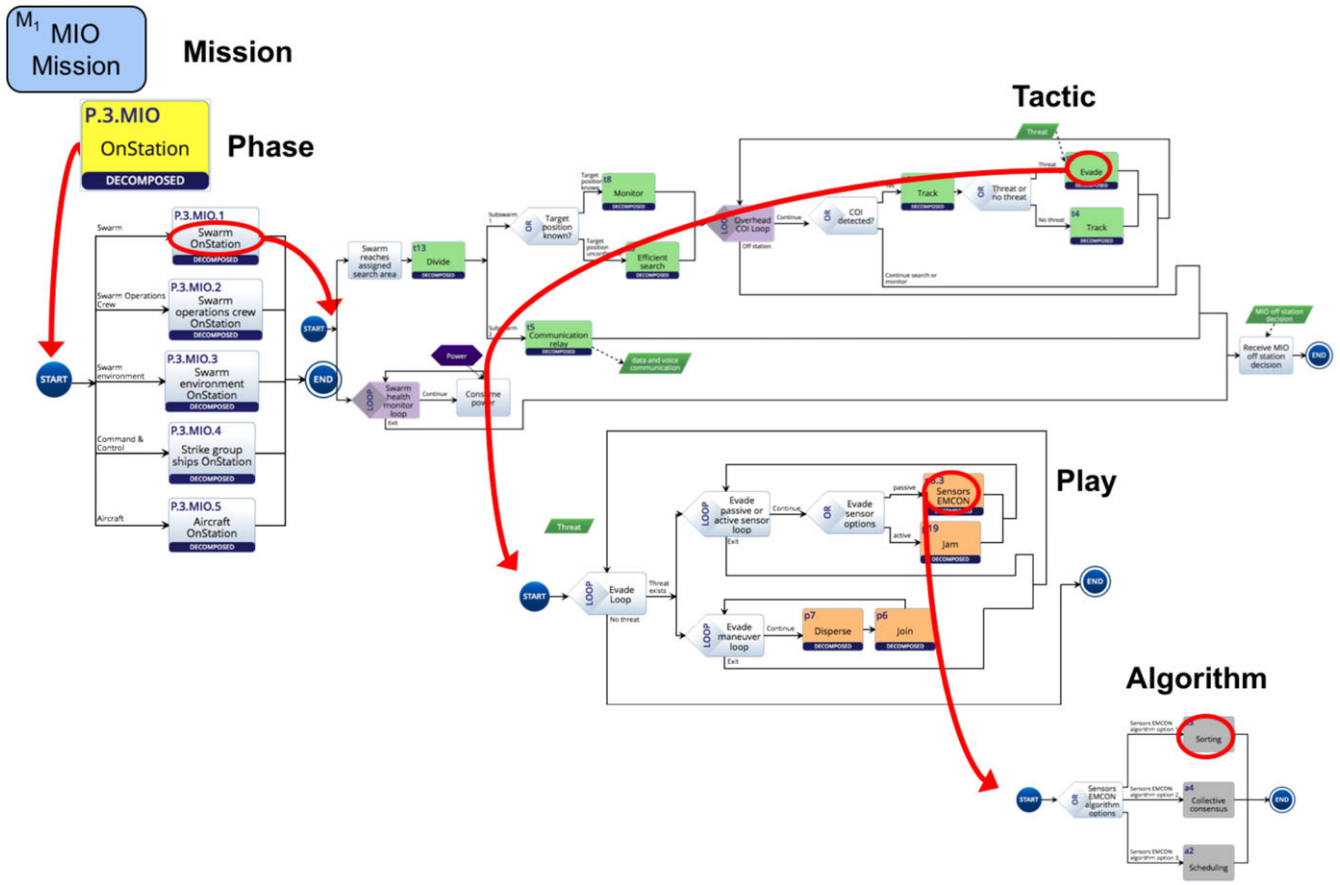

FIGURE 4 The composability of a mission is shown using the MIO mission as an example. The mission is composed of phases, which are composed of tactics, plays, and algorithms. The OnStation is used as the example. The mission runs as a discrete event or Monte Carlo simulation in Innoslate

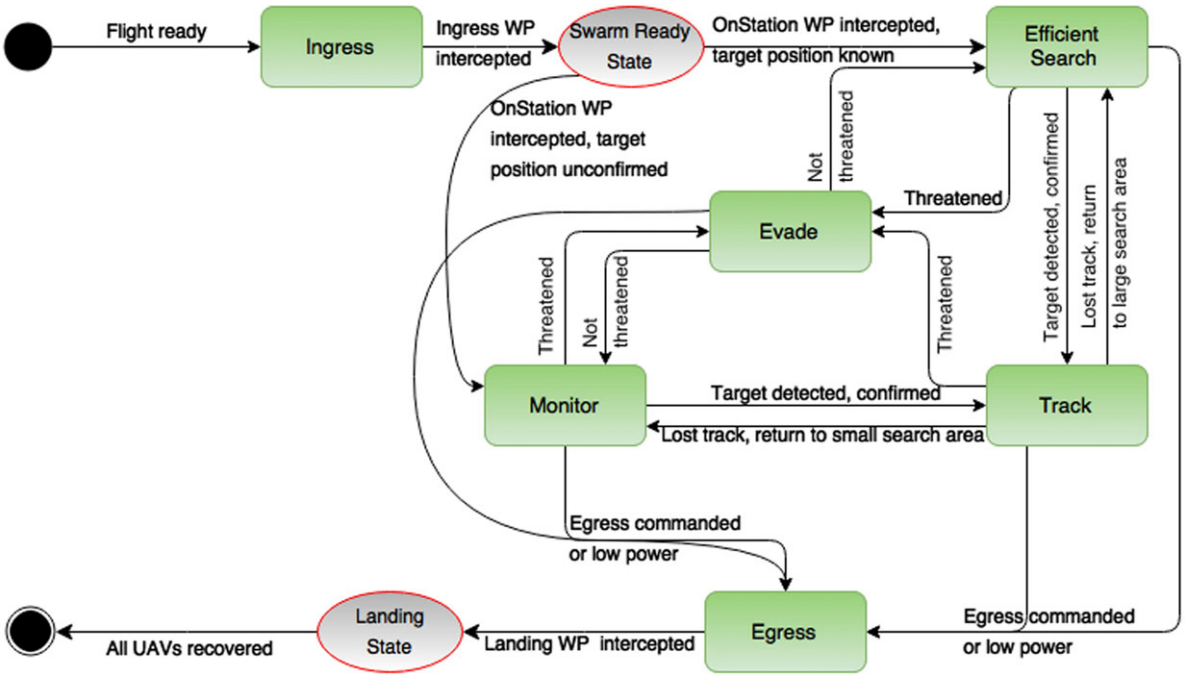

FIGURE 5 State machine diagram depicting tactics used by UAV Subswarm 1 operating in the MIO mission

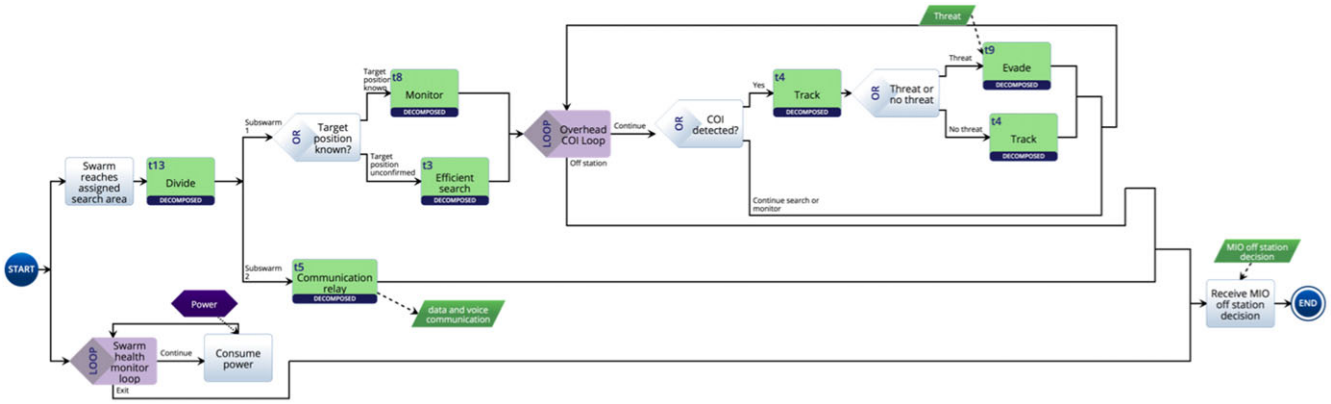

FIGURE 6 Action diagram for a UAV swarm operating in an MIO mission during the OnStation phase. The upper branch shows the tactics in green and the lower branch captures power consumption by the system (developed in Innoslate[46])

UAV swarm in an MIO mission using swarm tactics. A natural extension of this pattern to support swarm doctrine for wide-area search is to use multiple subswarms to conduct reconnaissance over different areas and at varying altitudes. This becomes especially advantageous in regions with varying terrain where widely dispersed subswarms assigned to relay communications support improved connectivity between coordinating units.

4.3 | Checking for logical errors

The MIO action diagram developed using the Innoslate software tool was also executable as a simulation, which proved useful for logic checks and troubleshooting. Infinite loops, broken connections between tactics, and incomplete interactions between components could be detected immediately. MP was used in a parallel design effort to closely examine the potential behaviors of the different scenario players and the implications of tactics combinations. MP is an NPS-developed formal language and method for modeling operational processes and system behaviors.[47] MP enables behavior specification for each system component or agent, separate from the interactions among the agents. Events are described in terms of precedence and inclusion relationships, and behaviors may be depicted as alternative, concurrent, iterative, or optional actions. A benefit of MP over other modeling language is that it generates an exhaustive set of use cases that are visually depicted;[48] this is particularly useful for complex system modeling and dealing with the state explosion problem. However, even a relatively simple system's state space exceeds a human's ability to extrapolate every possible combination of outcomes. MP augments other top-down behavior models such as activity and state machine diagrams by generating a complete set of use cases and providing automated assertion checking capability, thus enabling a means for identifying unwanted behavior and validating system behavior models.[49]

The MP model facilitated identification of potential undesired behaviors or logical flaws in transitions between tactics. Figure 7 shows an event trace from a model containing the roots: contact of interest (COI), the Swarm Commander, and a UAV swarm depicted as Subswarm 1 and Subswarm 2. This model characterizes the subswarms as root events that behave according to their assigned tactics. The tactics are composite events which include the plays. For this iteration, the model was run at the tactics level to focus on tactics transitions. Of the 11 use cases, there were two potential cases of undesired behavior in which Subswarm 1 initiated `Track` after `Evade` while the COI was still threatening the swarm. The code could be modified such that the Subswarm 1 only initiates the `Track` tactic (in cases following `Evade`) if it is not threatened by the COI. On the other hand, overconstraining the interaction rules between the tactics or between the roots resulted in desired use cases not being generated. MP provided a quick and effective means to experiment with different options for interactions that could be fed back into the Innoslate simulation model. MP also raised some usability-related questions into handling the transition of swarm tactics when the swarm detects a target: should the swarm automatically begin tracking a target and prompt the Swarm Commander to determine if the target is "of interest"? Should the swarm automatically return to `Efficient search` if the target is not of interest? Should `Track` be a play-level function instead of a tactic? This question was debated several times by the authors, and eventually, `Track` was categorized as a tactic to support reusability and modularity among the missions. In the future MASC versions, `Track` may be redesigned as a play.

4.4 | Review the implementation with the stakeholder

In addition to evaluating MASC within the confines of the models and simulations, human subjects research augmented the evaluation by gathering prospective user feedback on the swarm tactics and plays using a table-top exercise and the MIO scenario script . The purpose of the human subjects exercise was to collect feedback from subject matter experts and to determine if there was a difference in operator-perceived workload between mission planning at the tactics level versus at the play level. The 15 volunteers, targeted for their maritime aviation experience, were randomly assigned to two groups: group 1 developed a UAV swarm mission plan for the notional MIO scenario using the 16 available swarm tactics, while group 2 developed a plan using only the 27 available swarm plays and without knowledge of the swarm tactics. Following the swarm mission plan construction,

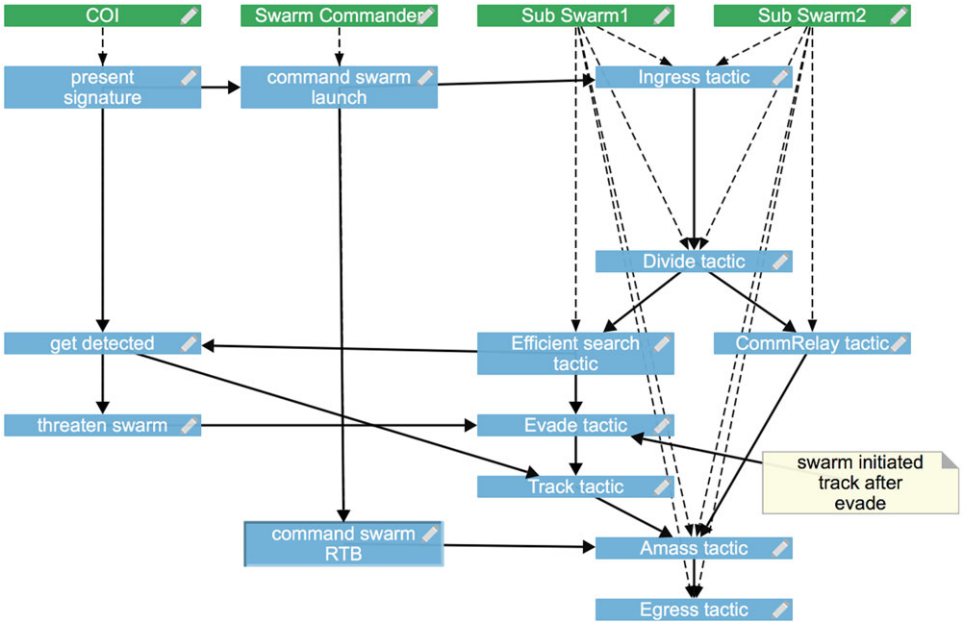

FIGURE 7 MP uses case for the MIO mission showing the main tactics-level interactions between COI, Swarm Commander, Subswarm 1, and Subswarm 2

each participant responded to a series of questions designed to support measuring MASC operator usability. Quantitative data were collected using the NASA Task Load Index method[50] to assess participant perception of their task workload, along with the time required to complete the task. We reviewed the selection of tactics or plays, their placement within the phases of the mission, and their relation to the mission triggers in the scenario. Participant responses to questions about the architecture's structure, and type of tactics and plays offered were also reviewed to assess usability. There was no statistically significant difference between the two groups in this small sample size regarding task completion time and perceived workload. However, valuable qualitative feedback collected from the participants was incorporated into MASC. Two tactics suggested by group 1, "Disguise RHIB" and "Disrupt Fire," closely resembled the existing `Deter` tactic. However, using the swarm as a visual shield for camouflage was not considered in the original playbook. When specifically asked if predefined combinations of plays would make the task easier, seven of eight group 2 participants responded positively, and two identified `Ingress` and `Egress` combinations specifically. Another participant grouped `Follow target` and `Smart greedy shooter` for a "Harassment" effect much like the existing `Harass` tactic.

5 | DISCUSSION AND CONCLUSIONS

After applying the MASC framework to notional UAV swarm missions, MASC was evaluated for its effectiveness in formalizing a mission-based framework of common patterns in swarm missions that promote architecture reusability. MASC's modularity, composability, and mission doctrine integration were assessed to support the evaluation.

MASC's modularity and composability were evaluated as supporting attributes for architecture reusability. Figure 4 is an example of a swarm mission composed of its MASC architecture elements. Beginning in the upper left, an MIO mission is composed of five phases, of which only OnStation is depicted. The OnStation phase is further broken down into the activities of the primary participants. The tactics level is next and shows the swarm's activities described in terms of swarm tactics (green rectangles). The `Evade` tactic is circled and decomposed into its swarm play elements (orange rectangles), and finally the `Sensors EMCON` play is shown in terms of the algorithms available to support its implementation.

The MASC framework demonstrated modularity in the phases, tactics, and plays that were used across different missions and within each mission. Table 1 shows the distribution of tactics used for the mission scenarios. MASC's modular design facilitated its composability as demonstrated in the semantically logical LML action diagram simulations. The MASC framework was modified multiple times during the mission case study model development. Phases were not part of the original framework, but added later to facilitate tactic and play reuse across missions. The `Deter` tactic was added later in development to support mission scenarios in which rules of engagement forbid UAV weapon carriage. This research was developed under the assumption of a swarm composed of small (DoD category 1–2), fixed-wing UAV to support the endurance requirements of the MIO and HADR scenarios. However, the same type of framework could be applied to smaller fixed-wing and rotary aircraft. Adjustments for platform performance and capability differences can be accounted for in the play parameters.

Mission doctrine integration was demonstrated in the common mission patterns that were applied across missions in the simulations and by the human subjects research participants. The three mission case studies were construed using the same mission phases and the tactics were designed to support a diverse mission set by varying the

TABLE 1 Table showing the distribution of tactics used across the three mission case studies

Tactic	Swarm v. Swarm	MIO	MIO HSR	HADR
Ingress	B	B	7	B
Evasive search	B		4	
Efficient search		B	5	B
Track	B	B	7	B
Communication relay		B	7	B
Attack	B			
Battle damage assessment	B		3	
Monitor		B	7	B
Evade	B	B	7	
Harass			3	
Defend			1	
Deter			2	
Divide		B	7	B
Amass		B	6	B
Egress	B	B	6	B
Option	N/A	N/A	1	N/A

"B" indicates a tactic used by the author in the Innoslate baseline model. The numbers in the fourth column indicate the number of Group 1 participants (7 total) who selected the tactic during the human subjects. The MIO rules of engagement did not allow the participants to use Attack

play parameters according to mission-specific rules of engagement and tactics, techniques, and procedures. A common pattern within the OnStation phase was dividing the swarm into subswarms for distributing capabilities between intelligence collection and communication relay using the `Divide` and `Amass` tactics. Several of the commonly used tactics patterns have the potential to support another level of MASC elements *above* the tactics level. For example, a predefined combination of tactics (established during mission planning) dividing the swarm into a close-in monitoring subswarm, a subswarm monitoring a threat from a safe standoff distance, and a third subswarm performing communication relay could be called "MIO Surveillance 1."

A significant finding for MBSE practitioners interested in the patterns-based approach[36] is the elegant execution of the playbook structure in simulation, using and reusing selected plays in different tactics, and selected tactics in different missions simply by adding the same parent activity (named for the tactic or play) to different mission models, and running the mission model to "call" the lower level models in the mission simulation. The MIO mission model example in Figure 4 illustrates how this nesting of behavior patterns was done in LML executable models. The UAV swarm models using the playbook structure contribute a domain application example that can, in future work, be accompanied by other examples in different domains and in modeling languages other than LML that implement the same type of nesting structure. A solution-neutral, top-down behavior architecture composed of reusable modules that execute in simulation brings the benefits of architecture reuse into practicing organizations, and lays the foundation for subsequent analysis using those models.

6 | FUTURE WORK

This exploratory research lays the ground work for future experiments with clearly defined control groups. Swarm system design is so new that there is no established history for comparing approaches. There are several areas of future work in this research domain. These include reconsidering the appropriateness of the architecture element levels, incorporating MASC into a prototype virtual environment, developing swarm system measures of performance, and using MASC to define swarm system test cases. As previously mentioned, an improvement to MASC may consist of redesigning the tactics level to a higher level of abstraction providing a greater distinction between tactics and plays. Employing MASC into an interactive simulation environment is necessary for progressing swarm doctrine, experimenting with tactics combinations, and expanding mission sets. The simulation environment could be used to develop a graphical user interface suitable for swarm missions, incorporate adversary activities, and continue to gather stakeholder response throughout system development. Developing a swarm system graphical user interface that leverages the strengths of humans and machines as a team is an important research area for swarm systems. Swarm system measures of performance are needed to support acquisition. These measures could be developed based on joint capability area attributes such as timeliness, latency, survivability, connectivity, stealth, endurance, strike, expeditionary, and interoperability to assess that algorithms are meeting play objectives, plays are meeting tactics objective, etc. Finally, this research focused on the operational architecture—the missions, phases, tactics, and plays—rather than the solution architecture. The algorithms mentioned in this research exist in ARSENL behaviors or other robotics applications. The swarm algorithms reside at the boundary at which the operational architecture and the solution architecture meet. There is much work to be done in assigning the right algorithm to each swarm play.

ORCID

Kathleen Giles https://orcid.org/0000-0002-9908-4632

REFERENCES

1. Sahin E. Swarm robotics: from sources of inspiration to domains of application. In: *International Workshop on Swarm Robotics*. 631. Santa Monica, CA: Springer Berlin Heidelberg; 2005.
2. Scharre P. *Robotics on the Battlefield Part II The Coming Swarm*. Washington, DC: Center for a New American Security; 2014.
3. Parasuraman R, Sheridan TB, Wickens CD. A model for types and levels of human interaction with automation. *IEEE Trans Syst Man Cybern A*. 2000;30:286–297.
4. Gold R. *Mission Engineering*. Springfield, VA: Office of the Deputy Assistant Secretary of Defense for Systems Engineering; 2016.
5. Beam DF. *Systems Engineering and Integration as a Foundation for Mission Engineering* [M.S. thesis]. Monterey, CA: Naval Postgraduate School; 2015.
6. Tritten JJ. Naval perspectives for military doctrine development. Naval Doctrine Command; 1994.

7. Parasuraman R, Galster S, Miller C. Human control of multiple robots in the RoboFlag simulation environment. In: *IEEE International Conference on Systems, Man and Cybernetics*. vol. 4. Washington, DC: IEEE; 2003:3232–3237.
8. Squire PN, Galster SM, Parasuraman R. The effects of levels of automation in the human control of multiple robots in the Roboflag simulation environment. In: Vincenzi DA, Mouloua M, Hancock PA, eds. *Human Performance, Situation Awareness and Automation: Current Research and Trends*. vol. 2. Daytona Beach, FL: Embry-Riddle Aeronautical University; 2004:48–53.
9. Chung TH, Clement MR, Day MA, Jones KD, Davis D, Jones M. Live-fly, large-scale field experimentation for large numbers of fixed-wing UAVs. In: Okamura A, ed. *Proceedings of the IEEE International Conference on Robotics and Automation*. Stockholm, Sweden: IEEE; 2016:1–8.
10. Brambilla M, Pinciroli C, Birattari M, Dorigo M. Property-driven design for swarm robotics. In: *AAMAS International Conference on Autonomous Agents and Multiagent Systems*. June. Valencia: International Foundation for Autonomous Agents and Multiagent Systems; 2012:139–146.
11. Maza I, Ollero A, Casado E, Scarlatti D. Classification of multi-UAV architectures. *Handbook of Unmanned Aerial Vehicles*. Springer: Dordrecht; 2015:953–975.
12. Arai T, Pagello E, Parker LE. Guest editorial advances in multirobot systems. *IEEE Trans Robot Autom*. 2002;18:655–661.
13. Buede D. *The Engineering Design of Systems: Models and Methods*. 2nd ed. Hoboken, NJ: John Wiley & Sons, Inc.; 2009.
14. Bonabeau E. Agent-based modeling: methods and techniques for simulating human systems. *Proc Natl Acad Sci USA*. 2002;99:7280–7287.
15. Borshchev A, Filippov A. From system dynamics and discrete event to practical agent based modeling: reasons, techniques, tools. In: *The 22nd International Conference of the System Dynamics Society*. vol. 22. Oxford, England: Wiley; 2004:5–6.
16. McCune R, Purta R, Dobski M, et al. Investigations of DDDAS for command and control of UAV swarms with agent-based modeling. In: *Proceedings of the 2013 Winter Simulation Conference: Making Decisions in a Complex World*. Washington, DC: IEEE; 2013:1467–1478.
17. Munoz MF. *Agent-Based Simulation and Analysis of a Defensive UAV Swarm Against an Enemy UAV Swarm* [M.S. thesis]. Monterey, CA: Department of Operations Research, Naval Postgraduate School; 2011.
18. Soysal O, Sahin E. Probabilistic aggregation strategies in swarm robotics systems. In: *IEEE Swarm Intelligence Symposium*. Pasadena, CA: IEEE; 2005.
19. Martinoli A, Easton K, Agassounon W. Modeling swarm robotic systems: a case study in collaborative distributed manipulation. *Int J Robot Res*. 2004;23:415–436.
20. Parunak HVD. Making swarming happen. In: Inbody D, Chartier C, DiPippa D, McDonald B, eds. *Conference on Swarming and C4ISR*. Tysons Corner, VA: Joint C4ISR Decision Support Center; 2003.
21. Weiskopf F, Gion T, Elkiss D, et al. Control of cooperative, autonomous unmanned aerial vehicles. In: *AIAA's 1st Technical Conference and Workshop on Unmanned Aerospace Vehicles*. 1. Portsmouth, VA: AIAA; 2002:1–5.
22. Brambilla M, Ferrante E, Birattari M, Dorigo M. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intellig*. 2013;7:1–41.
23. Nicolescu MN, Matarić MJ. A hierarchical architecture for behavior-based robots. In: *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems Part 1*. 2002, pp. 227–233.
24. Brooks RA. A robust layered control system for a mobile robot. Boston, MA: Massachusetts Institute of Technology; 1985. 864f.
25. Choi HL, Brunet L, How JP. Consensus-based decentralized auctions for robust task allocation. *IEEE Trans Robot*. 2009;25:912–926.
26. Davis DT, Chung TH, Clement MR, Day MA. Consensus-based data sharing for large-scale aerial swarm coordination in lossy communications environments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Daejeon, Korea: IEEE; 2016:3801–3808.
27. DeLoach SA, Wood MF, Sparkman CH. Multiagent systems engineering. *Int J Softw Eng Knowl Eng*. 2001;11:231–258.
28. Cloutier RJ, Verma D. Applying the concept of patterns to systems architecture. *Syst Eng*. 2007;10:138–154.
29. Szajnfarber Z, Vrolijk A. A facilitated expert-based approach to architecting openable complex systems. *Syst Eng*. 2018;21:47–58.
30. Sheard S, Mostashari A. Principles of complex systems for systems engineering. *Syst Eng*. 2009;12:295–311.
31. Beni G. From swarm intelligence to swarm robotics. In: *International Workshop on Swarm Robotics*. Santa Monica, CA: Springer Berlin Heidelberg; 2004:1–9.
32. Goldman RP, Miller CA, Funk HB, Meisner J. Optimizing to satisfice: using optimization to guide users. In: *Proceedings of the American Helicopter Society's International Specialists Meeting on Unmanned Aerial Vehicles*. January. Phoenix, AZ: AHSI International, Inc.; 2005:18–20.
33. Simmons R, Apfelbaum D, Fox D, et al. Coordinated deployment of multiple, heterogeneous robots. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Takamatsu, Japan: IEEE; 2000:127–140.
34. Coppin G, Legras F. Autonomy spectrum and performance perception issues in swarm supervisory control. In: *Proceedings of the IEEE*. vol. 100; 2012:590–603.
35. Sheard S, Cook S, Honour E, et al. A complexity primer for systems engineers. *INCOSE Complex Systems Working Group*. 2015, pp. 1–17.
36. Madni AM, Sievers M. Model-based systems engineering: motivation, current status, and research opportunities. *Syst Eng*. 2018;(April):172–190.
37. McLurkin JD. *Stupid Robot Tricks: A Behavior-Based Distributed Algorithm Library for Programming Swarms of Robots* [M.S. thesis]. Boston, MA: Department of Electrical Engineering and Computer Science, MIT; 2004.
38. Browning B, Bruce J, Bowling M, Veloso M. STP: skills, tactics, and plays for multi-robot control in adversarial environments. *Proc Inst Mech Eng Part I: J Syst Contr Eng*. 2004;219:33–52.
39. DARPA TTO. *Broad Agency Announcement (BAA) Offensive Swarm Enabled Tactics (OFFSET) Tactical Technology Office (TTO)*. Arlington, VA: DARPA TTO; 2017.
40. Beni G, Wang J. Swarm intelligence in cellular robotic systems. In: Dario P, Sandini G, Aebischer P, eds. *Robots and Biological Systems: Towards a New Bionics*? Berlin Heidelberg: Springer-Verlag; 1993:703–712.
41. Chung TH. *Dr. Chung directed study notes 1*. Monterey, CA; 2015.
42. Matarić MJ. Issues and approaches in the design of collective autonomous agents. *Robot Auton Syst*. 1995;16:321–331.
43. Chung TH. *Dr. Chung directed study notes 2*. Monterey, CA; 2015.
44. Giles K. *Mission-Based Architecture for Swarm Composability* [Ph.D. dissertation]. Monterey, CA: Department of Systems Engineering, Naval Postgraduate School; 2018.
45. Operations CoN. *Visit, board, search, and seizure operations (NTTP3-07.11M)*. Norfolk, VA: Department of the Navy; 2013.
46. Innovations S. Innoslate; 2017. Available from: www.innoslate.com. Last accessed May 22, 2018.
47. Auguston M. *System and software architecture and workflow modeling language manual*. Monterey, CA: Naval Postgraduate School; 2017.

48. Giammarco K, Troncale L. Modeling isomorphic systems processes using monterey phoenix. *Systems*. 2018;*6*:18.
49. Giammarco K, Giles K. Verification and validation of behavior models using lightweight formal methods. In: Madni AM, Boehm B, Erwin DA, Ghanem R, Wheaton M, eds. *15th Annual Conference on Systems Engineering Research*. Redondo Beach, CA; 2017:444–445.
50. Hart SG, Staveland LE. Development of NASA-TLX (Task Load Index): results of empirical and theoretical research. *Adv Psychol*. 1988;52:139–183.

AUTHORS' BIOGRAPHIES

Kathleen Giles is an Assistant Professor in the Department of Systems Engineering at the Naval Postgraduate School, where she teaches courses in systems test and evaluation and government acquisition, and conducts research in UAV swarms. Dr. Giles is a member of INCOSE. She has earned a Ph.D. in Systems Engineering from NPS, an M.S. in Technical Management from Johns Hopkins University, and a B.S in Oceanography from the U.S. Naval Academy.

Kristin Giammarco is an Associate Professor in the Department of Systems Engineering at the Naval Postgraduate School, where she teaches courses in system architecture and design, system integration, systems software engineering, and model-based systems engineering, and conducts research in the use and development of formal methods for systems architecture modeling. Dr. Giammarco is a member of INCOSE and the International Society for Systems Pathology (ISSP), and serves as the Joint Executive Systems Engineering Management (SEM-PD21) Program Academic Associate. She has earned a Ph.D. in Software Engineering, an M.S. in Systems Engineering Management, and a Certificate in Advanced Systems Engineering from NPS. She holds a B.E. in Electrical Engineering from Stevens Institute of Technology.

How to cite this article: Giles K, Giammarco K. A mission-based architecture for swarm unmanned systems. *Systems Engineering*. 2019;22:271–281. https://doi.org/10.1002/sys.21477