

Article

Metamodel-Driven Modeling of UAF-Based Cooperative Drone Combat Systems with AutoCL-MAPPO

Yimin Feng ^{1,*} , Yuting Li ¹, Pengwei Zhang ^{1,*}, Jingxia Chen ¹, Guanhui Zhao ^{2,3}, Yusheng Liu ⁴, Hongyu Li ⁵ and Yaguang Huang ⁵

¹ School of Electronic Information and Artificial Intelligence, Shaanxi University of Science & Technology, Xi'an 710021, China; 241612114@sust.edu.cn (Y.L.); jingxiachen@sust.edu.cn (J.C.)

² School of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China; 12321305@zju.edu.cn

³ China Ship Research and Design Center, Wuhan 430064, China

⁴ State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310027, China; yslu@cad.zju.edu.cn

⁵ Hangzhou Hope System Technology Co., Ltd., Hangzhou 310030, China; ihy.ast@gmail.com (H.L.); huangyaguang@mbse.com.cn (Y.H.)

* Correspondence: fengyimin@sust.edu.cn (Y.F.); zhangpengwei@sust.edu.cn (P.Z.)

Highlights

Please indicate how your work links to systems science via your contributions to systems practice, theory, and or methodology.

- Establishes a five-stage Unified Architecture Framework (UAF) process combined with a Drivers–Challenges–Opportunities–Goals (DCOG) framework, providing a systematic methodology to translate top-level mission intent into structured operational specifications for system-of-systems design.
- Leverages UAF Domain Metamodel (DMM) semantic rules to drive the theoretical transformation of static UAF models into executable Markov Decision Processes (MDPs), bridging the semantic gap between qualitative architectures and quantitative decision making.

What are the main findings and/or the implications of the main findings?

- Demonstrated a 73.4% mission success rate in autonomous drone swarm combat under communication decay conditions using Automatic Curriculum Learning Multi-Agent Proximal Policy Optimization (AutoCL-MAPPO), significantly outperforming MAPPO (65%), HAPPO (50.8%), and QMIX (40.2%) baselines in cooperative adversarial scenarios.
- Validated learned decision sequences against Systems Modeling Language (SysML) architectural rules, enabling constraint deviation analysis as empirical feedback to iteratively refine early-stage system designs and close the digital thread between modeling and execution.



Academic Editors: Hidekazu Nishimura and James Martin

Received: 5 June 2026

Revised: 25 July 2026

Accepted: 28 July 2026

Published: 1 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Abstract

Traditional MBSE faces challenges in verifying the dynamic performance of autonomous systems due to a semantic gap between static architecture and executable algorithms. To address this limitation, this study proposes a system-of-systems (SoS) modeling methodology that connects the Unified Architecture Framework (UAF) with multi-agent reinforcement learning. A Drivers–Challenges–Opportunities–Goals (DCOG) framework maps strategic intent to operational capabilities, which are formalized in the UAF. Driven by the UAF Domain Metamodel (DMM), a pipeline transforms behavioral and resource specifications into a Markov Decision Process (MDP). An Automatic Curriculum Learning

Multi-Agent Proximal Policy Optimization (AutoCL-MAPPO) algorithm then resolves the MDP. The autonomous fleet achieves a 73% mission success rate, outperforming the standard MAPPO baseline (65%). These performance metrics are averaged over 1000 independent test episodes to ensure statistical significance, with baseline algorithms evaluated under identical environmental conditions. Using Systems Modeling Language (SysML) as a verification carrier, activity simulations confirm that the generated decision sequences conform to UAF structural logic. Metric constraint deviation analysis provides empirical feedback for iterative design refinement. This methodology establishes a verifiable digital thread, closing the loop between architecture modeling and learned behavior for autonomous SoS and Human–AI Teaming.

Keywords: system-of-systems; UAF; UAF domain metamodel; multi-agent reinforcement learning; Markov decision process; curriculum learning; digital thread

1. Introduction

The proliferation of unmanned aerial vehicle (UAV) technologies has fundamentally transformed modern warfare, driving a shift from platform-centric operations toward highly networked, collaborative system-of-systems (SoS) paradigms [1–3]. Within this context, the design and verification of UAV combat SoSs present two intertwined challenges. First, translating high-level strategic intent—expressed through architecture frameworks—into quantitative, executable models remains an open problem. Second, the dynamic and adversarial nature of aerial engagements demands that autonomous agents learn complex cooperative behaviors that static architecture models cannot capture [4,5].

Model-Based Systems Engineering (MBSE) and architecture frameworks provide a rigorous foundation for managing SoS complexity [6,7]. Among the available frameworks, the Unified Architecture Framework (UAF) stands out for its multidimensional, hierarchical description capability, covering strategy, operations, resources, and personnel through more than 80 standardized viewpoints [8–10]. When combined with the Systems Modeling Language (SysML), the UAF enables the formal specification of operational activities, resource allocations, and behavioral sequences. However, existing UAF-based methodologies are predominantly static; they excel at documenting system structure but lack the mechanisms for validating dynamic, emergent behavior at the mission level [11,12]. This semantic gap—between qualitative architectural specifications and quantitative algorithmic execution—limits the utility of MBSE in the early design of autonomous SoSs [13].

Concurrent advances in multi-agent reinforcement learning (MARL) offer a complementary perspective. Algorithms such as MAPPO [14], QMIX [15], and their variants have achieved remarkable results in cooperative multi-agent tasks, demonstrating the capacity to derive emergent coordination policies from reward signals alone [16,17]. Nevertheless, MARL-based approaches are typically constructed ad hoc, without grounding in a formal system architecture. The lack of architectural traceability means that learned policies cannot be directly interpreted, validated, or reused within a broader systems engineering context [18]. Furthermore, the sample inefficiency of MARL in sparse-reward, high-dimensional combat scenarios remains a persistent bottleneck [19,20].

To bridge these two research streams, we developed an integrated methodology that connects UAF architecture modeling with MARL-based autonomous decision making. Specifically, a Drivers–Challenges–Opportunities–Goals (DCOG) framework is introduced to systematically map top-level strategic intent onto UAF metamodel elements, providing a principled entry point for architecture design. A metamodel-driven pipeline then

transforms UAF behavioral and resource views into a Markov Decision Process (MDP), ensuring semantic alignment between architectural specifications and the learning environment. An Automatic Curriculum Learning Multi-Agent Proximal Policy Optimization (AutoCL-MAPPO) algorithm is subsequently employed to resolve the MDP, progressively increasing task complexity to accelerate policy convergence [21,22]. The resulting decision sequences are fed back into UAF activity simulations to verify consistency with structural constraints, establishing a verifiable digital thread for the SoS.

The main contributions of this work are threefold. (1) A DCOG-to-UAF mapping mechanism is formalized, enabling non-expert engineers to systematically decompose strategic objectives into operational capabilities and resource constraints. (2) A metamodel-driven MDP transformation pipeline is developed, providing a method for converting UAF models into structured reinforcement learning environments. (3) An AutoCL-MAPPO algorithm with multi-discrete action branching is proposed, demonstrating improved sample efficiency and cooperative task performance in a 4-vs.-4 drone combat scenario under communication decay constraints. Together, these contributions offer a feasible reference for enhancing the architectural consistency and verifiability of autonomous SoS design.

The remainder of this paper is organized as follows. Section 2 reviews the related work on SoS architecture frameworks, MARL algorithms, and curriculum learning. Section 3 describes the proposed methodology, including UAF modeling, MDP formulation, and the AutoCL-MAPPO algorithm. Section 4 presents the experimental results and analysis. Section 5 discusses the advantages and disadvantages of the proposed approach. Section 6 concludes this paper and outlines future research directions.

2. Literature Review

2.1. Architecture Frameworks and MBSE for Autonomous Systems

Architecture frameworks have evolved substantially since Zachman's seminal reference model [23]. Defense-oriented frameworks—including DoDAF [24], NAF [25], and, more recently, UAF [8]—provide structured viewpoints that capture operational intent, system structure, and resource allocation. UAF, standardized by the Object Management Group (OMG) in 2017 and updated to version 1.3 in 2025 [9], consolidates prior frameworks through an Architecture Grid that cross-references eleven domains (strategy, operations, resources, personnel, etc.) with twelve model types, yielding over 80 viewpoints. This rich semantic structure enables comprehensive system-of-systems specification while maintaining alignment with ISO/IEC/IEEE 42010 [26].

In the MBSE literature, UAF has been applied to carrier-based warfare analysis [27], disaster relief SoS design [28], and operational SoS modeling [29]. Lopez et al. [30] further extended MBSE toward digital twin development by integrating model-driven and data-driven paradigms, laying the conceptual groundwork for closed-loop design verification. Fang et al. [31] demonstrated an integrated architecture selection and reconfiguration method for SoS resilience using DoDAF models, highlighting the potential of architecture-level formulations for operational decision support. Despite these advances, a consistent gap persists: existing UAF-based approaches do not provide automated pathways for translating static architectural models into executable, dynamic simulation or learning environments.

2.2. Multi-Agent Reinforcement Learning for Cooperative UAV Systems

The rapid development of MARL has opened new possibilities for autonomous coordination in adversarial environments. Centralized training with decentralized execution (CTDE) has become the dominant paradigm, with representative algorithms including

QMIX [15], MADDPG [32], and MAPPO [14]. Yu et al. [14] demonstrated that properly tuned MAPPO matches or exceeds more complex algorithms in cooperative tasks, making it a strong baseline for multi-agent coordination. Extensions incorporating recurrent networks (GRU-MAPPO) [33] and state-space models (Mamba-MAPPO) [34] have further improved performance in partially observable, long-horizon scenarios.

In the domain of UAV combat and weapon–target assignment (WTA), Li et al. [18] formulated the WTA problem as an MDP, providing a rigorous mathematical foundation for sequential decision making. Zhang et al. [35] systematically analyzed the potential of deep reinforcement learning to advance military command and control intelligence. Recent studies have explored heterogeneous UAV teaming [36], UAV swarm target tracking [37], and swarm tactics [38]. However, these studies typically constructed their MDP environments empirically, without grounding in formal architecture specifications. This limits the interpretability and reusability of learned policies within a broader system engineering lifecycle [11,12]. To align static architectures with dynamic decision making, Li et al. proposed an integrated framework that bridges natural language and DoDAF views using retrieval-augmented generation and formal Petri nets. This closed-loop methodology effectively mitigates semantic gaps and decision hallucinations in collaborative mission systems, underscoring the necessity of grounding advanced computational algorithms within formal systems-engineering structures [39]. Cai et al. proposed an evolutionary framework that integrates artificial intelligence with Model-Based Systems Engineering (MBSE) across the aerospace system lifecycle. Their work highlights the potential of compiling formal architectural constraints into state action spaces for reinforcement learning, thereby establishing a theoretical bridge from qualitative design specifications to executable dynamic validation [40].

2.3. Curriculum Learning for Multi-Agent Training

Curriculum learning addresses the sample inefficiency of MARL by structuring the training process from simple to complex tasks. Bengio et al. [41] introduced the foundational concept of scheduling training examples by difficulty. In the multi-agent context, Klink et al. [42] proposed automatic curriculum learning (AutoCL), which dynamically adjusts task parameters based on agent performance without manual intervention. Multi-agent AutoCL (MA-ACL) [43] and PORTAL [44] have demonstrated significant improvements in convergence speed and final policy quality across cooperative tasks. Curriculum scheduling criteria—based on win rate, reward growth, and policy stability—have proven effective in reducing the brittleness of policies trained under fixed task distributions [45]. Integrating AutoCL with architecture-driven MDP formulations represents a promising direction for bridging the gap between high-level system design and robust autonomous behavior.

Despite growing advances in both UAF-based architecture engineering and MARL-driven cooperative decision making, critical gaps remain at their intersection. On the architecture side, UAF modeling studies [27–29] and DoDAF-aligned executable verification methods [31,39] excel at standardizing architectural descriptions and validating predefined behavioral logic through rule-based simulation or formal methods, but they all rely on manually specified deterministic execution logic and provide no end-to-end pipeline to compile formal architectural specifications directly into reinforcement learning environments for emergent strategy optimization. On the MARL side, combat-oriented multi-agent algorithms [18,35] and conceptual MBSE-AI integration frameworks [40] explore dynamic decision making in SoS contexts, yet their MDP formulations are mostly constructed empirically without formal grounding in a standard architecture metamodel, resulting in poor architectural traceability of learned policies and disconnection from standard MBSE design iteration workflows. The core novelty of this work lies in its unique

three-tier integrated framework—DCOG-based strategic intent decomposition, UAF Domain Metamodel (DMM)-driven automated MDP transformation, and AutoCL-MAPPO paired with SysML-based closed-loop consistency validation—that establishes a complete, traceable design–optimization–validation cycle.

3. Materials and Methods

3.1. UAF Architecture Development Process

During combat operations, reconnaissance (S), decision making (D), and effect-related (I) equipment from both sides cooperate internally, while infrastructure and other assets serve as mutual targets (T), forming their respective operational loops. With the closed operational loop as the core concept, this paper constructs a five-stage UAF architecture modeling process.

1. Define mission objectives and core capabilities: The strategic motivation view (St-Mv) defines mission objectives, such as disrupting the enemy's operational loop and shortening its loop length. The strategic structure view (St-Sr) defines core capabilities by traversing equipment entities to extract their attributes; for instance, a decision-making entity S encompasses both reconnaissance and communication capabilities.
2. Operational domain: First, the four types of entities in the operational loop are abstracted into technology-agnostic operational logic nodes. Based on this, seven types of operational activity information flows are extracted and constructed to achieve closure of the operational loop at the logical level. Operational Taxonomy (Op-Tx): Use SysML block definition diagrams (BDDs) to abstract the defined S, D, I, and T entities into standard operational roles. Operational Connectivity (Op-Cn): Define the data interaction boundaries between the four types of operational roles via internal block diagrams (IBDs) or information interaction matrices. Seven operational activities, such as “intelligence flow from S to D” or “collaborative decision flow from D to D,” are explicitly represented as standard information flows in this view. Operational Process (Op-Pr): Use activity diagrams to sequence the seven operational activities according to timing and control logic.
3. Resources domain view: Concrete logical nodes in the operational layer are instantiated as actual equipment entities. The resources taxonomy view (Rs-Tx) instantiates specific weapon systems and establishes inheritance relationships from logical roles to physical equipment. The resources connectivity view (Rs-Cn) describes physical links and communication interfaces between equipment; for example, a collaborative detection activity in the operational layer is concretized as a bidirectional communication interface established between UAV A and UAV B via a tactical data link.
4. Utilize the UAF traceability matrix (Op-Tr/Rs-Tr) to vertically bridge the ‘strategic capability–operational activity–resource equipment’ chain, ensuring that every collaborative capability is supported by corresponding activities and that every activity is executed by specific equipment, thereby guaranteeing the integrity of the architecture model.
5. Define the measures of effectiveness/measures of performance (MoEs/MoPs) for the closed operational loop and its constituent activities within the metrics view (Me-Pm-M), providing parametric data for the subsequent quantitative evaluation of the unmanned swarm system.

3.2. DCOG-to-UAF Strategic Mapping Mechanism

As the conceptual foundation of the UAF metamodel, the strategic domain captures and formalizes top-level strategic intent and organizational motivation. Within the strategic motivation view, the ontological structure comprises key entities including Goals, Ob-

jectives, and Stakeholders, alongside environmental and tactical drivers such as Drivers, Concerns, Issues, and Constraints. These architectural elements are logically coupled through standardized semantic relationships—including “Contributes To”, “Impacts”, “Is Addressed By”, and “Decomposes”—to construct a causal network that traces the path from raw strategic drivers to intended operational outcomes.

However, an example of the semantic gap is detailed as follows: Consider a UAF strategic goal, “Enhance operational survivability.” Without a formal mapping mechanism, an algorithm engineer might interpret this as “minimizing path length” for agility, while a resource engineer might interpret it as “increasing UAV armor”. This divergence leads to design conflicts, such as an overly heavy and slow swarm that fails mission requirements. The architecture, thus, remains a static document rather than an executable guide. Our DCOG-to-UAF methodology bridges this gap by decomposing “Survivability” into a specific “Challenge” (e.g., radar detection) and mapping it to a measurable “Opportunity” (e.g., electronic jamming), ensuring that all stakeholders work toward a unified, quantifiable objective.

The structural mapping and semantic alignment between the DCOG logic domain and the UAF metamodel domain are illustrated in Figure 1. Under this paradigm, (1) Drivers represent internal or external forces influencing strategic direction, mapping directly to UAF Driver and Concern elements. (2) Challenges represent tactical obstacles and capability bottlenecks, precisely codified as UAF Issue or Constraint elements. (3) Opportunities embody the technological pathways, management innovations, or resource configurations derived to bypass or solve these challenges, mapping onto UAF elements that extend from Drivers (Derived from Driver). (4) Goals define the teleological end state of the system architecture, directly matching UAF Goal and Objective elements.

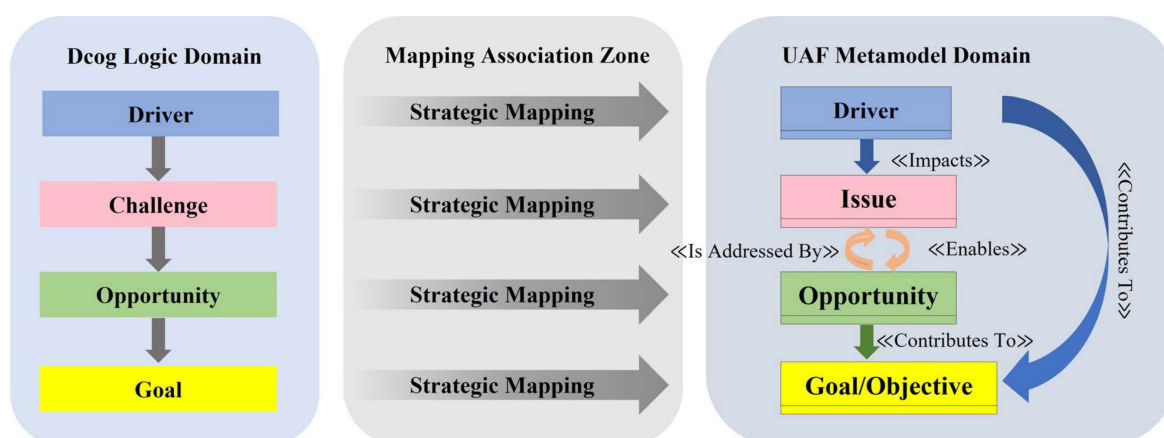


Figure 1. DCOG-UAF mapping logic diagram.

To demonstrate the application of the specific mapping process of the DCOG model within the UAF architecture, a collaborative drone combat scenario is established as a representative backdrop in Figure 2. The UAF models were constructed using M-Design v4400 (Hangzhou Hope System Technology Co., Ltd., Hangzhou, Zhejiang, China). Within this strategic motivation view, (1) Drivers D1 (reducing Blue Team casualties) and D2 (increasing the Blue Team’s probability of completing the strike mission) constitute the top-level drivers, which directly influence two types of core challenges: C1—uncertainty in future combat scenarios (environmental complexity) and C2—Red Force operational threats (external adversarial factors). (2) Based on these challenges, the system transforms them into specific strategic opportunities through a driver-derived mechanism. This case study identifies three types of opportunity spaces: operational characteristics, operational meth-

ods, and exploitation of Red Force vulnerabilities. Operational characteristics encompass core capabilities, such as internal joint operations, precision strikes, and information-centric warfare. Operational methods propose novel operational approaches, such as intelligent coordination, UAV swarm raids, and cross-domain information sharing. Synergistic Effects: Heterogeneous opportunity elements do not exist in isolation but achieve functional complementarity through the “Enables” or “Contributes To” relationships within the UAF (e.g., “information-dominated operations” provide critical resource support for “drone swarm control”), forming a systematic set of solutions. (3) All tactical opportunities converge logically to collectively point toward the logical end state of the system architecture: the operational objective of an intelligent system.

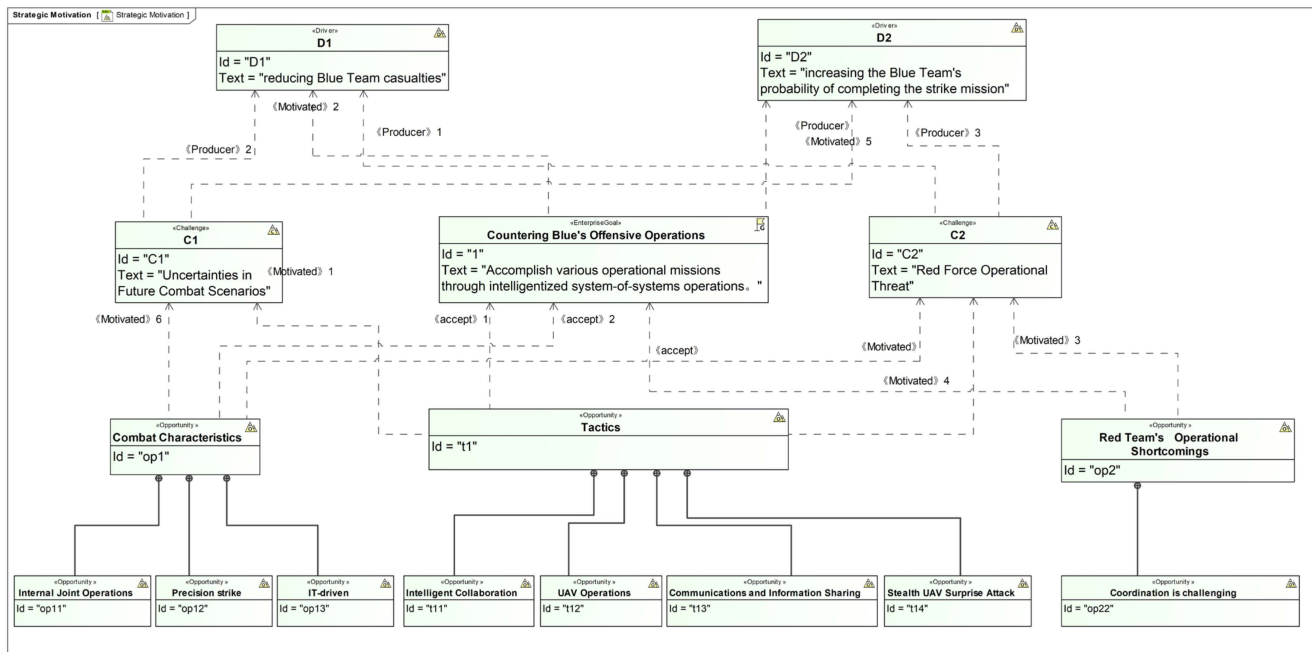


Figure 2. Strategic motivation view.

3.3. UAF Metamodel-Driven Markov Decision Process Modeling

3.3.1. Underlying Principles of Metamodel-Driven Architecture Transformation

DMM laid the foundation for the implementation of UAF and has been widely adopted in many applications. As shown in Figure 3, a mission in the metamodel [46] is represented by a Type, and is directly connected to other elements through three sets of relationships defined by a Tuple. These relationships include trace, refine, and satisfy, which respectively describe the capabilities required to fulfill the mission, the business activities required from a behavioral perspective, and the resource performers from a structural perspective. Capabilities can be mapped by a Process and exhibited by a Performer. Performers are capable of executing (isCapableToPerform) processes. Processes are categorized into OperationalActivity and Function, while performers are categorized into OperationalPerformer and ResourcePerformer.

Analyzing the domain concepts and relationships of an architectural framework based on the DMM ensures the accuracy of the concept and relationship classification. Therefore, we devise a method that uses the semantics of the DMM as a driving source to transform the framework view into a Markov Decision Process.

Evolution of the State Space (S), as shown in Figure 4: Rather than relying on empirical definitions, this is driven by the attribute instances of ResourcePerformer in the resource structure (R_s - S_r) view and the ownedAttribute mappings defined in the metamodel, ensuring the objectivity of the state dimension.

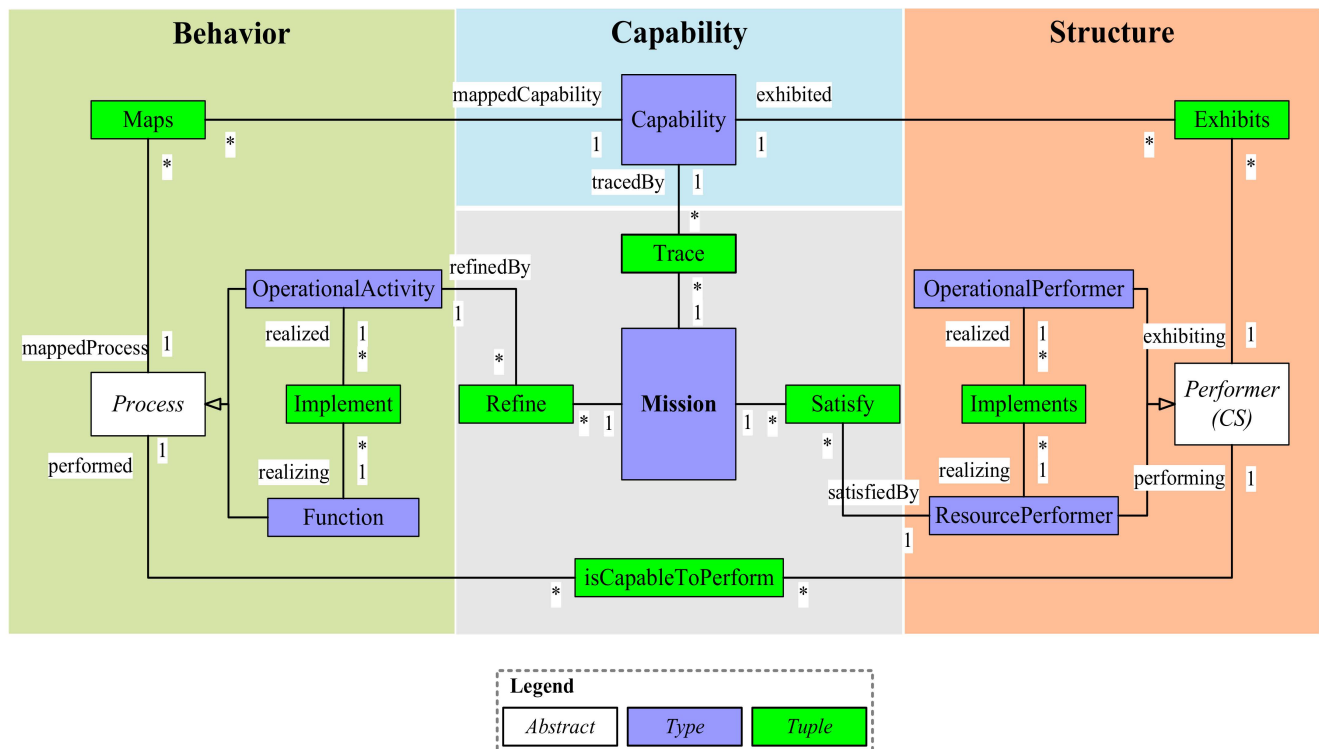


Figure 3. Key concepts of the mission-centric architecture metamodel.

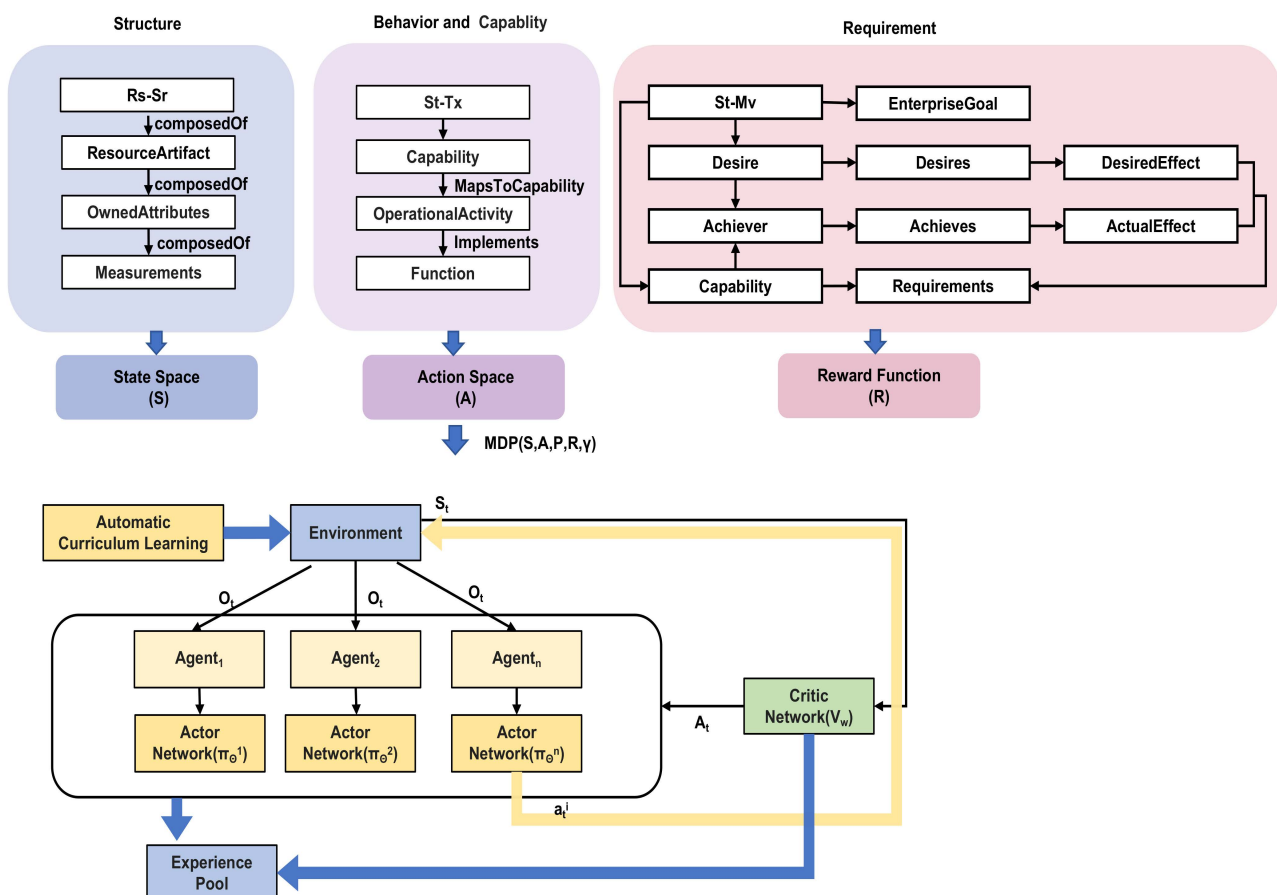


Figure 4. An intelligent decision-making model for MDP based on the UAF architecture.

Evolution of the action space (A): By tracing the isCapableToPerform relationship chain from Capability to OperationalActivity and then to the Function at the Resource level, the system automatically aggregates and generates the agent's functional action set, achieving semantic alignment from capability design to behavioral execution.

Evolution of the reward function (R): The mission objectives in strategic motivation and the DesiredEffect are transformed into the objective function of the MDP. Reward weights are automatically parameterized through the association between ActualMeasurement in the metamodel and requirement priorities.

Based on the Markov Decision Process model described above, this study employs the Multi-Agent Proximal Policy Optimization algorithm to train the Blue Team's agent cluster in collaborative decision making. At each time step t , the environment generates a global state S_t . Each agent $i \in \{1, \dots, n\}$ receives a corresponding local observation O_t^i derived from the environment state. Each agent i maintains an independent policy network Actor, which maps the local observation O_t^i to a specific action a_t^i . The joint action $a_t = (a_t^1, \dots, a_t^n)$ generated by all agents is executed within the environment, which subsequently provides a reward signal r_t and transitions to the next state S_{t+1} . The trajectory data produced during this interaction—comprising local observations, actions, rewards, and global states (O_t^i , a_t^i , r_t , and S_{t+1})—is collectively stored in an experience replay buffer. The centralized critic network V_w takes the global state S_t as input to estimate the value of the current state. In addition, the framework introduces an automatic learning mechanism that dynamically adjusts task difficulty to guide the agent through a structured learning process, progressing from simple tasks to more complex ones.

3.3.2. Specific Programmatic Mechanisms for Software Implementation and the Metadata Extraction Chain

To achieve automated conversion from a static UAF architectural model to an executable MDP simulation environment, this study developed a metadata parsing and code generation toolchain (UAF-to-MDP pipeline). The core of this pipeline is to decouple "model parsing" from "code generation," thereby eliminating the semantic ambiguities associated with manually writing simulation environments. As shown in Algorithm 1, the metadata extraction and mapping mechanism is serialized into the following algorithm.

3.3.3. Formulation of the Combat-Oriented Markov Decision Process Model

The combat scenario is structured as follows: The Blue Team consists of two attack UAVs, one electro-optical reconnaissance UAV, and one electronic warfare aircraft. The objective is to destroy four Red Team fighter aircraft within the detection range. As the attacking force, the Blue Team's core mission is to achieve precision strikes against Red Team targets while preserving its own capabilities to the greatest extent possible.

State Space Definition. To accurately describe the battlefield situation, this study formalizes the attributes of combat units based on the UAF resource structure view and resource taxonomy view, as shown in Figures 5 and 6. By refining and extracting the resource attributes of firepower strike units (fighter jets, UAVs, etc.), a comprehensive state space encompassing information on both Red and Blue forces as well as the environment was constructed.

At decision time t , S_t is determined by the state of the Blue units, the state of the Red units, and the global operational state:

$$S_t = \{S_{t,blue}, S_{t,red}, S_{t,meta}\} \in R^{80} \quad (1)$$

Algorithm 1. UAF-to-MDP transformation algorithm.

Input: UAF export model file UAF_Model.xml and environment template file Env_Template.j2

Output: Executable Python simulation environment code UAVCombatEnv.py

1. Initialize Parsing Engine:

```
DOM = lxml.etree.parse("UAF_Model.xmi")
MetaDict = {"States": [], "Actions": [], "Rewards": []}
```

2. State Space Extraction:

```
for each P in DOM.xpath("//uaf:ResourcePerformer") do
  if P ∈ Core Domains then
    Extract attributes from P (e.g., ID, Name, HP_init, Ammo_max)
    MetaDict["States"].append(Serialize(P.attributes))
  end if
end for
```

3. Action Space Aggregation:

```
for each F in DOM.xpath("//uaf:Function") do
  Trace = F -> OperationalActivity -> Capability
  if F.hasAttribute("isCapableToPerform") then
    Primitives = Extract (Task, Target, Maneuver)
    MetaDict["Actions"][F.owner_id] = Primitives
  end if
end for
```

4. Reward Function Parameterization:

```
for each M in DOM.xpath("//uaf:ActualMeasurement") do
  Terms = ReadStereotype(M, "RewardFunctionTerm")
  Mapping = MatchActivity(M, OperationalActivity)
  MetaDict["Rewards"].append(MapToScalarFormula(Terms, Mapping))
end for
```

Code Generation:

```
EnvTemplate ← Load("Env_Template.j2")
RenderedCode ← EnvTemplate.render(MetaDict)
Save (RenderedCode, "UAVCombatEnv.py")
return UAVCombatEnv.py
```

To clarify the problem scale, the global state vector S_t (80 dimensions) is strictly decomposed as follows: the Blue Team state contributes 36 dimensions, the Red Team state contributes 32 dimensions, and the global battlefield meta-information accounts for the remaining 12 dimensions. For decentralized execution, each agent i receives a 21-dimensional local observation vector O_t^i , focusing on its private status and local situational awareness.

$S_{t,blue}$, $S_{t,red}$ encode the states of Blue and Red units, respectively, each characterized by basic attributes such as unit type, survival status, health, ammunition, and normalized coordinates, with Red units additionally including detected target count and Blue units further described by detection status, jamming intensity, and threat level. $S_{t,meta}$ aggregates global battlefield information including normalized timestep, communication reliability, missiles in flight, cumulative damage, survival ratios of both sides, average Red–Blue distance, threat distribution, total ammunition, total jamming, detection coverage, and battlefield center offset. Communication reliability $C(t)$ exhibits exponential decay as a

function of interaction time t . Setting the initial communication reliability to $C_0 = 1.0$ and the decay factor to $\gamma = 0.995$, the communication reliability at time step t is expressed as $\max(C_{min}, C_0 \cdot \gamma^t)$.

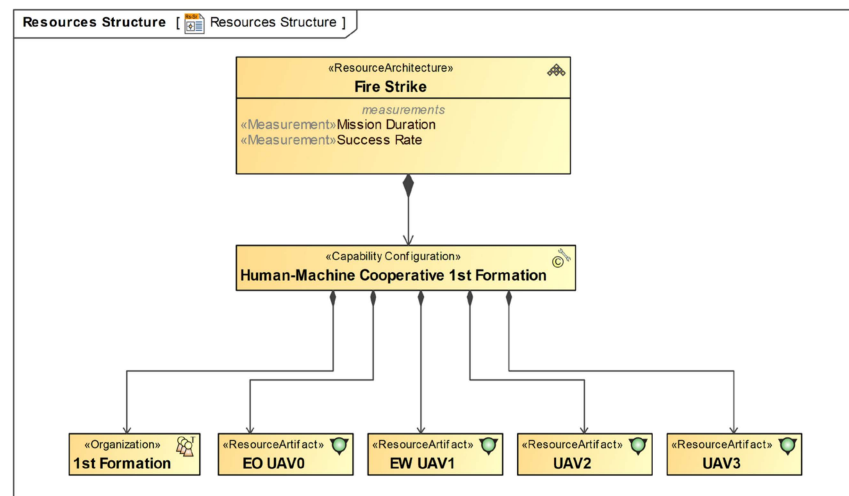


Figure 5. Resource structure view.

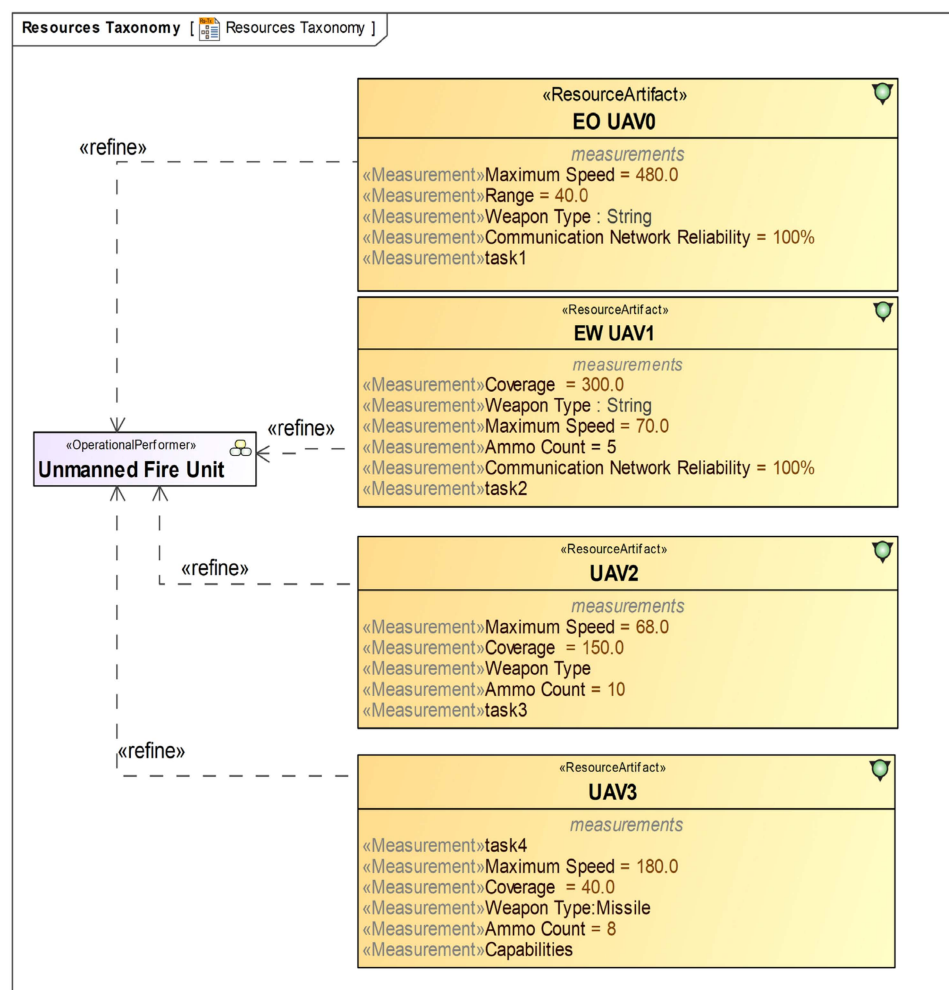


Figure 6. Resource taxonomy view.

Action Space Design. Within the framework of system architecture design, capability elements in the strategic domain serve as the core foundation for analyzing system objec-

tives [7]. As shown in Figure 7, operational missions are concretized through capability inheritance relationships: using low-altitude airspace capabilities as the baseline, core capabilities such as early warning and reconnaissance, firepower strikes, electronic jamming, coordinated strikes, standby, and evasion are further derived based on operational requirements. Furthermore, for specific operational scenarios, key operational factors must be introduced to supplement the framework. For example, the capabilities of a military command center influence the entire operational command process, while information monitoring and sharing capabilities provide critical technical support for the real-time transmission of battlefield situational awareness.

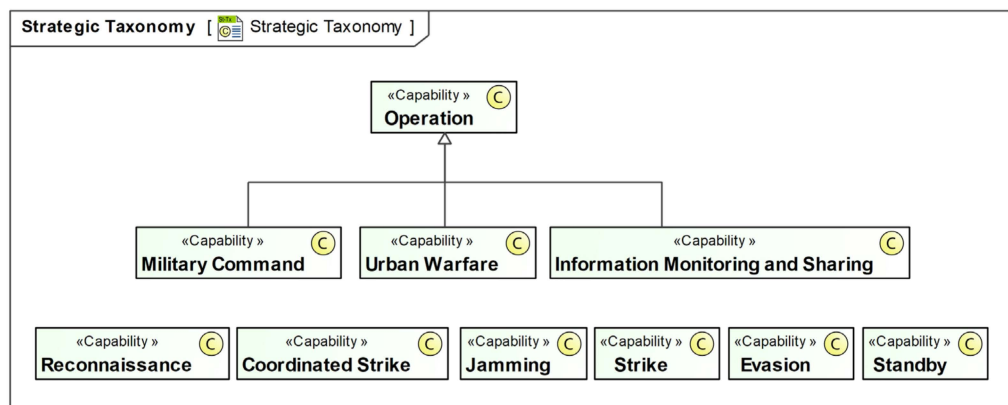


Figure 7. Strategic taxonomy view.

Based on the above definition of strategic capabilities, the agent's action space is designed as a multidimensional discrete space. Specifically, it comprises three dimensions: task type, target strategy, and maneuver direction.

$$a_{i,t} = (T_{i,t}, G_{i,t}, D_{i,t}) \in A, \quad A = T \times G \times D \quad (2)$$

In Equation (2), the multi-discrete branching architecture results in a flattened action space of 216 dimensions ($6 \times 4 \times 9$). $a_{i,t}$ represents the multi-discrete action vector executed by individual agent i at time step t , which is a joint configuration drawn from the comprehensive action space A . The parameter $T_{i,t}$ denotes the specific task type selected by the agent from the discrete task set T , which dictates the current operational mode. The variable $G_{i,t}$ represents the target selection strategy chosen from the strategy set G , guiding how the agent selects its intended adversarial unit. The term $D_{i,t}$ signifies the maneuver direction chosen from the spatial movement set D .

Formally, the task type set T covers six primitives: standby, reconnaissance, jamming, evasion, single-target attack, and swarm attack. The target strategy set G includes four rule-based heuristics: nearest target, weakest defense, highest value, and random selection. The maneuver direction set D comprises nine discrete spatial choices, including eight uniform azimuth vectors and one in-place hold maneuver.

Reward Function Formulation. To guide policy convergence within the high-dimensional adversarial search space and mitigate severe reward sparsity, a composite global reward function $R(S_t, a_t, S_{t+1})$ is formulated. The structural traceability between the reward components and the UAF measurement metrics is enforced via Refine relationships within the architecture model. As illustrated by the mapping matrix in Figure 8, this study establishes the logical traceability between MDP reward terms and UAF architectural elements through Refine relationships. In this matrix, the rows represent Measurements within the architecture, acting as the individual components of the reward function, while the columns correspond to Operational Activities, which constitute the agent's action space

A. At the metamodel level, the stereotype “RewardFunctionTerm” is extended to formally assign specific weight coefficients and attribute values to each measurement. Specifically, each Operational Activity is refined by corresponding Measurements, which parameterize the state transition outcomes into the scalar reward signals R defined in Equations (3)–(8).

		Operational Structure					
		Apply_Electronic_Counte	Conduct_Tactical_Evasio	Engage_Specific_Target	Execute_Coordinated_Swa	Maintain_Standby_Orbit	Perform_Target_Reconnai
Reward Function Components							
BattlefieldAwarenessGain : Real	2	↗					↗
CombatDamageEffectiveness : Real	2		↗	↗			
JointCoordinationEfficiency : Real	1				↗		
MissionAreaProximity : Real	2					↗	↗
OperationalTempoPenalty : Duration	6	↗	↗	↗	↗	↗	↗
ResourceConstraintCompliance : Real	3		↗	↗	↗		

Figure 8. Traceability matrix between UAF operational activities and MDP reward measurements.

As defined in Equation (3), $R(S_t, a_t, S_{t+1})$ represents the overall global reward signal generated during the environment transition from state S_t to S_{t+1} under the joint action vector a_t . The coefficient σ serves as the total reward scaling factor used to normalize the composite signal magnitude for gradient stability. The scalar feedback is linearly deconstructed into six independent sub-reward dimensions: R_{atk} (combat damage bonus), R_{det} (situational awareness gain), R_{guide} (spatial guidance and exploration reward), R_{coord} (coordination gain), R_{cons} (multi-constraint penalty), and R_{time} (time pressure penalty).

$$R(S_t, a_t, S_{t+1}) = \sigma \cdot (R_{atk} + R_{det} + R_{guide} + R_{coord} + R_{cons} + R_{time}) \quad (3)$$

To enhance the transparency of the composite reward function $R(S_t, a_t, S_{t+1})$, we provide a summary of the reward components and their associated design parameters in Table 1. The specific formulations and mathematical components of each sub-reward dimension are elaborated below.

Table 1. Summary of reward function components and associated parameters.

Reward Term	Purpose	Key Parameters
R_{atk}	Quantify offensive damage effectiveness	$V_j, \Delta HP_j, \mu_{env}$
R_{det}	Incentivize situational awareness and tracking	$first_det_j, Jam_j$
R_{guide}	Spatial guidance toward mission areas	R_{move}
R_{coord}	Encourage multi-asset cooperative strikes	$\omega, C(t)$
R_{cons}	Enforce survival and resource constraints	R_{ammo}, R_{logic}
R_{time}	Encourage rapid mission completion	λ_{time}

Combat Damage Bonus (R_{atk}): This metric is designed to quantify the system's core offensive effectiveness by providing feedback based on the real-time physical damage inflicted on enemy units:

$$R_{atk} = \sum_{j \in Red} [1(kill_j) \cdot (\alpha_1 \cdot V_j + \alpha_2) + \beta \cdot \Delta HP_j \cdot V_j \cdot \mu_{env}] \quad (4)$$

In Equation (4), the combat damage bonus R_{atk} aggregates the destruction and damage metrics across all enemy units j belonging to the adversarial Red force set Red. The indicator function $1(kill_j)$ dynamically outputs a value of 1 if enemy unit j transitions to a destroyed status at the current timestep, and 0 otherwise. This event-triggered reward is weighted by the kill coefficient α_1 and scaled by the specific tactical value weight V_j of the targeted unit. The constant parameter α_2 represents a flat team-wipe bonus awarded to the fleet only when the entire enemy formation is eliminated. For continuous damage tracking, β acts as the base damage coefficient that maps the real-time health reduction ΔHP_j of enemy unit j , which is further modulated by its value weight V_j . Finally, μ_{env} serves as a situational bonus multiplier that significantly amplifies the strike reward if the target unit is under active Blue detection or electronic jamming during the engagement.

Situational Awareness Gain (R_{det}): Information superiority is a prerequisite for coordinated operations. This reward term enhances the system's battlefield awareness by incentivizing agents to expand their detection range and ensure continuous target tracking:

$$R_{det} = \sum_{j \in Red} \left[1(first_det_j) \cdot \eta_1 - 1(N_{det} = 0) \cdot \eta_2 + \Delta Jam_j \cdot \eta_3 \right] \quad (5)$$

In Equation (5), the situational awareness reward R_{det} sums the informational gains across the Red force set. The indicator function $1(first_det_j)$ yields 1 upon the very first instance of detecting enemy unit j , granting a one-time exploration bonus governed by the coefficient η_1 . Conversely, the indicator function $1(N_{det} = 0)$ triggers a total blindness penalty factor η_2 if the total number of detected enemy targets across the entire Blue fleet drops to zero, penalizing complete loss of situational awareness. The continuous variable ΔJam_j denotes the real-time increment in electronic warfare jamming intensity successfully applied to target j , scaled by the disturbance increment weight coefficient η_3 .

Guidance and Exploration Reward (R_{guide}): Construct a grid-based spatial potential to guide Blue units to maneuver toward key mission areas. The specific expression is

$$R_{guide} = 1(new_grid) \cdot \epsilon_1 + R_{move} \quad (6)$$

As formulated in Equation (6), the exploration reward relies on the indicator function $1(new_grid)$, which takes a value of 1 whenever any Blue agent traverses into a spatial grid sector that has not been previously uncovered, thereby providing a discovery bonus ϵ_1 . The continuous term R_{move} represents the target proximity guidance reward, which is formally expanded in Equation (7):

$$R_{move} = \sum_{t \in Blue} (1(d > d_{atk}) \cdot \Delta d \cdot \epsilon_2 + 1(d \leq d_{atk}) \cdot (\Delta d \cdot \epsilon_3 + \epsilon_4)) \quad (7)$$

In Equation (7), the proximity reward is accumulated across all active agents within the Blue force set Blue. The variable d measures the instantaneous geometric line-of-sight distance between the combat unit and its selected target, while d_{atk} represents the pre-defined weapon engagement or attack range threshold. The indicator function $1(d > d_{atk})$ activates when the unit is outside the effective strike zone, multiplying the positive distance reduction Δd by the external movement guidance weight ϵ_2 to pull the agent closer.

Conversely, the indicator function $1(d \leq d_{atk})$ triggers once the agent successfully maneuvers within the attack range, scaling the distance adjustment by the internal guidance coefficient ε_3 and appending a constant proximity holding bonus ε_4 to encourage steady engagement positioning.

Coordination Gain R_{coord} : To reflect the synergistic effects of systems engineering, when N missiles strike the same target within a predetermined time window, the system gains an additional coordination gain of $C(t) \cdot \omega \cdot N$ to encourage multi-capability coordinated operations. Here, N represents the exact count of cooperative assets or missiles engaging the identical target within the designated temporal window, while ω denotes the collaborative attack baseline weight.

Multi-constraint penalty term (R_{cons}): This term penalizes cell loss, ammunition waste, and logical violations:

$$R_{cons} = \sum_{i \in Blue} \left[-1(death_i) \cdot \lambda_{type,i} + R_{ammo} + R_{logic} \right] \quad (8)$$

In Equation (8), the operational constraint penalty R_{cons} combines discrete survivability losses with continuous efficiency metrics across all Blue agents i . The indicator function $1(death_i)$ monitors agent survival, outputting 1 if agent i is destroyed, which incurs a negative penalty scaled by the survival cost coefficient $\lambda_{type,i}$ corresponding to its specific platform type. The term R_{ammo} represents the ammunition efficiency bonus function, which mathematically evaluates the system's payload utilization by rewarding successful hits and penalizing wasted or missed rounds. The terminal component R_{logic} imposes severe scalar penalties for behavioral anomalies or logical violations, such as firing without target locks or remaining inert in high-threat sectors.

Time Pressure Penalty R_{time} : To encourage rapid mission completion, a penalty of $-\lambda_{time}$ is applied at each active simulation step, where $\lambda_{time} = 0.5$.

The explanations of the mathematical notations in this section are listed in Table 2.

Table 2. Explanations of mathematical notations.

Notation	Description
σ	Total reward scaling factor
α_1/α_2	Kill weight/team wipe bonus
β	Base damage coefficient
η_1/η_2	First-time detection bonus/total blindness penalty factor
η_3	Disturbance increment weight
$\varepsilon_2/\varepsilon_3$	Movement guidance outside/inside the attack range
λ_{type}	The cost of survival
ω	Collaborative attack on the basis weight
V_j	The value weight of enemy unit j
μ_{env}	Situational bonus: when the target is under detection or jamming, this coefficient significantly increases the strike incentive
Δd	Distance reduction
d_{atk}	Attack range threshold
R_{ammo}	Ammunition efficiency bonus, which includes a hit bonus, and a miss penalty
R_{logic}	Imposes severe penalties for irrational inaction or blind attacks

3.4. Improved MAPPO Algorithm for Collaborative Decision Making

This study constructs a typical Red-versus-Blue adversarial scenario in low-altitude airspace. The Blue Team consists of units with different operational roles, including reconnaissance, electronic jamming, and attack units, with the aim of achieving coordinated operations through multi-agent reinforcement learning. The Red Team employs pre-defined rules-based logic to conduct autonomous defense and counterattacks. This serves as the baseline adversarial environment for the evolution of Blue Team strategies.

To address the complexity and high-dimensional nature of the action space in Red-versus-Blue adversarial scenarios, we design a multi-discrete action branching mechanism for MAPPO to efficiently explore large-scale state-action spaces. The agent's Actor network adopts a decoupled architecture. By sharing a common feature encoder at the base layer, the network first extracts general semantic information from 21-dimensional local observation features and then outputs action distributions for task type (Task), target policy (Target), and maneuver direction (Maneuver) through three independent branch heads. This design decomposes the single discrete space—which originally had up to 216 combinations—into independent dimensions of [4,6,9], thereby reducing the complexity of policy exploration. To enhance training stability, the Critic network receives 80-dimensional global state information, utilizing a global perspective to assist in the accurate estimation of the value function $V(s)$. Additionally, the RunningMeanStd module is introduced to dynamically normalize the state inputs, and gradient clipping is applied to prevent gradient explosion, ensuring smooth policy updates in complex adversarial environments.

An Automatic Curriculum Learning (AutoCL) scheduler is integrated into the training pipeline to dynamically adjust environmental difficulty by monitoring the learning status of agents. The core logic of this mechanism dictates that a transition to a more challenging task is triggered only when three criteria—performance thresholds, learning convergence, and behavioral stability—are simultaneously satisfied within the current task. The training process is partitioned into three progressive phases based on engagement range and task complexity:

- Phase I (6 km): Serving as the initial stage, focus is placed on training agents in fundamental target detection, individual unit attack, and evasive maneuver logic within a constrained decision space.
- Phase II (8–10 km): This intermediate stage introduces requirements for continuous situational tracking and guides agents in learning temporal coordination among reconnaissance, electronic warfare, and attack units.
- Phase III (15 km): As the target stage, complex battlefield conditions are simulated by increasing the frequency of enemy attacks. Agents are required to optimize full-spectrum engagement strategies within highly uncertain environments.

Training data are recorded in real time via a sliding-window mechanism, and transitions to subsequent phases are determined based on the following three quantitative criteria. (1) Performance Fulfillment: Preset thresholds for win rate, detection rate, kill count, and ammunition efficiency must be achieved. To reflect the physical characteristics of varying difficulties, the detection rate threshold is scaled to decay as the engagement range increases, whereas the win rate and kill count thresholds are correspondingly elevated. (2) Progress Convergence: Learning progress is assessed by calculating the rate of change in win rates and reward growth within the most recent window. When the composite growth rate falls below 0.01, the policy learning for the current difficulty level is deemed to have reached a plateau, where further training yields no significant gains. (3) Behavioral Stability: The standard deviation of the win rate over the last 20 checkpoints is calculated. A standard deviation below 0.02 is required to confirm that policy performance has stabilized, thereby excluding transient performance peaks caused by environmental stochasticity. Training data are recorded in real time via a sliding-window mechanism, and the transition to subsequent phases is governed by the quantitative criteria specified in Table 3. These thresholds were established through a preliminary pilot study, in which the performance plateaus of a standard MAPPO baseline were systematically observed. We established these values as the 'performance gates' to ensure that agents have mastered foundational behaviors—such as unit survival and basic detection—before encountering more demanding adversarial conditions.

Table 3. Quantitative transition thresholds for the curriculum learning phases.

Phase	Distance	Win Rate Threshold	Detection Rate Threshold	Kill Count Threshold
Phase I	6 km	60%	53%	2.5
Phase II	8 km	65%	50%	3.0
Phase III	10 km	70%	45%	3.5

Environmental parameters are automatically adjusted to the next phase once the aforementioned conditions are met and the agent has remained at the current difficulty for a minimum number of checkpoints. The parameters of the Actor and Critic networks are seamlessly transferred to the subsequent phase without re-initialization. This ensures that the low-level control policies acquired in simpler scenarios serve as a prior for more complex strategic learning.

The specific algorithmic steps are shown in Algorithm 2.

Algorithm 2. MAPPO-multi-discrete algorithm flowchart1.

1. Initialize the Actor network $\pi(a|s; \theta)$ (which includes a shared feature layer and an independent policy head) and the Critic network $V(s; \Phi)$
 2. Initialize the experience pool B and the online normalization module RMS, and set the discount factor gamma, the pruning coefficient epsilon, and the GAE parameter lambda
 3. for episode = 1 to N do
 4. reset (); Reset the environment state to S_0 and set the initial distance based on the course progress
 5. for $t = 0$ to T_{max} do

Perform online normalization of the global state S_t using RMS
 Decision-making phase: The Actor network processes the local observations of each agent and concurrently outputs the action distributions for the Blue agents.
 Joint sampling: Assign tasks, target policies, and movement directions to each unit, and synthesize joint actions a_t .
 Execution phase: Execute at in the environment, obtain the reward r_t , the next state S_{t+1} , and the termination flag done
 Data storage: Compute the joint log-probability $\log \pi_{\theta}(a_t | s_t)$ and the state value estimate $V(s_t)$, and store them in the experience pool B .
 Update the environment state: $S_t = S_{t+1}$
 end for
 6. Update the RMS statistics and compute the GAE advantage function at and the target value r_t
 7. Update network settings
 8. Regularly save the model and adjust the course difficulty based on the win rate metric
 9. end for
-

4. Results

4.1. Simulation Setup and Parameters

The experimental environment runs on Ubuntu 20.04 with Python 3.10 and PyTorch 2.5.1. The hardware GPU has a computational capacity of over 30 TFLOPS (single-precision).

In the experimental scenario, the Blue Team deploys four combat units, each possessing basic physical attributes; however, their operational effectiveness varies depending on the weightings assigned to reconnaissance, electronic jamming, and strike capabilities. The Red Team serves as the opposing force, controlled by heuristic rules rather than learned policies. Each Red Team unit possesses a maximum HP of 120 (initialized at 80), eight ammunition rounds, and an attack cooldown of two steps after each strike. The physical parameters of the Blue Team's heterogeneous units are summarized in Table 4, and the environmental parameters for both sides are listed in Table 5.

Table 4. Blue Team unit attributes.

Unit	Type	Max HP	Max Ammo	Ability
B1	EO-UAV	100	8	Reconnaissance
B2	EW-UAV	80	10	Jamming
B3	Attack UAV	180	20	Strike/Attack
B4	Attack UAV	200	20	Strike/Attack

Table 5. Environmental parameters.

Configuration Keys	Default Values	Description
blue_recon_range	5000 m	Detection range for reconnaissance missions
blue_jam_range	6000 m	Electronic warfare jamming radius
blue_attack_range	4000 m	Weapon range
blue_move_speed	120 m/step	Distance per step
red_attack_range	3500 m	Enemy weapon range
red_move_speed	80 m/step	Distance the enemy moves per turn
red_attack_damage	50	Maximum damage per attack
comm_decay	0.995	Communication reliability decreases with each step
comm_min	0.4	Minimum communication reliability

To balance the exploration efficiency and convergence stability of the training, the PPO core training parameters used in this algorithm are shown in Table 6.

Table 6. Training parameters.

Parameters	Set Value
actor_lr	1×10^{-4}
critic_lr	3×10^{-4}
gamma	0.99
gae_lambda	0.95
clip_epsilon	0.1
entropy_coef	0.02
batch_size	4000
hidden_dim	256
max_grad_norm	0.5

4.2. Agent Policy Evolution and Performance

To validate the effectiveness of the proposed model, this study constructed a simulation environment based on Pygame 2.6.1 to conduct experiments, as shown in Figure 9. Figure 9a depicts the initial reconnaissance and jamming phase: the Blue Team uses reconnaissance units to detect the target area and the Red Team's deployment and implements jamming to suppress the enemy's situational awareness. Figure 9b illustrates the subsequent coordinated strike phase: after fully assessing the battlefield situation, Blue Team units share data and launch a coordinated attack against Red Team targets. Regarding training, this study employs an Automatic Curriculum Learning mechanism to dynamically

adjust the complexity of the training environment and the difficulty of tasks based on the agents' learning progress, thereby preventing agents from being unable to learn due to tasks being too difficult.

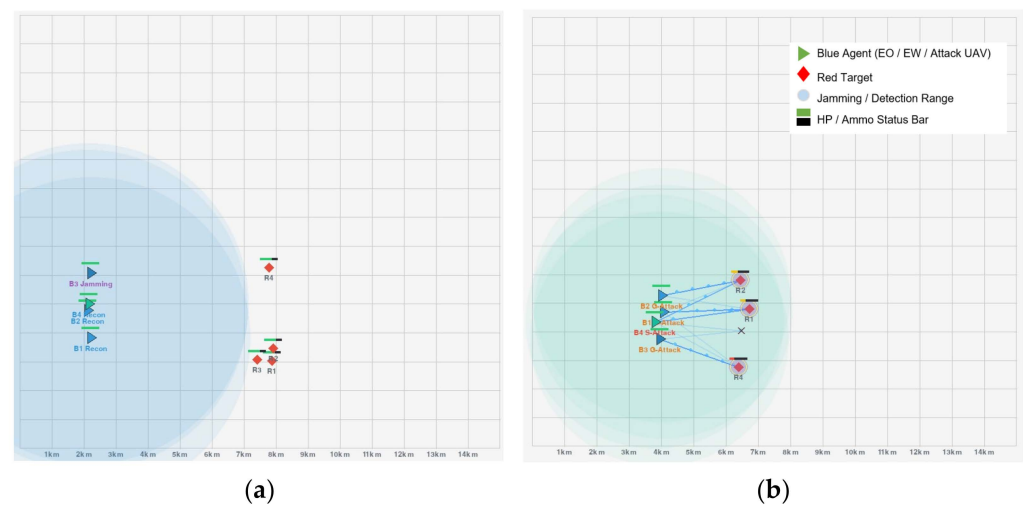


Figure 9. Tactical evolution of the Blue Team agents in the simulation environment. (a) Initial reconnaissance and jamming phase. (b) Subsequent coordinated strike phase.

To rigorously evaluate the performance and systemic advantages of the proposed AutoCL-MAPPO algorithm, we conduct comprehensive baseline comparisons and ablation studies under identical environmental configurations. The baselines include (i) MAPPO, the core multi-agent policy optimization algorithm serving as the ablation benchmark to isolate the impact of the curriculum learning mechanism; (ii) HAPPO, a heterogeneous multi-agent trust region method; and (iii) QMIX, a classic value factorization algorithm used as an external baseline to evaluate the centralized training with decentralized execution (CTDE) paradigm under high-dimensional action branching. All experiments are executed across five random seeds to ensure statistical validity, as summarized in Table 7.

Table 7. Quantitative performance comparison of different algorithms.

Algorithm	Mean Episode Return	Mean Win Rate
AutoCL-MAPPO	388.5 ± 12.4	$73.4 \pm 3.8\%$
MAPPO	325.2 ± 28.6	$65.0 \pm 4.2\%$
HAPPO	371.8 ± 45.3	$50.8 \pm 6.5\%$
QMIX	82.4 ± 15.1	$40.2 \pm 5.9\%$

As illustrated in Figure 10, the reward convergence profiles characterize the exploration efficiency and training stability of the evaluated algorithms across 15×10^3 episodes. Quantitatively, the proposed AutoCL-MAPPO achieves the most optimal asymptotic performance, stabilizing at a superior global return of 388.5, which outclasses the non-curriculum MAPPO baseline (325.2) by a significant margin.

In the initial training horizon (0 to 3×10^3 episodes), AutoCL-MAPPO exhibits an accelerated learning slope, quickly mastering target localization and localized electronic warfare because the curriculum restricts agent interaction to a simplified environment (Phase I, 6 km). Crucially, HAPPO exhibits a delayed convergence signature in the first half of training; due to its sequential agent-by-agent update trust-region mechanism, HAPPO converges slowly and requires approximately 9×10^3 episodes to achieve the baseline return levels of MAPPO. However, its high final return is accompanied by a volatile variance envelope (45.3), implying localized policy brittleness. Meanwhile, QMIX suffers

from a stark representation bottleneck. Following a modest initial ascent, QMIX fails to cross-transfer its value-monotonicity factorization structure effectively as tasks scale up, yielding a severely constrained reward floor of 82.4 in the terminal stage.

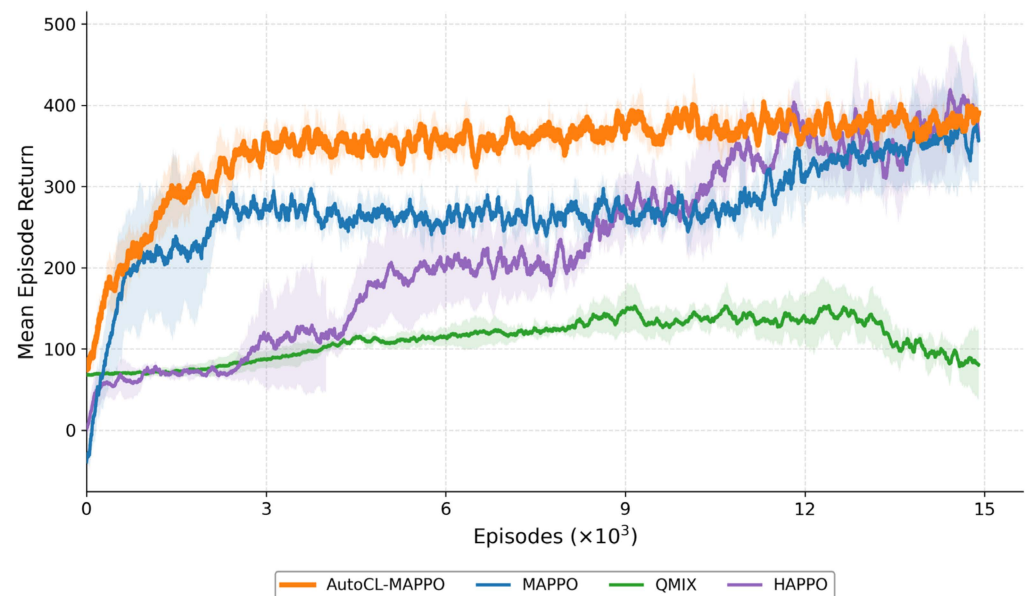


Figure 10. Cumulative reward convergence curves of different MARL algorithms.

Figure 11 illustrates the empirical win rate curves across the training horizon under communication decay constraints, revealing a distinct hierarchical distribution among the evaluated methods (Algorithm 1). The proposed AutoCL-MAPPO achieves a superior state-of-the-art success rate of 73.4%, outperforming the non-curriculum MAPPO (65.0%) baseline and demonstrating a significantly smoother, monotonically ascending trajectory with a tighter terminal variance envelope. By progressively routing the fleet through incremental distance gates, the electronic warfare aircraft (B2) and reconnaissance UAV (B1) master stable orbital synchronization and coordination prior to full-scale engagements. This curriculum scheduling stabilizes policy updates and immunizes the fleet against environmental stochasticity, ensuring deterministic mission completion in Phase III.

Conversely, algorithms trained without a phased curriculum suffer severe performance or stability trade-offs. HAPPO exhibits a highly volatile, bipolar trajectory. It experiences an early exploratory spike above 80% (2×10^3 episodes) due to short-range random actions accidentally bypassing threats before communication decays. However, as the episode horizon extends, the cumulative steps trigger exponential communication degradation. Under this heightened tactical complexity, HAPPO's sequential agent update mechanism fails, causing a catastrophic win rate drop-off to 50.8% and persistent oscillations. Meanwhile, QMIX remains trapped in sub-optimal coordination modes, plateauing at 40.2%. In the simulation, QMIX agents oscillate between erratic blind missile fires and operational inaction, proving that standard monotonic joint value factorization struggles with credit assignment in non-stationary, multi-discrete action spaces ($6 \times 4 \times 9$). Ultimately, these win rate metrics confirm that AutoCL-MAPPO provides the adaptive decision-making logic necessary to validate the dynamic behavioral feasibility of the UAF architecture.

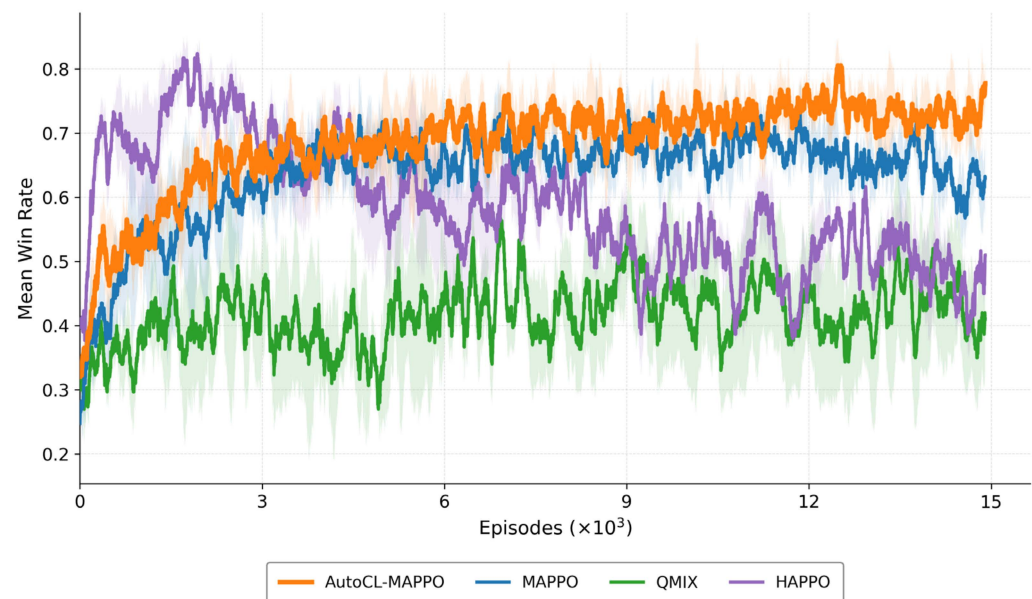


Figure 11. Empirical mission success rate curves of different MARL algorithms.

4.3. Architecture Logic Verification

To close the loop between intelligent decision making and system architecture design, this section presents a method for transforming MAPPO-generated tactical sequences into executable SysML activity diagrams, followed by a lightweight consistency validation to assess architectural feasibility.

4.3.1. Mapping from MAPPO Decisions to SysML Elements

Each MAPPO decision at time step t consists of a tuple (task_type, target_strategy, and maneuver_direction) for each Blue unit, as defined in Equation (2). Table 8 defines the mapping rules that aggregate low-level movement steps rather than representing them individually.

Table 8. Mapping rules from MAPPO decision primitives to SysML elements.

MAPPO Decision Primitives	SysML Element	Mapping Rule
task_type	CallBehaviorAction	Non-standby tasks become Action nodes typed by operational capability
target_strategy	InputPin/ObjectNode	Target selection logic (nearest/weakest/highest) attached as constraint
detected_targets	ObjectNode	Detection state flows as ObjectNode enabling subsequent Attack actions
timestep	ControlFlow/Duration	Consecutive same-task steps merged into single Action with duration = $n_steps \times step_time$

Beyond syntactic mapping, three constraint categories extracted from UAF views are injected: temporal constraints from duration aggregation; resource constraints, where when ammunition is exhausted, an $[ammo > 0]$ guard is inserted and a Retreat branch is generated; and coordination constraints, where multiple units attacking the same target within Δt steps are grouped into a CoordinatedStrike node. Algorithm 3 presents the pseudocode for the interpreter, including the aggregation logic. Figure 12 shows an example Operational Process Flow View generated from an aggregated MAPPO tactical sequence.

Algorithm 3. Pseudocode for Strategy-to-SysML Interpreter

-
1. Input:
Algorithm_Output_Strategy (S)
Output:
SysML_Activity_Script.csv2.
 2. Initialize CSV_Buffer with headers ["Action_ID", "Source_Unit", "Target_Unit", "Action_Type", "Sequence_Node", "Constraints"]
 3. For each tactical_step in S do:
 4. source = extract_blue_unit(tactical_step)
 5. action = map_to_sysml_action_type(tactical_step.action_category)
 6. target = extract_red_unit(tactical_step)
 7. seq_info = determine_control_flow(tactical_step.timestamp, tactical_step.dependency)
 8. duration = calculate_temporal_constraint(tactical_step.n_steps)
 9. model_entry = {ID, Owner: source, Name: action_type, Target: target, Flow: seq_info, Dur: duration}
 10. if validate_consistency(model_entry) return True then:
 11. CSV_Buffer.append(model_entry)
 12. else:
 13. Handle architectural violation
 14. end for
 15. Export CSV_Buffer to SysML_Activity_Script.csv
-

4.3.2. Consistency Validation

To assess whether the generated architecture preserves the learned tactical logic, a lightweight Python constraint checker evaluates each action sequence against UAF-defined rules. The checker iterates through all actions, updates the system state following the original environment dynamics, and evaluates five constraint categories.

As shown in Table 9, the generated intelligent cooperative schemes were mapped back to the UAF metamodels for a multidimensional logic consistency verification. The statistical results indicate that Semantic Allocation (involving St-Tx, Op-Tx, and Rs-Tx views) and Traceability (involving Am-Tr and St-Tr views) achieved a 100% pass rate. This suggests that, during the model transformation process, the Interpreter followed the static logical rules defined by the UAF.

In contrast, the pass rate for Metrics Constraints (primarily associated with Me-Pm-M and Am-Pm views) was 88.2%. Analysis of the simulation data reveals that the remaining 11.8% of violations mostly occurred in high-intensity adversarial scenarios, such as Phase III, at an engagement range of 15 km, where communication decay intensifies significantly. The reconnaissance units (B0/B1) bypassed the reconnaissance command center decision-strike sequential logic pre-defined in the Op-Sq view, opting instead to broadcast target coordinates directly to the strike units (B3/B4).

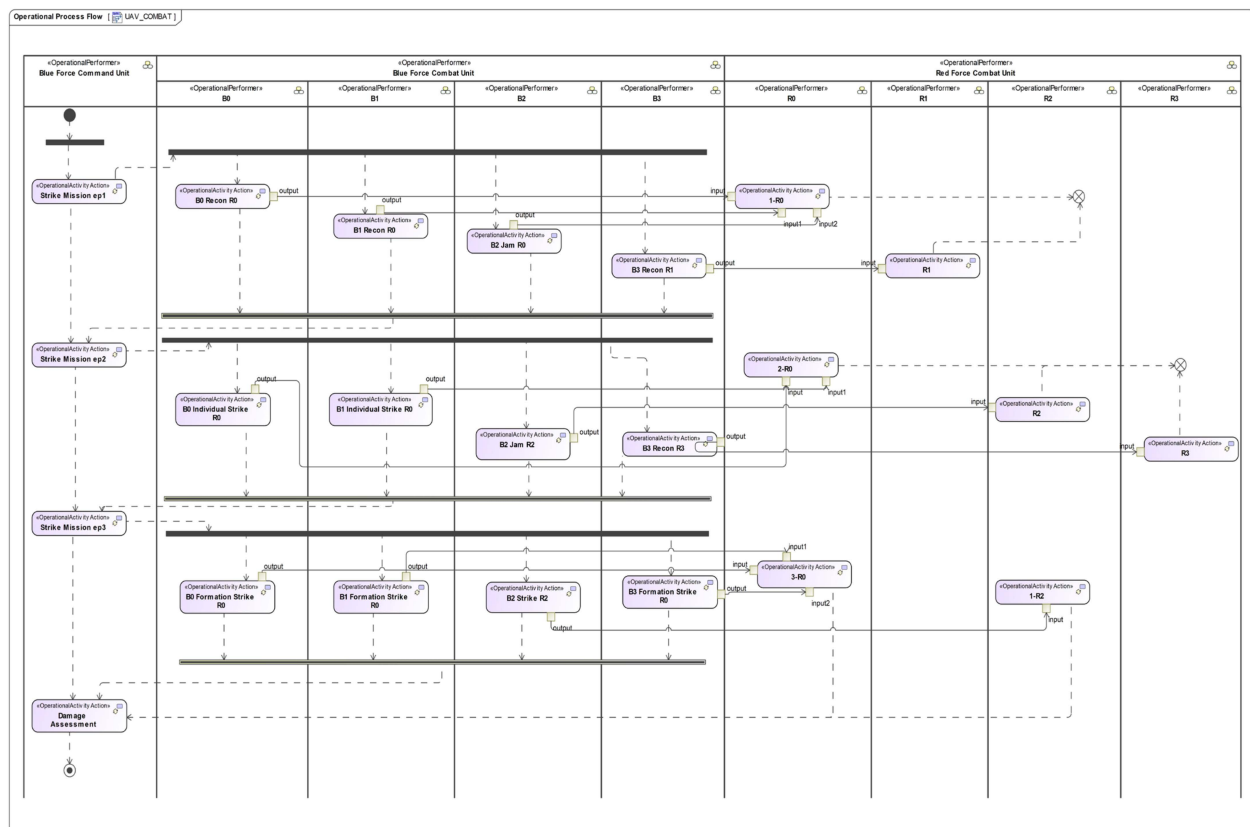


Figure 12. Operational process flow view.

Table 9. Constraint validation results.

Constraint	Source UAF View	Rule	Pass Rate
Semantic Allocation	St-Tx/Op-Tx/Rs-Tx	Check whether the execution unit has a corresponding Capability inheritance or isCapableToPerform association in the resource classification graph.	100%
State Validity	Op-St/Sv-St/Rs-St	The sequence of actions must follow the transition paths defined in the state machine diagram; unauthorized state transitions are strictly prohibited.	95.5%
Interface Matching	Op-Cn/Rs-cn	Verify that the information exchange in the sequence stream conforms to the Resource Exchange types and interface specifications defined in the connectivity diagram.	92.3%
Synchronization	Me-Pm-M/Am-Pm	Check whether real-time consumption data (ammunition, energy, and duration) exceeds the measurement thresholds defined in the metrics view.	88.2%
Traceability	Am-Tr/St-Tr/Op-Tr	Verify whether the Action nodes generated at the lower levels can be traced back through the traceability matrix to the objectives in the Strategic Motivation.	100%

5. Discussion

5.1. Engineering Applicability and Digital Thread

The UAF-driven Markov decision framework developed in this study provides a scalable paradigm for practical systems-of-systems engineering. By bridging the gap between high-level conceptual modeling and low-level tactical execution, this methodology establishes a continuous digital thread.

In practical engineering applications, the configurable design parameters and resource taxonomies exported from the UAF models can be directly mapped to downstream high-

fidelity defense simulation engines, such as the Advanced Framework for Simulation, Integration, and Modeling (AFSIM) or Virtual Battlespace (VBS). The physical constraints and initial states extracted from the UAF can automatically generate standardized configuration files (e.g., XML/JSON formats compliant with AFSIM/VBS specifications) to initialize the simulation scenarios.

Furthermore, the operational behaviors and interaction sequences codified in the UAF views provide the semantic backbone for behavioral scripts. The optimized tactical decision sequences generated by the MAPPO algorithm can be deployed as intelligent Computer-Generated Forces (CGF) controllers. By interfacing with the simulation engine via gRPC or RESTful APIs, the trained agents can replace traditional heuristic behavior trees, driving the simulation with intelligent, adaptive maneuvers. Finally, the high-fidelity quantitative metrics validated by the physics engine (e.g., actual fuel consumption, radar cross-section effects, and precise kill chains) are fed back into the UAF parametric diagrams, fulfilling a comprehensive, closed-loop Model-Based Systems Engineering (MBSE) lifecycle.

5.2. Computational Performance and Parameterization Boundaries

To evaluate the practical implementation readiness of the AutoCL-MAPPO framework, the computational overhead was benchmarked. Utilizing the hardware platform described in Section 4.1 (equipped with a single GPU delivering >30 TFLOPS), the complete training process spanning 15,000 episodes required 11.4 GPU hours. The average wall-clock time per active simulation step was 2.1 ms in Phase I (6 km) and scaled to 3.8 ms in Phase III (15 km) due to the prolonged tracking and adversarial interactions in larger spaces. Compared with the baseline standard MAPPO, the AutoCL curriculum scheduling mechanism introduced a negligible computational time expansion (+4.6%).

Currently, the alignment of the composite reward function relies on direct parameterization linked to the UAF Metric views, which significantly minimizes empirical “trial-and-error” tuning brittleness. To further fortify the deterministic boundaries of this architecture-driven MDP, our immediate next step involves decoupling the interdependent reward terms (such as isolating α , ϵ and R_{logic} coefficients) to execute an exhaustive mathematical sensitivity analysis. Mapping behavioral policy variance under structured reward weight perturbations will provide systems engineers with a clearer understanding of the mathematical trade-offs between offensive aggression and platform survivability boundaries during the conceptual architecture phase.

5.3. Future Evolutionary Trajectories

Despite the promising validation results, certain limitations inherent in the proposed methodology must be acknowledged and systematically addressed in future research.

1. Scalability and the curse of dimensionality: Currently, the feature extraction process is primarily focused on macro-level operational elements, and the model’s performance is constrained by the exponential growth of the joint state-action space. While the 4-vs.-4 adversarial scenarios effectively validate the core coordination logic, scaling the multi-agent system to larger operational tiers (e.g., a 16-vs.-16 swarm confrontation) would significantly exacerbate the difficulty of credit assignment, potentially leading to non-stationarity and training collapse. To overcome this, future work will explore hierarchical multi-agent reinforcement learning (HMARL) or Mean-Field Games (MFGs) to decouple macro-level strategic command from micro-level tactical execution, thereby improving the scalability for massive UAV swarms.
2. Integration of high-level expert logic and rules: The current approach heavily relies on scalar reward shaping to guide policy evolution, which may lead to prolonged exploration phases. High-level logical constraints, such as specific rules of engage-

ment (RoE), airspace restrictions, and expert tactical doctrines, have not yet been fully integrated as hard constraints during the gradient update process. Consequently, future research will focus on the paradigm of neuro-symbolic AI, embedding deterministic operational rules and expert knowledge graphs as constitutive logical constraints into the reinforcement learning pipeline. This trajectory will ensure that the generated tactics are not only mathematically optimal but also strictly compliant with military doctrines, thereby enhancing the operational utility and explainability of the SoS architectures.

3. Bounded scope of UAF domains and viewpoints: The intelligent model transformation framework proposed in this study relies on a highly distilled subset of the UAF architectural grid, selectively prioritizing the Strategic and Operational domains to derive the core MDP state, action, and reward spaces. While this capability-oriented pruning optimizes computational throughput and focus, it represents a partial coverage of the comprehensive UAF ecosystem, which encompasses 11 distinct domains and over 80 standard viewpoints. Neglecting other critical dimensions—such as comprehensive Service, Security, and Project views—restricts the framework’s capacity to evaluate full-lifecycle software services, cybersecurity vulnerabilities, and programmatic evolutionary dependencies within the system-of-systems. Consequently, the current dynamic optimization reflects a bounded behavioral abstraction rather than an exhaustive architectural verification. Future iterations must systematically scale the semantic interpreter to ingest a wider configuration of UAF views to achieve a complete digital thread.

6. Conclusions

This study aims to address the challenges of knowledge fragmentation, loose cross-view semantic correlations, and the lack of quantitative behavioral verification inherent in traditional static MBSE. To bridge the gap between qualitative conceptual architecture and dynamic algorithmic execution, an SoS modeling methodology is proposed, integrating the UAF with MARL. Structurally, a DCOG mapping framework is established within the strategic domain to decompose top-level teleological mandates into operational capabilities. Operationally, a UAF DMM-driven transformation pipeline is introduced to compile static behavioral and resource specifications into an MDP environment, providing a structured approach for model-to-code semantic conversion.

The feasibility of the proposed framework is evaluated within a representative 4-vs.-4 adversarial engagement scenario. Driven by the AutoCL-MAPPO optimization engine, the autonomous Blue fleet demonstrated the ability to execute operational tactics and swarm coordination under communication decay constraints, achieving a mission success rate of approximately 73%. Compared with baseline algorithms such as QMIX and heuristic approaches, the proposed method showed observable improvements in operational tempo and ammunition efficiency within the tested parameters. Furthermore, activity simulations executed via the Simulation Toolkit verified that the generated tactical decision sequences generally conform to UAF structural boundaries and interactive logic.

Importantly, the quantitative feedback loop established through metric constraint deviation analysis offers systems architects empirical data to iteratively refine physical design parameters at the conceptual phase. By linking high-level operational desires to computational models, this methodology outlines a continuous digital thread across the system-of-systems lifecycle. While the current framework represents an initial step toward dynamic MBSE verification and is bounded by specific scenario configurations, this collaborative paradigm contributes a feasible reference for enhancing the operational

utility and architectural consistency of autonomous systems-of-systems. It offers valuable insights for future Human–AI Teaming implementations in complex adversarial domains.

Author Contributions: Conceptualization, Y.F. and P.Z.; methodology, Y.F., Y.L. (Yuting Li) and Y.L. (Yusheng Liu); software, G.Z. and H.L.; validation, Y.L. (Yuting Li) and H.L.; formal analysis, Y.L. (Yuting Li) and Y.F.; investigation, Y.F. and Y.H.; resources, Y.H.; data curation, Y.L. (Yuting Li); writing—original draft preparation, Y.L. (Yuting Li) and Y.F.; writing—review and editing, Y.L. (Yuting Li), Y.F., P.Z. and J.C.; visualization, Y.L. (Yuting Li) and G.Z.; supervision, Y.F., P.Z., Y.L. (Yusheng Liu) and J.C.; project administration, Y.F.; funding acquisition, Y.L. (Yusheng Liu). All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China (grant number 2023YFB3307202); and the Natural Science Foundation of Zhejiang Province (Grant No. LMS26F020022).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to the nature of the simulation-based experiments and the ongoing nature of this research project.

Acknowledgments: The authors thank the Intelligent Systems Engineering Laboratory at Shaanxi University of Science & Technology for technical support and computational resources. During manuscript preparation, GPT-4 was used solely for English language polishing. No AI tools were used for research design, data analysis, or core content writing, and all scientific outcomes were independently verified by the authors.

Conflicts of Interest: Authors Hongyu Li and Yaguang Huang are employed by Hangzhou Hope System Technology Co., Ltd. The remaining authors declare no commercial or financial conflicts of interest. The funders had no role in the design of this study; in the collection, analyses, or interpretation of the data; in the writing of this manuscript; or in the decision to publish the results.

References

1. Khalil, M.; Wang, K.; Choi, J. Performance Evaluation of High Power Microwave Systems Against UAVs A Probabilistic Antenna Propagation Framework with Sensitivity Analysis. *arXiv* **2025**, arXiv:2510.16495. [CrossRef]
2. Maier, M.W. Architecting principles for systems-of-systems. *Syst. Eng.* **1998**, *1*, 267–284. [CrossRef]
3. Boardman, J.; Sauser, B. System of systems—The meaning of “of”. In Proceedings of the IEEE/SMC International Conference on System of Systems Engineering, Los Angeles, CA, USA, 24–26 April 2006; pp. 1–6. [CrossRef]
4. Jamshidi, M. (Ed.) *System of Systems Engineering: Innovations for the 21st Century*; Wiley: Hoboken, NJ, USA, 2009. [CrossRef]
5. Keating, C.; Rogers, R.; Unal, R.; Dryer, D.; Sousa-Poza, A.; Safford, R.; Peterson, W.; Rabadi, G. System of systems engineering. *Eng. Manag. J.* **2003**, *15*, 36–45. [CrossRef]
6. Object Management Group. Unified Architecture Framework (UAF) Version 1.2. Available online: <https://www.omg.org/spec/UAF/1.2/Beta1/About-UAF/> (accessed on 31 May 2026).
7. Madni, A.M.; Sievers, M. Model-based systems engineering: Motivation, current status, and research opportunities. *Syst. Eng.* **2018**, *21*, 172–190. [CrossRef]
8. Object Management Group (OMG). *Unified Architecture Framework (UAF) Domain Metamodel*; OMG: Needham, MA, USA, 2017. Available online: <https://www.omg.org/spec/UAF/1.1/DMM/PDF> (accessed on 31 May 2026).
9. Object Management Group (OMG). Unified Architecture Framework (UAF) 1.3. Available online: <https://www.omg.org/spec/UAF/1.3/> (accessed on 31 May 2026).
10. Hause, M.; Bleakley, G.; Morkevicius, A. Technology Update on the Unified Architecture Framework (UAF). *INSIGHT* **2017**, *20*, 71–78. [CrossRef]
11. Friedenthal, S.; Moore, A.; Steiner, R. *A Practical Guide to SysML: The Systems Modeling Language*; Morgan Kaufmann: Burlington, MA, USA, 2014.
12. Weillkiens, T. *Systems Engineering with SysML/UML: Modeling, Analysis, Design*; Morgan Kaufmann: Burlington, MA, USA, 2011.
13. Estefan, J.A. *Survey of Model-Based Systems Engineering (MBSE) Methodologies*; INCOSE: San Diego, CA, USA, 2008.

14. Yu, C.; Velu, A.; Vinitzky, E.; Gao, J.; Wang, Y.; Bayen, A.; Wu, Y. The surprising effectiveness of PPO in cooperative multi-agent games. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24611–24624. [CrossRef]
15. Rashid, T.; Samvelyan, M.; de Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018*; Dy, J., Krause, A., Eds.; PMLR: Cambridge, MA, USA, 2018; Volume 80, pp. 4295–4304. Available online: <https://proceedings.mlr.press/v80/rashid18a.html> (accessed on 1 June 2026).
16. Huh, D.; Mohapatra, P. Multi-Agent Reinforcement Learning: A Comprehensive Survey. *arXiv* **2023**, arXiv:2312.10256. [CrossRef]
17. Gronauer, S.; Diepold, K. Multi-agent deep reinforcement learning: A survey. *Artif. Intell. Rev.* **2022**, *55*, 895–943. [CrossRef]
18. Li, S.; He, X.; Xu, X.; Zhao, T.; Song, C.; Li, J. Weapon-Target Assignment Strategy in Joint Combat Decision-Making Based on Multi-Head Deep Reinforcement Learning. *IEEE Access* **2023**, *11*, 113740–113751. [CrossRef]
19. Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. A survey and critique of multiagent deep reinforcement learning. *Auton. Agents Multi-Agent Syst.* **2019**, *33*, 750–797. [CrossRef]
20. Papoudakis, G.; Christianos, F.; Schäfer, L.; Albrecht, S.V. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv* **2021**, arXiv:2006.07869. [CrossRef]
21. Narvekar, S.; Peng, B.; Leonetti, M.; Sinapov, J.; Taylor, M.E.; Stone, P. Curriculum learning for reinforcement learning domains: A framework and survey. *J. Mach. Learn. Res.* **2020**, *21*, 1–50. [CrossRef]
22. Portelas, R.; Colas, C.; Weng, L.; Laversanne-Finot, K.; Oudeyer, P.Y. Automatic curriculum learning for deep RL: A short survey. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI), Yokohama, Japan, 7–15 January 2021*; pp. 4819–4826. [CrossRef]
23. Zachman, J.A. A framework for information systems architecture. *IBM Syst. J.* **1987**, *26*, 276–292. [CrossRef]
24. U.S. Department of Defense. DoD Architecture Framework (DoDAF) Ver 2.02. Available online: <https://dodcio.defense.gov/DoDAF/> (accessed on 31 May 2026).
25. North Atlantic Treaty Organization. NATO Architecture Framework Version 4. Available online: <https://www.sto.nato.int/document/nato-architecture-framework-version-4/> (accessed on 31 May 2026).
26. ISO/IEC/IEEE 42010; Systems and Software Engineering—Architecture Description. ISO: Geneva, Switzerland, 2022. Available online: <https://www.iso.org/standard/74393.html> (accessed on 1 June 2026).
27. Cui, R.J.; Sun, J.B.; Yang, K.W.; Li, M.H. Construction method of equipment operational test indicator system based on UAF. *J. Syst. Eng. Electron.* **2025**, *47*, 1536–1550. Available online: <https://www.sys-ele.com/CN/10.12305/j.issn.1001-506X.2025.05.16> (accessed on 29 May 2026).
28. Dridi, C.E.; Benzadri, Z.; Belala, F. A Unified Architecture Framework Supporting SoS's Development: Case of the Aircraft Emergency Response System-of-Systems. *Int. J. Organ. Collect. Intell.* **2023**, *13*, 1–30. [CrossRef]
29. Abhaya, L. UAF (Unified Architecture Framework) Based MBSE (UBM) Method to build a System of Systems Model. *INCOSE Int. Symp.* **2021**, *31*, 227–241. [CrossRef]
30. Lopez, V.; Akundi, A. A Conceptual Model-based Systems Engineering (MBSE) approach to develop Digital Twins. In *Proceedings of the 2022 IEEE International Systems Conference (SysCon), Montreal, QC, Canada, 25–28 April 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 1–5. [CrossRef]
31. Fang, Z.; Li, H.; Chen, D. An integrated method for selecting architecture alternatives and reconfiguration options towards system-of-systems resilience. *Systems* **2025**, *13*, 9. [CrossRef]
32. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6379–6390. [CrossRef]
33. Chen, C.; Guo, Z.; Luo, D.; Xu, Y.; Duan, H. UAV cooperative air combat maneuvering decision-making using GRU-MAPPO. In *Proceedings of the 2024 IEEE 18th International Conference on Control & Automation (ICCA), Reykjavik, Iceland, 23–26 June 2024*; IEEE: Piscataway, NJ, USA, 2024; pp. 1120–1125. [CrossRef]
34. Daniel, J.; de Kock, R.; Ben Nessir, L.; Abramowitz, S.; Mahjoub, O.; Khelifi, W.; Formanek, C.; Pretorius, A. Multi-agent reinforcement learning with selective state-space models. *arXiv* **2024**, arXiv:2410.19382. [CrossRef]
35. Zhang, M.; Dou, Y.; Chen, Z.; Jiang, J.; Yang, K.; Ge, B. Review of deep reinforcement learning and its applications in military field. *Syst. Eng. Electron.* **2024**, *46*, 1297–1308.
36. Liu, D.; Dou, L.; Zhang, R.; Zhang, X.; Zong, Q. Multi-agent reinforcement learning-based coordinated dynamic task allocation for heterogeneous UAVs. *IEEE Trans. Veh. Technol.* **2023**, *72*, 5489–5501. [CrossRef]
37. Xia, Z.; Du, J.; Wang, J.; Jiang, C.; Ren, Y.; Li, G.; Han, Z. Multi-agent reinforcement learning aided intelligent UAV swarm for target tracking. *IEEE Trans. Veh. Technol.* **2022**, *71*, 1057–1070. [CrossRef]
38. Zheng, Z.; Wei, C.; Duan, H. UAV swarm air combat maneuver decision-making method based on multi-agent reinforcement learning and transferring. *Sci. China Inf. Sci.* **2024**, *67*, 180204. [CrossRef]
39. Li, H.; Li, D.; Liu, Y.; Ma, J.; Wang, G.; Ai, J. LLM-Driven Modeling and Decision Support Methods for Cross-Domain Collaborative Mission Systems. *Appl. Syst. Innov.* **2026**, *9*, 80. [CrossRef]

40. Cai, L.; Zheng, H. Exploring the application of AI in aerospace MBSE. In Proceedings of the SPIE 13953, Fourth International Conference on New Materials, Machinery, and Vehicle Engineering (NMMVE 2025), Changsha, China, 11–13 July 2025; Volume 13953. [\[CrossRef\]](#)
41. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In Proceedings of the International Conference on Machine Learning (ICML), Montreal, QC, Canada, 14–18 June 2009; pp. 41–48. [\[CrossRef\]](#)
42. Klink, P.; Abdulsamad, H.; Neumann, G.; Peters, J.; Pajarinen, J. A probabilistic interpretation of self-paced learning with applications to reinforcement learning. *J. Mach. Learn. Res.* **2021**, *22*, 1–52.
43. Jiang, J.; Dun, C.; Huang, T.; Lu, Z. Graph convolutional reinforcement learning. In Proceedings of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 26–30 April 2020. Available online: <https://openreview.net/forum?id=S8icDSeqfvy> (accessed on 1 June 2026).
44. Rutherford, A.; Ellis, B.; Gallici, M.; Cook, J.; Lupu, A.; Ingvarsson, G.; Willi, T.; Khan, A.; de Witt, C.S.; Souly, A.; et al. JAXMARL: Multi-agent RL environments and algorithms in JAX. *arXiv* **2023**, arXiv:2311.10090. [\[CrossRef\]](#)
45. Huang, P.; Xu, M.; Zhu, J.; Shi, L.; Fang, F.; Zhao, D. Curriculum reinforcement learning using optimal transport via gradual domain adaptation. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 10656–10670. Available online: https://proceedings.neurips.cc/paper_files/paper/2022/hash/4556f5398bd2c61bd7500e306b4e560a-Abstract-Conference.html (accessed on 1 June 2026). [\[CrossRef\]](#)
46. Feng, Y.; Zou, Q.; Zhou, C.; Liu, Y.; Peng, Q. Ontology-Based Architecture Process of System-of-Systems: From Capability Development to Operational Modeling. *Appl. Sci.* **2023**, *13*, 5419. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.