2024 IEEE 24th International Conference on Software Quality, Reliability and Security (QRS) | 979-8-3503-6563-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/QRS62785.2024.00043

Application Scenario Modeling and Verification for Unmanned Aerial Vehicle Swarm

Manqing Zhang[1], Renliang Wu[1], Kang Su[1], Yunwei Dong[1,*], and Tao Zhang[2]
[1]School of Software, Northwestern Polytechnical University, Xian, China
[2]School of Computer Science and Engineering, Macau University of Science and Technology, Macao 999078, China
zmqgeek@mail.nwpu.edu.cn, PiedPiper911@163.com, sukang1999@163.com, yunweidong@nwpu.edu.cn
tazhang@must.edu.mo
*corresponding author

Abstract—An unmanned aerial vehicle (UAV) swarm is a cluster system composed of multiple UAVs and is widely used in military and civilian fields. The UAV swarm has a large number of resources, complex functions, space-time constraints, and task-driven characteristics. However, existing UAV swarm task description methods are usually limited to a specific task and cannot adapt to detailed descriptions of dynamic and complex application scenarios. To this end, we propose a UAV swarm application scenario model based on meta-level theory. Specifically, we abstract three types of meta-models from UAV application scenarios: mission meta-model, resource meta-model, and constraint meta-model. Based on this model, we design and implement a UAV swarm application scenario modeling language (ASML) to support the formal description and analysis of the model. Furthermore, we define the conversion rules from ASML to timed automata. We model a logistics handling application scenario and use the model checking tool UPPAAL to verify the correctness of the scenario.

Keywords–Unmanned aerial vehicle swarm; Application scenarios; Meta-level theory; Formal verification

1. Introduction

Unmanned aerial vehicles (UAVs) have been widely used to undertake some boring, repetitive, inconvenient, and dangerous tasks. However, with the increasing diversification of UAV missions, the capabilities of a single UAV make it difficult to meet the needs of complex scenarios. Therefore, it has promoted academic research on Unmanned Aerial Vehicles Swarm (UAV Swarm) [1] technology. The concept of a UAV swarm originates from the swarming behavior of organisms and refers to a swarm system composed of multiple UAVs. UAV swarms have high flexibility and adaptability and are widely used in military and civilian fields. UAV swarms are used in collaborative operations [2], [3] in the military field. In the civilian field, it is widely used in search and rescue [4], [5], logistics and transportation [6], traffic inspection [7], [8], and other scenarios.

Existing research only focuses on specific mission scenarios, such as only applicable to regional monitoring [9] or search and rescue [10] mission scenarios. However, The widespread application of UAV swarms makes them face more complex scenarios, namely task diversification and environmental uncertainty. Existing task modeling methods cannot achieve reuse and accurately describe complex application scenarios when dealing with diverse application scenarios.

We propose a general method for modeling UAV swarm application scenarios to address this limitation. Specifically, we apply meta-level theory to abstract UAV swarm application scenarios into task, resource, and constraint meta-models. UAV swarm application scenarios can be characterized by applying meta-models to establish task, resource, and constraint models. We formalize the metamodel into a modeling language and define its lexical and syntactic rules. In addition, we convert the modeling scenario into a UPPAAL (Developed jointly by Uppsala University and Aalborg University) timed automata to verify the correctness of the scenario. In summary, this paper makes the following contributions:

- We propose a UAV swarm application scenario modeling method based on meta-level theory. This modeling method can accurately describe complex and diverse UAV swarm application scenarios in detail.
- We design and implement a UAV swarm Application Scenario Modeling Language (ASML). We have developed a graphical tool interface that enables low-code and flexible programming.
- We define the conversion rules from ASML to timed automata, which can realize the formal verification of the constructed application scenarios.
- We verify the feasibility of the proposed modeling method and language in a logistics handling scenario.

2. Modeling method

2.1 Overview

In order to abstract and describe the UAV swarm application scenario in detail, we adopt a method rooted in meta-level theory to model the UAV swarm application scenario. Specifically, as shown in Figure 1, we first summarize many UAV swarm application scenario examples and abstract the UAV swarm application scenario into three models: environment, resources, and constraints. By extracting common features between application scenarios, we establish a task meta-model, a resource meta-model, and a constraint meta-model to support the construction of application scenario models. Metamodel [11] can be regarded as the ”model” of the model, which can be used to define the model concept and create corresponding elements for the model. The metamodel is equivalent to the model’s template, and the model can be regarded as an instance of the metamodel. Through the template

2693-9177/24/$31.00 ©2024 IEEE
DOI 10.1109/QRS62785.2024.00043

definitions and concepts provided by the metamodel, models can be abstracted and interactive relationships between models can be described, providing high scalability and flexibility. The application scenario model ultimately consists of a task model, a resource model, and a constraint model.

Moreover, existing Domain-Specific Languages (DSL) have the characteristic of limited expression capabilities [12] and cannot fully cover multi-scenario requirements such as tasks, resources, and environmental constraints. Therefore, we propose and implement an application scenario modeling language ASML for UAV swarms, aiming to provide a formal description method for UAV swarm application scenarios and represent the application scenario model in detail. At the same time, to verify the established scenario's correctness, we convert the scenario model into a timed automaton and use the model detection tool UPPAAL to verify the properties.

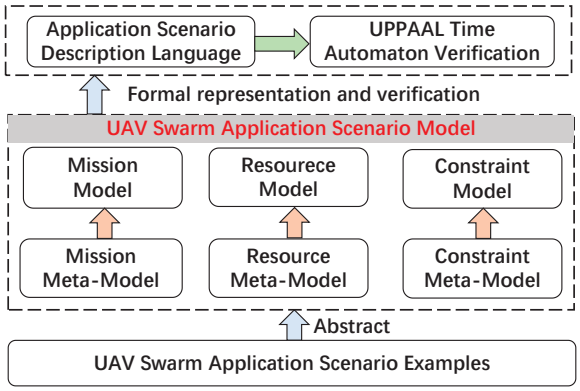

Figure 1: Overview of UAV swarm scenario modeling

As shown in Formula 1, the scenario model is a triplet, including $Mission, Resource, and\ Constraint$. The scenario mission model describes the UAV's target tasks in a specific scenario. The resource model describes the capabilities of the UAV's resources during the execution of its mission. The constraint model models the resource constraints and mission area, weather, obstacles and other environmental factors to enable the drone to better adapt to various environmental conditions.

$$Scenario =< Mission, Resource, Constraint > \quad (1)$$

2.2 Mission Model

In order to clearly and accurately define the mission information in the mission meta-model, we use the 5W1H abstract domain ontology and relationships. Laswell proposed the "5W analysis method" [13], [14] and gradually formed a complete "5W1H" model [15]. 5W1H has been used to model domain ontologies conceptually [16]. Specifically, as shown in Figure 2, we abstract the UAV mission ontology elements from six aspects: purpose (Why), execution body (Who), corresponding location (Where), specific time (When), resource or event (What) and action (How). In addition, we also extract the mission status, mission relationship and terminal flags that relate to each mission.

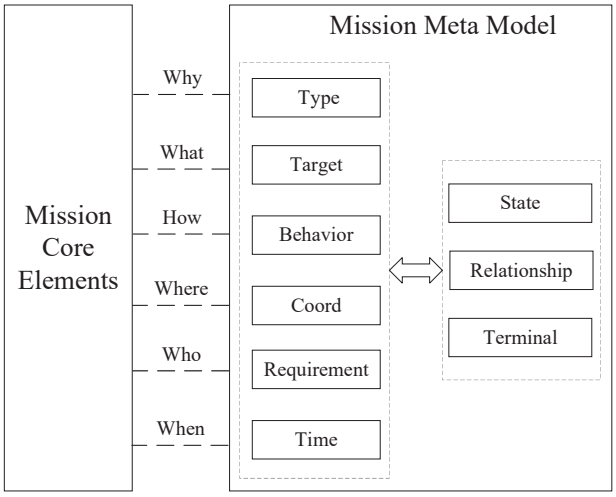

Figure 2: Mission metamodel element extraction model

The mission metamodel abstracts the common characteristics between missions. As shown in Formula 2, the mission metamodel is ultimately composed of 5-tuples, including $id, Name, Description, < Property >, < Include >$. id represents the identifier of the scenario mission, $Name$ represents the name of the scenario mission, and $Description$ represents the description of the scenario mission. $< Property >$ is a set that represents the set of all properties of the scenario mission. As shown in the Formula 3, the property is a 9-tuple that contains the properties corresponding to the mission. $< Include >$ indicates the number of lower-level tasks and their corresponding IDs included in this layer of tasks. For example, we divide scenario tasks into a task structure of up to three levels: Mission, task, and subtask. A mission can include one or more tasks, and a task can also contain one or more subtasks.

$$Mission =< id, Name, Description, < Property >, < Include >> \quad (2)$$

$$Property =< Type, State, Time, Coord, Behavior, Relationship, Requirement, Target, Terminal > \quad (3)$$

Below we discuss in detail the meaning of each element of the task property:

Mission Type: The mission type indicates the level of the task, which corresponds to mission, task or subtask. Mission types can facilitate us to deal with the inclusion relationship between tasks, clarify the corresponding type of tasks, and distinguish whether the tasks are located in the terminal.

State: As shown in Formula 4, State represents the assignment status of the task, and the values are Allocated, UnAllocated and PartiallyAllocated one of the three states. Among them, the status of the terminal task does not include partial allocation. The status of task allocation is bottom-up. The allocation

status of terminal tasks or subtasks affects the allocation of their parent tasks. Assume that the assignment status of the task is State, and its variables can be non-terminal tasks and Terminal tasks.

$$\begin{cases} X \in \{NonTerminal\}, Y \in \{Terminal\} \\ State(X) \in \{Allocated, UnAllocated, \\ PartiallyAllocated\} \\ State(Y) \in \{Allocated, UnAllocated\} \end{cases} \tag{4}$$

Time: Time represents the time information of the mission. Formula 5 defines the constraint relationship of task time. Time sub-elements include $StartTime, EndTime\ and\ EstimatedTime$. The default value of the start time is the current time, and the value needs to be greater than or equal to the current time. The end time T_{end} of the same mission needs to be greater than or equal to the start time T_{start}. The estimated time $T_{estimate}$ needs to be less than or equal to the difference between the end time T_{end} and the start time T_{start}.

$$\begin{cases} Time =< StartTime, EndTime, \\ EstimatedTime > \\ T_{end} \geq T_{start} \geq T_{current} \\ T_{estimate} \leq T_{end} - T_{start} \end{cases} \tag{5}$$

Coord: Coord consists of two triples, including the longitude, latitude and altitude information of the enter and leave coordinate points during task execution. The enter coordinate point is the starting position where the UAV starts to perform the mission, and the UAV enters the designated mission area at this point. The leave coordinate point represents the end position of the UAV leaving the mission area after completing the mission.

Behavior: Behavior represents the mission behavior of the UAV, which is a unique element of the terminal mission. Mission behavior describes the scenario behavior of the UAV under the terminal mission. As shown in Table 1, Behavior defines the specific actions the UAV needs to perform, such as TAKE_PHOTO, RECORD_VIDEO, and TRANSPORT.

Relationship: Relationship represents the temporal relationship between missions. As shown in Figure 3, mission relationships include parallel execution (Cobegin), sequential execution (Sequence), branch execution (Fork) and convergence execution (Join). In particular, when the mission is a terminal task, it is the smallest atomic task, so parallel execution is not supported.

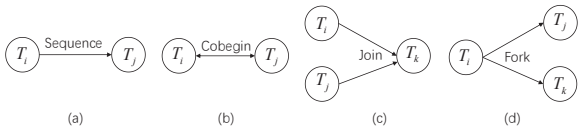

Figure 3: Temporal relationship constraints on mission

$Sequence(T_i, T_j)$ represents the linear execution relationship between tasks, Task T_j can only start execution after task T_i has been executed. $Cobegin(T_i, T_j)$ represents the relationship between task T_i responsible and task T_j starting at the same time. $Join(T_i, T_j, T_k)$ represents that task T_k can only be executed after task T_i and task T_j have been executed. $Fork(T_i, T_j, T_k)$ represents that after the execution of task T_i is completed, task T_j and task T_k will be executed.

Table 1. UAV mission behavior

Keywords	**Parameters**	**Description**
TAKE_PHOTO	shooting location	shooting
FILM_VIDEO	Shooting location and duration	Video
LIGHTING	open sign	illuminate
PATROL	starting point and ending point	Inspection
SEARCH	Task target information	search region
LOAD	Cargo location	Loading and unloading cargo
TRANSPORT	starting point and ending point	transport cargo
DROP	placement	unload location

Requirement: Requirement represents the resources required to complete the task and is used to match the capability scope corresponding to the resource model. For example, performing night missions requires UAVs with resources corresponding to lighting capabilities.

Target: The Target element represents the target characteristics of the task and is a unique element of the terminal task, including TargetLocation, TargetType, TargetShape, TargetEvent, TargetSize and TargetCoords.

Terminal: Terminal indicates the terminal status of the task and determines whether the task is a terminal task. Terminal mission refers to an indivisible mission unit and is also the smallest unit for a drone to perform a mission. Similar to a leaf node in a tree, its out-degree is 0 and it is a terminal node. We use the task meta-model to indicate whether it is a terminal subtask based on the value of 0 or 1 of the task terminal attribute (Termination). If no other tasks will be executed after a task is executed, then the task is called a terminal task. Depending on whether it is a terminal task or not, the elements appearing in the task meta-model are divided into two situations for discussion. Table 2 shows the meaning of the elements of the task metamodel and its relationship with the terminal tasks.

2.3 Resource Model

UAV resources refer to resources such as drones and related equipment, technology and knowledge needed to complete specific missions.

$$Resource =< Info, State, Domain, Service, \\ Log, Performance > \tag{6}$$

Table 2. The meaning of task property elements and their relationship with terminal tasks

Element	Description	Terminal Tasks	Non-terminal Tasks
State	Task State	✓	✓
Type	Task Type	✓	✓
Time	Task Time	✓	✓
Coord	Task Coord	✓	✓
Relationship	Subtask collaboration relationship	×	✓
Requirement	Task resource requirements	✓	✓
Target	Task Target	✓	×
Behavior	Task Behavior	✓	×

Among them, $Info$ represents the basic information of the resource, including four attributes: $id, name, type$ and $description$, which respectively represent the unique identifier, name, type and description of the basic information of the resource. $State$ represents the status information of the resource, which is $idle, inuse, abnormal$ or $invalid$. $Domain$ represents the domain information of the resource, that is, the field in which the resource can be used. The value includes fields such as logistics and transportation, search and rescue, and highway inspection. $Service$ represents the service information of the resource, including the attributes Number and Time and the sub-element TaskList, which respectively represent the quantity of the resource, the available time and the set of tasks that can be completed. Log represents the historical information of the resource, including the attributes Time and Note, which respectively represent a certain historical time of the resource and its corresponding record.

$Performance$ element represents the performance information of the UAV resources, that is, the UAV's resources, payload resources, etc. that can be called when the UAV performs tasks. As shown in the following formula, we manually analyzed the information manual of the UAV official website in detail and extracted the Performance elements into five tuples. UAV performance can be used to describe the capabilities displayed by UAV resources. Therefore, as shown in Table 3, we summarized the five types of capabilities that UAVs need to have based on the performance of UAVs.

$$Performance =< Move, Communication, Sense, Payload, Endurence > \tag{7}$$

2.4 Constraint Model

Various constraints during the mission execution of the UAV swarm are the basis for ensuring the safe execution of the mission. Accurately describing UAV swarm constraints is also a prerequisite for optimal scheduling of UAV swarm operations. In order to accurately describe the constraints of UAV swarm application scenarios, we summarized two types of constraints: UAV resource constraints and environmental constraints.

(a) Resource constraint: Resource constraints limit the resource scheduling during the execution of the UAV swarm, which can make the UAV swarm's task allocation and path planning more accurate and efficient. Resource constraints are deeply bound to the performance of the resource model. The resource model only represents the performance range of the corresponding resource. For example, the theoretical flight time of three UAVs in the endurance capacity is 2 hours, 3 hours, and 5 hours respectively. Assume that a task takes 3.5 hours to execute, and its corresponding sustained performance constraint is 3.5 hours. We match the corresponding capabilities of the drone based on this constraint and select a drone that can fly for 5 hours to perform the task.

$$Resource\_Constraint =< Move_c, Communication_c, Sense_c, Payload_c, Endurence_c > \tag{8}$$

(b) Environment constraint: Environmental constraints can accurately depict the geographical information environment in the scene, ensuring that the drone group can operate safely and efficiently under specific environmental conditions. Formula 9 shows the composition of environmental constraints. Next, we introduce the meaning of each environmental element in detail.

$$Environment\_Constraint =< Obstacle, NoFlyZone, SafeDistance, Weather > \tag{9}$$

Obstacle: Obstacle refers to objects encountered by the drone during flight that cannot be passed horizontally but can be avoided by adjusting the height. As defined by the Formula 10, Obstacle elements include coordinates (coord), minimum height (minAlt), and maximum height (maxAlt). Assume that the coordinates of the UAV during the flight are (U_x, U_y, U_z) and the height range of the obstacles is (Alt_{min}, Alt_{max}). When encountering an obstacle, the UAV needs to lower or raise its height to avoid the obstacle. The z-axis coordinates U_z of the UAV need to be lower than the lowest height of the obstacle $(U_z < Alt_{min})$ or higher than the highest height of the obstacle $(U_z > Alt_{max})$.

$$\begin{cases} Obstacle =< coord, minAlt, maxAlt > \\ U_z < Alt_{min}, U_z > Alt_{max} \end{cases} \tag{10}$$

NoFlyZone: NoFlyZone is the airspace, area or specific location that drones are prohibited from entering or flying over during flight. There are three types of areas in the no-fly zone including rectangular area (Rectangle), circular area (Circle) or polygonal area (Polygon). UAV must strictly comply with the no-fly zone restrictions during flight, that is, they must not be within the longitude, latitude, and altitude of the no-fly zone.

SafeDistance: SafeDistance defines the minimum safe distance and safe height limit between the UAV and the external environment and the UAV. It is described by the three elements minDis, minAlt and maxAlt, which respectively represent the

Table 3. UAV performance keywords and descriptions

Performance Name	Keywords	Description
MoveAbility	maxAscendingSpeed	maximum ascent speed
	maxDescendingSpeed	maximum descend speed
	maxHorizontalSpeed	maximum horizontal flight speed
	maxWindResistance	maximum wind resistance speed
	maxTakeoffAltitude	maximum takeoff altitude
	maxTiltAngle	maximum tilt angle
	maxRotationSpeed	maximum rotation angular speed
	maxHoveringTime	maximum hover time
CommunicationAbility	GNSS	global navigation satellite system
	workingFrequency	working frequency
	imageQuality	image transmission quality
	imageDelay	image transmission delay
	maxsIgnalRange	maximum signal range
SenseAbility	senseabilityType	sensing system type
	obstacleAvoidance	obstacle avoidance
PayloadAbility	emptyWeight	bare metal weight
	maxPayload	maximum payload
	gimbalQuantity	number of UAV gimbals
EnduranceAbility	maxFilghtRange	maximum flight range
	flighttime	theoretical flight time
	currentFlightTime	current flight time
	remainingFlightTime	remain flight time
	workingTemperature	working temperature
	protection	protection level

minimum safe distance and the minimum and maximum flight height in the safety restrictions. The distance between the drone and other objects in the environment is Dis, the current height of the drone is Alt, the minimum safe distance of the drone is $minDis$, and the safe height limits are $minAlt$ and $maxAlt$, then the safety distance constraints $Safe_{distance}$ need to satisfy the following formula:

$$Safe_{distance} = \begin{cases} Dis \leq minDis \\ minAlt \leq Alt \leq maxAlt \end{cases} \tag{11}$$

Weather: Weather will affect the flight direction, track height, smooth hovering, etc. of the UAV. Weather constraints define the weather conditions encountered by the UAV group during the mission, including weather types (Type), such as high temperature, low temperature, heavy fog, thunder and lightning, rainfall, strong wind, etc. Mission planning for different weather constraints during mission execution can improve mission execution efficiency and safety. In bad weather conditions, UAV cannot pass, and the weather coverage area is impassable. The weather constraint tuple is as follows:

$$Weather =< Type, Duration, Temperature, CoveredArea, Wind > \tag{12}$$

Among them, $Duration$ represents the duration range of the weather. $Temperature$ represents the lowest temperature to the highest temperature, which is related to the UAV's working environment. $CoveredArea$ represents the weather coverage area. $Wind$ represents the wind level and wind speed under the current weather, which is related to the wind resistance speed in the UAV resource capability.

2.5 Application Scenario Modeling Language

Specific fields limit traditional Domain-Specific Languages (DSL) and cannot fully cover the needs of multiple UAV swarm scenarios such as missions, resources, and constraints. Therefore, we design an Application Scenario Modeling Language (ASML) to describe UAV swarm application scenarios formally. The structure of ASML follows the meta model of the UAV swarm application scenario. Specifically, ASML converts the mission metamodel, resource metamodel and constraint metamodel into Backus–Naur Form (BNF) to determine the corresponding grammar rules. In addition, in order to ensure the understandability, scalability and reusability of the language, ASML draws on the design method of the extensible markup language XML. XML has strong versatility, scalability and powerful schema support, and can provide basic language design support. Its clear structure and wide adaptability reduce the difficulty of ASML language implementation.

ASML defines basic lexical and grammatical rules. Lexical rules include components, data types, keywords, etc. The components clarify the meaning and composition relationship of each part of the scene modeling language. The data type represents the different types and formats of data and clarifies the storage form of the data. By designing the corresponding

Table 4. ASML to timed automata conversion rules

ASML	Timed Automata	Description
Mission model	Timed automata template	A mission model converted into UPPAAL's timed automata template.
Terminal task	$Location$	Terminal tasks are converted to Location elements in UPPAAL templates.
Task time attribute	$Guard$	If a terminal task contains time attributes, the time attributes are converted into time constraints of the edge $Guard$.
Task Behavior	$Channel$	The task behavior of the terminal task is converted into the Synchronization channel.
Resource constraint	$Update$	Resource constraints are converted into resource matching functions for edge update operations, and resource margins are updated.

vocabulary, ASML keywords obtain the part-of-speech tags and their meanings of application scenarios by designing corresponding vocabulary lists.

Syntax rules use the Backus-Naur Form (BNF) to define syntax rules for scene elements, their tasks, resources, and constraint sub-elements and attributes. BNF provides a more formal grammatical description structure system. As a metalanguage specifically used to define languages, it has the characteristics of concise syntax, and clear expression, and is conducive to syntax analysis and compilation. BNF can express grammar rules canonically, and the grammar it presents does not depend on a specific context. ASML uses XML Schema diagrams to intuitively express each element's attributes and sub-element composition in the scene model. We will not go into more detail about ASML here. We have open sourced [17] the lexical and grammatical rules and usage methods corresponding to ASML for your reference.

In addition, in order to reduce the difficulty of writing the language, we developed a graphical interface tool for the application scenario modeling function. The graphical interface can realize low-code programming and can generate and parse application scenario modeling language according to user configuration. This tool is based on the Qt Creator integrated development environment, uses the C++ programming language combined with the MSVC compiler and provides support for tool interface design through Qt Design.

2.6 Transformation rules for timed automata

The UAV swarm application scenario model can be mapped into a timed automata network. As shown in Table 4, the mission model can be mapped to a timed automata template in UPPAAL, and the terminal task can be defined as a $Location$. The time properties between tasks can be mapped to $Guard$ on the corresponding task edges of the timed automata. Resource constraints can be mapped to the allocation and consumption of resources on the edge of a timed automata.

However, the application scenario model can only be converted into a partially timed automata network. In order to achieve the correct operation of the timed automata, we preset the timed automata template. The timed automata template is preset with some dynamic behaviors of random simulation and some preset required functions. To this end, model conversion becomes a template optimization and completion task. A well-running timed automata network can be obtained as long as the application scenario model completes the corresponding conversion.

Algorithm 1 shows the conversion steps from the UAV swarm application scenario model to the timed automata network. We first convert the preset template to the style of the UPPAAL template. Then traverse all task models, and if the task is terminal, change it to the UPPAAL $Location$. If there are two terminal tasks l_i, l_j, and there is a sequential relationship between them, then the time constraint of terminal task l_i is converted into the $Guard$ of $edge(l_i, l_j)$, and the time constraint of terminal task l_i is converted into the $Update$ function of $edge(l_i, l_j)$. We end up with a converted, runnable timed automata network.

Algorithm 1 algorithm of transformation rules

Input: ASML, Preset Templates
Output: Timed Automata Network

```
begin
UPPAAL Templates ← Preset Templates
for each l_i ∈ task do
    if l_i == terminal then
        Location ← l_i;
        i++;
    end if
    for sequence(l_i, l_j) = True do
        if l_j == terminal then
            Guard(edge(l_i, l_j)) ← time_attribute(l_i);
            Update(edge(l_i, l_j)) ← resource_constrint(l_i);
        end if
        j++;
    end for
end for
return Timed Automata Network
```

3. Empirical results

3.1 Logistics Scenarios

The UAV swarm logistics scenario mainly refers to the realization of logistics transportation through UAV swarms. This logistics scenario is divided into two layers of task structure,

including mission and tasks. The mission is divided into three tasks: load, transport, and unload. These tasks have a sequential relationship and are performed sequentially. For these task types, different UAV mission behaviors are required. As shown in Table 1, executing these tasks requires the appropriate keywords including LOAD, TRANSPORT and UNLOAD as well as the incoming parameters.

3.1.1 Language Comparison

We compare the description capabilities of the ASML UAV scene modeling language and the other two general UAV swarm modeling languages. Table 5 shows that ASML can describe resource capabilities and resource constraints in detail. The other two languages can only describe the task execution of logistics scenarios, but cannot provide a fine-grained description of resource capabilities and resource constraints.

Table 5. Comparison of different modeling languages

Language	Task	Resource	Resource Constraint
GML [18]	✓	×	×
SL4U [19]	✓	×	×
ASML	✓	✓	✓

3.1.2 Environment

Figure 4 shows the setup information of the initial logistics scenario. The specific area size is 20*10, and the gird width as well as minimum safety is 5m. There are a total of 6 way points and they are connected in sequence. At the same time, the no-fly zone is a rectangle from (2,2) to (4,4), in which drones are not allowed to pass through. The constraints for this scenario are shown in Figure 5 according to the ASML. TaskTarget consists of five cargoes with different weights to be transported, each of which is a cubic type. Waypoints are sets of enter and leave points for individual tasks.

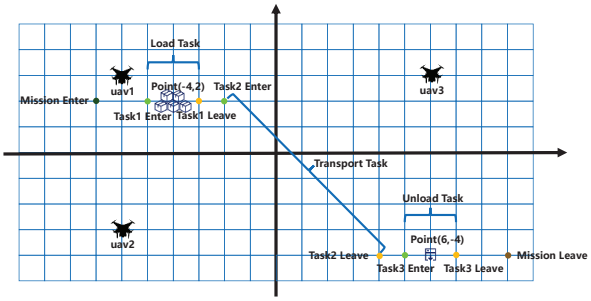

Figure 4: Set up for the initial logistics scenario

3.1.3 Mission

Mission describes the relationship between three tasks with sequences, executing task1, task2 and task3 sequentially, and it contains these three tasks as shown in Figure 6.In this scenario, the tasks of the same type are Cobegin for each UAV, while the tasks of the different types are Sequence for a single UAV. For

```
<Constraint>
    <WayPoints>
        <WayPoint id="WP001" longitude="-5" latitude="2" altitude="10">
            <Connectivity>0,1,0,0,0,0</Connectivity>
        </WayPoint>
        <WayPoint id="WP002" longitude="-3" latitude="2" altitude="10">
            <Connectivity>1,0,1,0,0,0</Connectivity>
        </WayPoint>
        <WayPoint id="WP003" longitude="-2" latitude="2" altitude="10">
            <Connectivity>0,1,0,1,0,0</Connectivity>
        </WayPoint>
        <WayPoint id="WP004" longitude="4" latitude="-4" altitude="10">
            <Connectivity>0,0,1,0,1,0</Connectivity>
        </WayPoint>
        <WayPoint id="WP005" longitude="5" latitude="-4" altitude="10">
            <Connectivity>0,0,0,1,0,1</Connectivity>
        </WayPoint>
        <WayPoint id="WP006" longitude="7" latitude="-4" altitude="10">
            <Connectivity>0,0,0,0,1,0</Connectivity>
        </WayPoint>
    </WayPoints>
    <NoFlyArea>
        <Rectangle id="01 " name="Rect1" minAlt="0" maxAlt="200">
            <Center longitude="3" latitude="3" altitude="100" />
            <NE longitude="4" latitude="3" />
            <SW longitude="2" latitude="2" />
        </Rectangle>
    </NoFlyArea>
    <SafeDistance id="sd1" name="Safe Zone">
        <minDis>5.0</minDis>
        <minAltitude>5.0</minAltitude>
        <maxAltitude>200.0</maxAltitude>
    </SafeDistance>
</Constraint>
```

Figure 5: ASML for constraint

example, each load task is parallel to each other, as is each load task, but there is a sequential relationship between the load and unload tasks. Task1, task2 and task3 describe the tasks to be performed by the UAV, including the task enter point, target point, leave point, and the corresponding behaviors respectively. The components of the logistics scenario mission are shown in Table 6. The relationship between each task is shown in Formula 13, task1 is executed first, followed by task2 and finally task3. The targets of task1 are five cargoes to be transported with masses of 1kg, 1kg, 2kg, 2kg and 3kg. Three UAVs will transport these cargoes from task1's point to task3's point.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Mission id="0" name="mission_0" describe="default">
    <MissionProperty>
        <MissionState>Allocated</MissionState>
        <MissionType>Logistics</MissionType>
        <MissionTime>
            <StartTime>2024-03-19T21:47:26</StartTime>
            <EndTime>2024-03-19T22:47:26</EndTime>
            <EstimatedTime>30min</EstimatedTime>
        </MissionTime>
        <MissionCoord>
            <EnterPoint longitude="-7" latitude="2" altitude="0" />
            <LeavePoint longitude="9" latitude="-4" altitude="0" />
        </MissionCoord>
        <MissionRelationship>
            <Sequence Tasks="task1,task2,task3" />
        </MissionRelationship>
        <MissionRequirements />
    </MissionProperty>
```

Figure 6: ASML for mission

$$\begin{cases} Sequence(Task1, Task2) \\ Sequence(Task2, Task3) \end{cases} \tag{13}$$

Table 6. Composition of logistics scenario tasks

Task attribute	Task1	Task2	Task3
Task State	allocated	allocated	allocated
Task Type	load	transport	unload
Task Time	00:00-00:05	00:05-00:25	00:25-00:30
Task Coord	(-5,2,0)	(1,-1,0)	(8,2,0)
Task Terminal	1	1	1
Task Requirement	Camera	Camera	Camera
Task Behavior	LOAD	TRANSPORT	DROP
Task Target	point	line	point

3.1.4 Resource

For the resource setup, the UAVs are required to have a certain load capacity to lift the cargoes, as well as sufficient battery life, and it need to be loaded with a camera to recognize the appropriate cargoes and drop-off points. Resource information for each UAV is shown in Table 7. The resource information about UAV1, UAV2 and UAV3 are depicted using ASML, where the information for UAV1 is shown in Figure 7. In the end, the three UAVs transported the five cargoes from task1 to task3 and returned to the starting position of UAV1, as shown in Figure 8. The state of each UAV is switched depending on whether it is currently performing a task(e.g., load task) or not for better collaborative tasking.

```
<?xml version="1.0" encoding="UTF-8"?>
<Resource>
  <Info id="1" name="uav1" type="uav" description="uav1 resource" />
  <State>Available</State>
  <Domain>Logistics</Domain>
  <Perfomance>
      <MoveAbility>
          <maxHorizontalSpeed>25.0</maxHorizontalSpeed>
      </MoveAbility>
       <CommunicationAbility>
          <maxSignalRange>10.5km</maxSignalRange>
      </CommunicationAbility>
      <PayloadAbility>
          <maxPayload>1.5kg</maxPayload>
          <missionPayload>Camera</missionPayload>
      </PayloadAbility>
      <EnduranceAbility>
          <maxflightTime>45min</maxflightTime>
      </EnduranceAbility>
  <Service number="1" time="00:00:00-12:00:00" />
 </Resource>
```

Figure 7: Resource information for UAV1

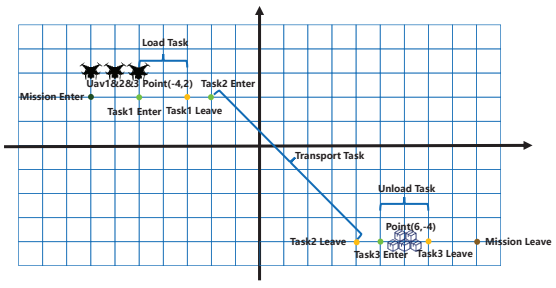

Figure 8: End of the scene for the logistics scenario

Table 7. Composition of initial UAV resources

UAV	Load Ability	State	Location	Endurance	Payload
UAV1	1.5kg	available	(-6,3,0)	45min	camera
UAV2	2.5kg	available	(6,-3,0)	45min	camera
UAV3	3.5kg	available	(6,3,0)	45min	camera

3.2 Experiment platform and result

In order to verify the validity of the scenarios constructed by ASML, this paper configures a UAV simulation environment and conducts the corresponding real-aircraft testing. This experiment's hardware and software setup are as follows:

- CPU: Intel(R) Core(TM) i7-9700
- Memory: 16GB
- Ubuntu: 20.04
- ROS: Noetic
- Gazebo: 11.11.0

The UAV team (UAV1, UAV2 and UAV3) will work together to complete the mission and eventually return to the designated point. As depicted in Figure 9, the initial points of the three UAVs are marked with red circles, and the five cargoes to be transported are in the yellow circles; the UAVs will transport these cargoes to the unloading point in the blue circles.

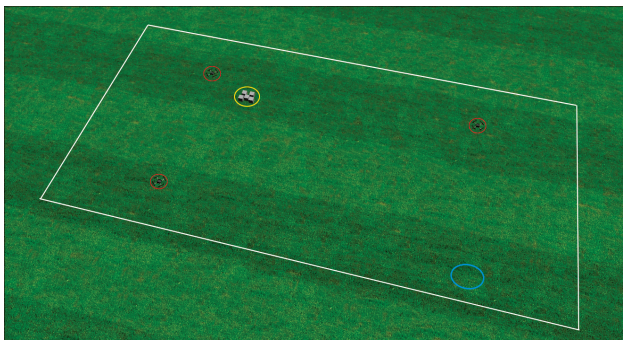

Figure 9: Initial setting of the logistics scenario

The final state of the scenario is shown in Figure 10, where the five cargoes are transported by the three UAVs in concert to the blue circle, while all UAVs return to the starting point of UAV1, marked by the red circle.

3.3 Verification

3.3.1 UPPAAL Models

We convert the logistics UAV swarm application scenario model into a UPPAAL timed automaton network. It includes task templates and task scheduling templates. As shown in Figure 11, the task template abstracts the execution process of each cargo handling task. The logistics scenario contains five cargo handling tasks, and each handling task is an instance of the task template. The cargo task initially is in the $Init$ state, and seeks a UAV assignment. If a drone meets the power and endurance constraints of the handling task, the nearest drone is selected to enter the cargo-carrying state. Otherwise, if no

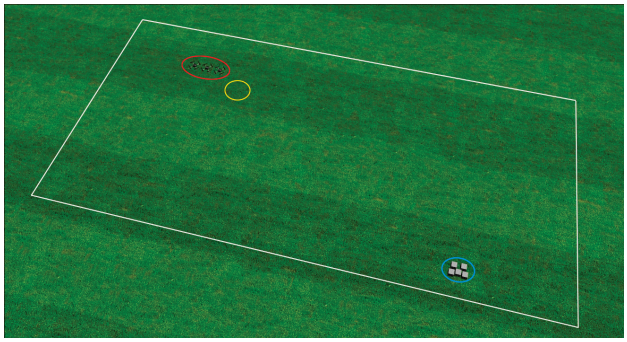

Figure 10: Final state of the logistics scenario

UAV meets the current mission constraints, the UAV needs to return to the base to recharge to maintain UAV swarm resource capabilities. After the drone is handled, transported, and unloaded, it needs to release itself to the unallocated state and send out a leave signal.

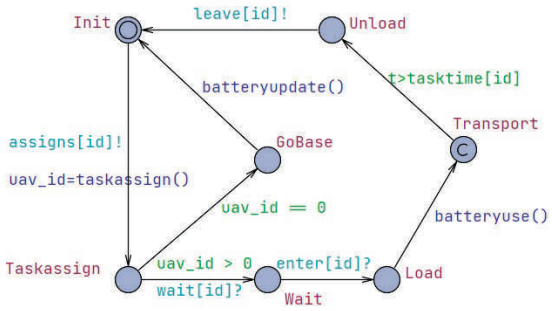

Figure 11: Logistics scenario task automaton

Tasks need to be scheduled correctly to prevent unexpected situations. We referred to UPPAAL's classic example of train scheduling [20] to complete the scheduling template of the drone swarm. Figure 12 shows the timed automata of this scheduling template. Since only one drone is required for each cargo handling, the task scheduling template maintains the sequential execution of tasks through queues to prevent multiple drones from picking up or unloading goods at the same time. When the task is assigned and completed, it is added to the task queue. The task at the end of the queue immediately sends a $wait$ signal from the waiting state, indicating that the drone is executing the task and waiting for completion, and sends an $enter$ task signal to carry the goods. After the UAV task automaton is unloaded, it sends a $leave$ signal and transmits it to the dispatching automaton, indicating that the task is completed and the handling task can continue.

3.3.2 Property verification

We use UPPAAL's model validator to verify the properties of logistics drone swarm temporal automata networks. UPPAAL is one of the most widely used model checking tools for timed automata. It can not only check models, but also

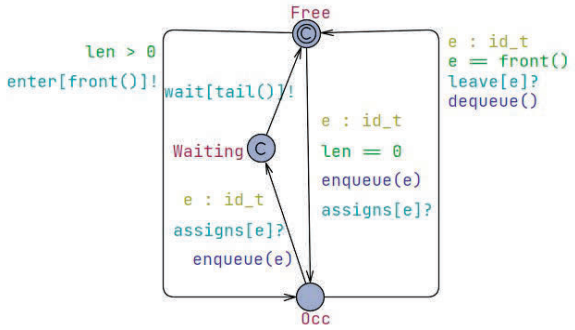

Figure 12: Logistics scenario task scheduling automaton

includes a deadlock detection mechanism. UPPAAL uses Time Computational Tree Logic (TCTL) to define the syntax of the property verification specification language. Verifiers only need to write corresponding query statements to verify the properties of the system. The specific syntax and meaning of the UPPAAL property query language are shown in Table 8.

Table 8. Property description language syntax

Formula	Description
$A[]\ p$	p always holds for all states of all paths.
$A <>\ p$	For all paths, p eventually holds.
$E[]\ p$	There exists a path for which all states p always hold.
$E <>\ p$	There is a path such that p eventually holds.
$p \rightarrow q$	Whenever p holds, q will eventually hold.

Specifically, we verify three major types of properties: deadlock, safety, and liveness. All verification properties are represented using Time Computational Tree Logic (TCTL). We cannot verify the collaborative properties of multiple UAVs, such as two UAVs moving an object simultaneously. Below we describe in detail the properties that require verification.

Deadlock: Deadlock means that a state itself or any of its delayed successors has no outgoing action transition, then the state is a deadlock state. Therefore, Formula 14 verifies whether a certain state will enter a deadlock, causing the system to remain stuck in a certain state.

$$A[]\ not\ deadlock \tag{14}$$

Safety: Safety properties mean that something bad will never happen. In the logistics drone swarm scenario, since only one drone is required for a handling task, two drones cannot load and unload at the same location at the same time to prevent collisions. Formula 15 is used to verify that when the drone executes one task, other drones cannot handle other tasks simultaneously and are in the loading state. The verification of unloading status is similar to this formula.

$$\begin{aligned} A[]\ forall(i : id\_t)\ forall(j : id\_t)\ Task(i).Load \\ \&\&Task(j).Load\ imply\ i == j \end{aligned} \tag{15}$$

Liveness: Liveness properties indicate that something good will eventually happen. Formula 16 indicates that when a path

prevents the UAV from being assigned, the UAV will return to the ground station to recharge to satisfy the UAV capabilities required for the task. Formula 17 indicates that task 0 can eventually be unloaded after being loaded by the UAV. Verify that other handling tasks are similar to task 0, so we will not discuss details here.

$$A <> forall(i : id\_t)\ Task(i).GoBase \tag{16}$$

$$Task(0).Load \rightarrow Task(0).Unload \tag{17}$$

Table 9 shows the verification results of the above properties. We can find that all properties are satisfied after verification. This shows that the UPPAAL model we built meets the quality requirements of not deadlock, safety and liveness.

Table 9. UPPAAL verification results

Property Type	Formula Number	Verification Results
Deadlock	(14)	Satisfy this property
Safety	(15)	Satisfy this property
Liveness	(16)	Satisfy this property
	(17)	Satisfy this property

4. Related work

4.1 Single UAV mission description

A UAV swarm is composed of multiple individual UAVs. Therefore, the individual UAV mission description is the basis of the UAV swarm. To accurately describe the diverse tasks performed by individual robots in different application fields, many scholars use individual task descriptions to describe the tasks and behaviors of robots.

The individual task description is initially based on the waypoint list method [21], which considers the individual task as a waypoint that the robot passes through and the action that should be performed at that point. However, the waypoint list is a fixed form of description and cannot adapt to dynamically changing mission environments when obstacles exist. Konolige et al. [22] proposed a robot control language COLBERT based on a Finite State Machine (FSM) with standard iteration, sequence and conditional structures. Tousignant et al. [23] further proposed the XRobots language based on hierarchical state machines and regarded states as behaviors. The XRobots language solves the limitations of FSM that are not easy to expand and maintain as the number of robot behaviors increases.

In addition, to improve the adaptability of UAVs to dynamic scenes, Molina et al. [10] designed and implemented a task-based mission specification language (TML) for search and rescue scenarios. TML describes tasks hierarchically through a tree structure. Lan et al. [24]use behavior trees as the main underlying structure to represent and further execute complex task plans. Li et al. [25] proposed an integrated real-time sensing UAV mission based on the Robot Operating System (ROS) for flight control systems that use behavior trees as decision-making control mechanisms to manage context changes and flight behaviors.

4.2 UAV swarm mission description

The UAV swarm mission description language can define the tasks that multiple UAVs perform together, the operations each UAV needs to perform, and the mutual restrictions between these operations.UAV swarm mission description methods can be divided into three aspects: quasi-programming language, declarative markup language and graphical interface.

Programming-like language: Programming-like languages are similar to programming languages in syntax and structure but are not primarily used for writing computer programs. These languages can design rules, logic, or procedures that express domain-specific rules, allowing users to programmatically express complex logic or operations. Merino et al. [26] proposed a framework for collaborative fire detection using heterogeneous UAV formations and applied it to the multi-UAV project COMETS. However, drones must be adapted to fit within this framework and function properly. Dantu et al. [27] proposed Karma, a system for programming and managing micro-UAV swarms. Dedousis et al. [28] proposed PaROS (PROgramming Swarm), a framework that can be used for UAV swarm and single UAV programming. PaROS provides developers with Abstract Swarm programming primitives to simplify the programming of drone swarms and eliminate the complexity of low-level programming. Mottola et al. [29] proposed the Team-level programming UAV programming model VOLTRON, which can dynamically allocate tasks to UAVs according to mission requirements to the greatest extent. Pinciroli et al. [30] proposed Buzz, a programming language designed for large-scale, heterogeneous robot clusters. Programming-like languages provide rich feature sets and high expressiveness but are unsuitable for rapid application in dynamic scenarios.

Declarative markup language: In the aviation field, XML has gradually become the data exchange standard in the current European and American air traffic management update program SESAR [31]. Therefore, the task description method of declarative markup language is mainly implemented through the extensible markup language (XML) [32]. Doherty et al. [33] proposed a task specification language based on Task Specification Trees (TST) and applied it to UAV collaborative systems. Bozhinoski et al. [9] proposed domain-specific languages for drone swarms, including Monitoring missions language (MML) and Quadrotor Behavior Language (QBL).

Silva et al. [34] designed a set of languages to describe multi-robot tasks, including the extensible markup-based mission description language Mission Description Language (MDL). The MDL can only describe the area, actions, sequence and time constraints for mission execution, and UAV requirements. In order to describe potential abnormal elements in multi-robot tasks, such as human interference and natural interference, Silva et al. [35] further proposed the Disturbance Description Language Disturbance Description Language (DDL). Silva et al. [36] also proposed Scenario Description Language (SDL) and Team Description Language (TDL) based on extensible markup language as static component expressions of scene and

task knowledge. SDL defines the physical scene and global operational constraints, while TDL defines the vehicle team and team-specific operational constraints. In order to meet the application needs of UAV swarms in multi-task scenarios, Jia et al. [18] constructed a UAV swarm mission model suitable for a variety of dynamic tasks and multiple scenarios, and designed an XML-based cluster task description language Group Mission Language (GML). Zhao et al. [19] proposed the UAV swarm description language SL4U, which divides UAV scenarios into environments and tasks. The declarative markup language approach has good readability, but it is less flexible in application areas and platforms.

Graphical interface: Graphical interface tools can be used to define drone missions and create simple flight plans, such as setting a set of waypoints to fly the drone along a specified route. UAV manufacturers·[37] Parrot and DJI have launched customized graphical interface tools. The customization tools are only open to drones from this manufacturer and are not available for drones from other brands. To formulate more detailed UAV missions, FlyMASTER [38] provides scholars studying UAV swarm systems with a software platform that facilitates integration, rapid development, and flexible use. However, it is only suitable for domain experts with strong technical expertise, and has the limitation of difficulty in interacting with ordinary users. Ruscio et al. [39] proposed MML, a monitoring task language that can be used by non-technical experts, and implemented FLYAQ, a platform for defining monitoring tasks in a graphical interface. However, FLYAQ does not support automatic detection of task regions with multiple geometries or visualization or planning of three-dimensional trajectories [40]. To solve these problems, Besada et al. [40] proposed a mission definition system that simultaneously supports pre-flight mission visualization and trajectory prediction. The graphical interface has a good interactive interface and is easy for users to use, but it is deeply bound to the application it is developed and difficult to migrate.

5. Conclusion

In order to accurately describe UAV swarm application scenarios, we propose a modeling method based on meta-level theory. We characterize UAV application scenarios with mission, resource, and constraint models. Furthermore, to formally represent the proposed modeling approach, we construct and implement an Application Scenario Modeling Language (ASML). ASML refers to the description method of extensible markup language XML and defines the corresponding lexical and BNF syntax rules. ASML can be further converted into timed automata to verify its correctness. We verify the reliability of the proposed modeling method in a logistics handling scenario.

Acknowledgment

This work was supported by the National Key R&D Program of China (Grant No. 2022YFB4501800).

References

[1] Y. Zhou, B. Rao, and W. Wang, "Uav swarm intelligence: Recent advances and future trends," *Ieee Access*, vol. 8, pp. 183 856–183 878, 2020.

[2] X. Wu, Y. Liu, S. Xie, and Y. Guo, "Collaborative defense with multiple usvs and uavs based on swarm intelligence," *Journal of Shanghai Jiaotong University (Science)*, vol. 25, pp. 51–56, 2020.

[3] S. Javaid, N. Saeed, Z. Qadir, H. Fahim, B. He, H. Song, and M. Bilal, "Communication and control in collaborative uavs: Recent advances and future trends," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[4] R. Arnold, J. Jablonski, B. Abruzzo, and E. Mezzacappa, "Heterogeneous uav multi-role swarming behaviors for search and rescue," in *2020 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*. IEEE, 2020, pp. 122–128.

[5] V. Lomonaco, A. Trotta, M. Ziosi, J. D. D. Y. Avila, and N. Díaz-Rodríguez, "Intelligent drone swarm for search and rescue operations at sea," *arXiv preprint arXiv:1811.05291*, 2018.

[6] J. Wen, L. He, and F. Zhu, "Swarm robotics control and communications: Imminent challenges for next generation smart logistics," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 102–107, 2018.

[7] Y. Zhong, S. Ye, Y. Liu, and J. Li, "A route planning method for uav swarm inspection of roads fusing distributed droneport site selection," *Sensors*, vol. 23, no. 20, p. 8479, 2023.

[8] N. A. Khan, N. Jhanjhi, S. N. Brohi, R. S. A. Usmani, and A. Nayyar, "Smart traffic monitoring system using unmanned aerial vehicles (uavs)," *Computer Communications*, vol. 157, pp. 434–443, 2020.

[9] D. Bozhinoski, D. Di Ruscio, I. Malavolta, P. Pelliccione, and M. Tivoli, "Flyaq: Enabling non-expert users to specify and generate missions of autonomous multicopters," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 801–806.

[10] M. Molina, R. A. Suarez-Fernandez, C. Sampedro, J. L. Sanchez-Lopez, and P. Campoy, "Tml: a language to specify aerial robotic missions for the framework aerostack," *International Journal of Intelligent Computing and Cybernetics*, vol. 10, no. 4, pp. 491–512, 2017.

[11] T. Kühne, "Matters of (meta-) modeling," *Software & Systems Modeling*, vol. 5, pp. 369–385, 2006.

[12] M. Fowler, *Domain-specific languages*. Pearson Education, 2010.

[13] P. Wenxiu, "Analysis of new media communication based on lasswell's "5w" model," *Journal of Educational and Social Research*, vol. 5, no. 3, pp. 245–250, 2015.

[14] K. Parton, K. McKeown, R. E. Coyne, M. T. Diab, R. Grishman, D. Hakkani-Tür, M. Harper, H. Ji, W. Y. Ma, A. Meyers *et al.*, "Who, what, when, where, why?

comparing multiple approaches to the cross-lingual 5w task," 2009.

[15] Y. Yu and Y. Bi, "A study on "5w1h" user analysis on interaction design of interface," in *2010 IEEE 11th International Conference on Computer-Aided Industrial Design & Conceptual Design 1*, vol. 1. IEEE, 2010, pp. 329–332.

[16] L. Yang, Z. Hu, J. Long, and T. Guo, "5w1h-based conceptual modeling framework for domain ontology and its application on stpo," in *2011 Seventh International Conference on Semantics, Knowledge and Grids*. IEEE, 2011, pp. 203–206.

[17] Renliang, Wu, "Application scenario modeling language," https://github.com/PiedPiper911/ASML, 2024.

[18] W. Jia, J. Ni, G. Yang, R. Wang, Y. Yao, and W. Wu, "Design and implementation of task description language for uav swarms," in *2022 IEEE International Conference on Unmanned Systems (ICUS)*. IEEE, 2022, pp. 158–164.

[19] Y. Zhao, Y. Yao, T. He, X. Zhou, and B. Shen, "Sl4u: a scenario description language for unmanned swarm," *The Journal of Supercomputing*, vol. 80, no. 4, pp. 5363–5389, 2024.

[20] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on uppaal," *Formal methods for the design of real-time systems*, pp. 200–236, 2004.

[21] B. L. Brumitt and A. Stentz, "Grammps: A generalized mission planner for multiple mobile robots in unstructured environments," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 2. IEEE, 1998, pp. 1564–1571.

[22] K. Konolige, "Colbert: A language for reactive control in sapphira," in *KI-97: Advances in Artificial Intelligence: 21st Annual German Conference on Artificial Intelligence Freiburg, Germany, September 9–12, 1997 Proceedings 21*. Springer, 1997, pp. 31–52.

[23] S. Tousignant, E. Van Wyk, and M. Gini, "Xrobots: A flexible language for programming mobile robots based on hierarchical state machines," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 1773–1778.

[24] M. Lan, Y. Xu, S. Lai, and B. M. Chen, "A modular mission management system for micro aerial vehicles," in *2018 IEEE 14th International Conference on Control and Automation (ICCA)*. IEEE, 2018, pp. 293–299.

[25] G.-Y. Li, R.-T. Soong, J.-S. Liu, and Y.-T. Huang, "Uav system integration of real-time sensing and flight task control for autonomous building inspection task," in *2019 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IEEE, 2019, pp. 1–6.

[26] L. Merino, F. Caballero, J. Martinez-de Dios, and A. Ollero, "Cooperative fire detection using unmanned aerial vehicles," in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 1884–1889.

[27] K. Dantu, B. Kate, J. Waterman, P. Bailis, and M. Welsh, "Programming micro-aerial vehicle swarms with karma," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, 2011, pp. 121–134.

[28] D. Dedousis and V. Kalogeraki, "A framework for programming a swarm of uavs," in *Proceedings of the 11th Pervasive Technologies Related to Assistive Environments Conference*, 2018, pp. 5–12.

[29] L. Mottola, M. Moretta, K. Whitehouse, and C. Ghezzi, "Team-level programming of drone sensor networks," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, 2014, pp. 177–190.

[30] C. Pinciroli and G. Beltrame, "Buzz: An extensible programming language for heterogeneous swarm robotics," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3794–3800.

[31] S. J. Undertaking *et al.*, "Sesar joint undertaking: single programming document 2020-2022." 2019.

[32] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (xml)," *World Wide Web Journal*, vol. 2, no. 4, pp. 27–66, 1997.

[33] P. Doherty, F. Heintz, and D. Landén, "A distributed task specification language for mixed-initiative delegation," in *International conference on principles and practice of multi-agent systems*. Springer, 2010, pp. 42–57.

[34] D. C. Silva, P. H. Abreu, L. P. Reis, and E. Oliveira, "Development of a flexible language for mission description for multi-robot missions," *Information Sciences*, vol. 288, pp. 27–44, 2014.

[35] L. P. R. D C Silva, P H Abreu and E. Oliveira, "Development of a flexible language for disturbance description for multi-robot missions," *Journal of Simulation*, vol. 10, no. 3, pp. 166–181, 2016.

[36] D. Castro Silva, P. Henriques Abreu, L. P. Reis, and E. Oliveira, "Development of flexible languages for scenario and team description in multirobot missions," *AI EDAM*, vol. 31, no. 1, pp. 69–86, 2017.

[37] J. Glossner, S. Murphy, and D. Iancu, "An overview of the drone open-source ecosystem," *arXiv preprint arXiv:2110.02260*, 2021.

[38] A. P. Lamping, J. N. Ouwerkerk, and K. Cohen, "Multi-uav control and supervision with ros," in *2018 aviation technology, integration, and operations conference*, 2018, p. 4245.

[39] D. D. Ruscio, I. Malavolta, P. Pelliccione, and M. Tivoli, "Automatic generation of detailed flight plans from high-level mission descriptions," in *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, 2016, pp. 45–55.

[40] J. A. Besada, L. Bergesio, I. Campaña, D. Vaquero-Melchor, J. López-Araquistain, A. M. Bernardos, and J. R. Casar, "Drone mission definition and implementation for automated infrastructure inspection using airborne sensors," *Sensors*, vol. 18, no. 4, p. 1170, 2018.