

Onboard Mission Management for a Decentralized Multi-UAV System using Behavior Trees

Karolin Thomessen[1], Simon Schopferer[1], Pranav Nagarajan[1]

[1]German Aerospace Center (DLR)

Abstract

This paper presents a modular Behavior Tree-based framework for decentralized task assignment in a multi-UAV system as part of the onboard mission management. This framework is motivated by the central challenge of robust and efficient coordination needed for autonomous multi-UAV missions. Real-world deployment of a multi-UAV system involves operation in dynamic environments in which tasks and mission objectives can change quickly, requiring UAVs to react and adapt their current mission. Decentralized coordination is a key technology to enable individual UAVs to make mission-related decisions based on their local knowledge and shared information from other UAVs, even in situations with limited communication. The decentralized approach allows for robust, scalable and more reactive coordination of the UAVs. However, this also requires integration of planning and decision-making algorithms into the autonomous onboard mission management. In this work, we present an onboard mission management for a decentralized multi-UAV system using Behavior Trees. For task assignment, the Consensus-Based Bundle Algorithm (CBBA), a decentralized auction-based task assignment algorithm, is integrated into the Behavior Tree for mission management. We demonstrate our decentralized multi-UAV system in an example surveillance use case in simulation, highlighting the ability to react to new tasks in dynamic environments. The presented onboard mission management enhances autonomous multi-UAV missions as it enables decentralized coordination in dynamic environments.

Keywords: decision-making, autonomy, multi-UAV, mission management, Behavior Trees

1. Introduction

There are several use cases for autonomous unmanned aerial vehicles (UAVs), including surveillance, inspection, delivery, and search and rescue. Missions with single UAVs have been researched to large extends, but are limited in flight time, area coverage, robustness and general mission performance. Increasing the number of UAVs in a flight mission to a multi-UAV mission where the UAVs intentionally cooperate on a common mission goal, reduces the mission time and increases mission performance [1]. Distribution of tasks across multiple drones allows for individual drones to be specialized for a certain type of tasks, e.g., sensors can be distributed onto multiple drones instead of one complex, sensor-heavy drone, which potentially reduces costs, increase mission performance, and robustness to single point of failure. Using autonomous multi-UAV systems, in which pilots are only used for monitoring the operation, reduces human workload, and can be useful in supporting emergency response missions, or inspections of infrastructure as in [2].

In multi-UAV systems, coordination describes how multiple UAVs work together to achieve a common mission goal by sharing information, planning tasks and avoiding conflicts. It is one of the major challenges, as it is subject to communication and computational constraints. The coordination of a multi-UAV system can be centralized, where UAVs receive mission commands from a single unit with global knowledge, or decentralized, where UAVs make their own decisions based on their surroundings and communication with the other members of the swarm. A decentralized approach offers more robustness, as it eliminates the single point of failure of a centralized approach, while also

offering more local autonomy, better scalability, and reduced communication requirements to a central unit. The onboard mission management is a key module for coordination of autonomous missions, as it is responsible for mission planning, execution, and monitoring. Task allocation, determining which UAV takes on which task, is an important function of a multi-UAV mission management. Algorithms for task allocation can be grouped in auction-based, optimization-based, or clustering-based [3]. Auction-based algorithms are widely used in decentralized multi-UAV systems as they provide a good compromise between applicability in safety-critical missions, communication requirements and the capability to integrate reactivity. One of those auction algorithms is the Consensus-Based Bundle Algorithm (CBBA) [4], in which the agents bid on bundles of tasks and then resolve their conflicts. The result of the task allocation is a task plan for each UAV that can then be executed by the onboard mission management. An open question addressed in this paper is how such task assignment algorithms can be integrated into autonomy architectures and scheduled for onboard execution alongside other autonomy capabilities. In autonomous systems, Behavior Trees are used as a control structure for task execution instead of Finite State Machines. They provide modularity, reusability, and the ability to react to dynamic changes [5]. Several works utilize Behavior Trees for UAV mission management, but mostly for a single UAV, focusing on using Behavior Trees for the execution of actions. Behavior trees are used coordination in multi-UAV systems in [6], [7], where the authors include the bid generation for task assignment in the Behavior Trees of each UAV, but they still rely on a central unit for task assignment. In this work, we explore the integration of auction-based task allocation algorithms within a skill-based autonomy architecture and Behavior Trees for reactive autonomous mission execution for a fully decentralized multi-UAV system. A widely-used auction-based task assignment algorithm, the CBBA, is integrated into the Behavior Tree used for onboard mission management. The modular design of the Behavior Tree enables switching between different algorithms for future work, as the task assignment subtree can be exchanged easily. The proposed solution is implemented and tested in case studies showcasing the effectiveness and general applicability of the proposed framework, showing the ability to reassign tasks when new tasks arrive.

The contributions of this work can be summarized as follows:

- Presenting a modular framework based on Behavior Trees for decentralized coordination in a multi-UAV system relying only on local autonomy onboard the UAVs
- Demonstrating the applicability of integrating an auction-based task assignment algorithm, the CBBA, into a behavior-tree based onboard mission management of UAVs in a multi-UAV system for dynamic mission environments allowing reassignment of tasks

2. Related Work

2.1 Multi-UAV Mission Management

Multi-UAV mission management has been studied from a variety of perspectives, encompassing mission planning frameworks, task allocation methods, and autonomy architectures for coordinated swarm operations. Comprehensive surveys of UAV swarm coordination strategies, including task allocation, formation control, and swarm intelligence, are provided in [8], [9], covering both civilian and military application domains. These works consolidate a broad range of algorithmic approaches but largely assume persistent communication or homogeneous autonomy capabilities, leaving adaptive coordination under degraded communication less explicitly addressed. Mission planning frameworks for multi-UAV swarm coordination are proposed in [10], [11], demonstrating the benefits of structured mission-level coordination. A broad review of autonomous mission planning and management systems for unmanned vehicles is provided in [12] highlighting the need for enhancing onboard intelligence in multi-UAV systems. Self-organized swarm concepts for autonomous mission planning are explored in [13], proposing a leader-follower hierarchy for the coordination of a multi-UAV system. In contrast to these works, our work focuses on the onboard integration of task assignment within a reactive mission management architecture, enabling decentralized coordination even in the absence of a central coordination instance. The reactive mission management onboard a single drone is implemented using Behavior Trees, on which the following section discusses related work and identifies the gaps to be closed for use in decentralized swarm coordination.

2.2 Behavior Trees for Multi-UAV Mission Management

For autonomous multi-UAV missions using a decentralized coordination approach, decision-making needs to be moved onto the individual UAVs. Multiple works have used Behavior Trees as control structure to model decision-making and action selection in autonomous robotic agents [5], [14]. A Behavior Tree is a rooted, directed tree that consists of nodes representing actions, conditions or control nodes. The tree is evaluated ("ticked") from the root from left to right. When nodes are ticked, they can return *success*, *failure* or *running*. There are two types of nodes: control flow nodes and leaf nodes. Control flow nodes describe how the children of that node are executed, while leaf nodes are either a specific action or a condition node. Control Nodes are for example a Sequence, which returns *success* only if all children return *success* and returns *failure* after the first child returned *failure*. Fallback nodes return *success* after the first child returned *success*. If a node within a Fallback returns *failure* the next node is ticked. Decorator nodes, e.g. Inverter or ForceSuccess, can manipulate the outcome of their child. The most commonly used and well-maintained open-source libraries for Behavior Trees are pytrees[1] and BehaviorTree.CPP[2]. The modular structure of Behavior Trees have promoted their use in robotic applications and projects for autonomous navigation frameworks, such as Nav2 [15] or Aerostack2 [16].

While Behavior Trees have mostly been utilized in single UAV applications, a few recent works used them in multi-UAV applications. In work by Dahlquist *et al.* [6], Behavior Trees have been used in an autonomy architecture for a multi-UAV system in an underground mining environment performing routine inspection tasks. The local autonomy capabilities of the individual UAVs include task execution, risk-aware path planning and reactive navigation. Their framework includes an auction-based task assignment approach. The individual agents compute the costs for bidding on individual tasks, but a central coordination unit is used to allocate the tasks based on the bids by the agents. They use back-chaining, i.e., using a goal condition to generate the required Behavior Tree for executing the inspection tasks based on available actions in an action library. Modularity and flexibility to mission-specific autonomy were key drivers for the authors to use Behavior Trees. The authors highlighted scalability issues in their approach due to using the centralized task allocation approach. The also stress the need for long-horizon planning, which could be achieved by bidding on bundles of tasks.

Another framework for autonomous multi-UAV missions using Behavior Trees was presented in work by Tadewos in [6], and further developed in [8]. They also utilized a market-based auctioning algorithm to assign tasks and used the approach of having a central auctioneer, which collects all bids and selects the winning UAV. In contrast to the previous work, they also include the bidding process in the Behavior Tree. They also use Behavior Tree synthesis to create Behavior Trees for mission-specific behaviors, like pick and place, during the mission. However, their approach was only tested in simple simulations not using a full autonomy flight stack.

In recent work by Aznar *et al.* [18], the authors use Behavior Trees for task assignment and mission execution and present an event triggered control CBBA for robotic applications. They trigger re-evaluation of the bundle generation and conflict resolution if an initial plan was generated, a bid changed significantly, if it can significantly outbid other agents on high-priority tasks, or a certain time has passed. Their Behavior Tree prioritizes task execution. With their solution, the communication rounds could be reduced significantly, showing promising results for using CBBA in a Behavior Tree structure for decentralized multi-robot systems. They test their system in a simulated Search and Rescue Operation using tuned parameters to get optimal results.

In contrast to the works in [6] and [8], our paper presents an approach to use a fully decentralized task assignment algorithm in behavior trees used by the mission management of an autonomy flight stack. Using a decentralized task assignment algorithm enables local decision-making and is more flexible, scalable and robust to the introduction of new tasks. While the approach of integrating CBBA in Behavior Trees in [18] is similar to ours, our integration in a full autonomy flight stack provides a deployable solution and our modular design of the mission Behavior Tree could make integrating new tasks and behaviors easier.

[1] Pytrees: https://py-trees.readthedocs.io/en/devel/index.html
[2] BehaviorTree.CPP: https://www.behaviortree.dev/

2.3 Task Assignment Problem Statement

There are multiple dimensions to the multi-robot task assignment problem, as task constraints, robot capabilities and the environment create different requirements for solution methods. A widely used framework to classify the multi-robot task assignment problem for multi-robot systems was introduced in [19]. Their classification uses three dimensions: robot task capabilities, task cooperation requirements, and temporal task assignment. In this paper, we consider the problem, where one UAV can only execute one task at a time, only one UAV is required for each task, and more tasks than robots are available resulting in a need to generate an execution order. This is classified as Single Task, Single Robot, Time-Extended Assignment (ST-SR-TA). The multi-robot task assignment problem is usually formulated as an optimization problem. Given a set of N_t tasks, and a set of N_u agents, the goal is to assign all tasks to the agents while maximizing the global reward for executing the tasks. The local score function c_{ij} of assigning task j to agent i depends on the tasks (vector of tasks $\boldsymbol{x}_i$) and the order of the tasks (vector path $\boldsymbol{p}_i$), which summed up makes up the global objective function (Equation 1) to be maximized.

$$\max \sum_{i=1}^{N_u} \left(\sum_{j=1}^{N_t} c_{ij}(\boldsymbol{x}_i, \boldsymbol{p}_i) x_{ij} \right) \tag{1}$$

Where x_{ij} is a binary variable describing whether a task j is assigned to agent i. The score function needs to return a value greater than zero and usually depends on the path, e.g. time to travel to a task or path length. There are multiple approaches for solving the task assignment problem for multi-agent systems, including optimization-based, auction-based or learning-based approaches. Auction-based approaches are usually used for decentralized systems, due to their scalability, flexibility and communication efficiency. Their distributed nature allows local decision-making through iterative bidding and consensus rounds to achieve the task assignment. Auction-based algorithms include CBBA [4], PI [20] or CNP [21]. They usually rely on reliable communication between the agents. Multiple extensions to these algorithms exist, typically to include task-based constraints, such as time-windows. While most researchers test their algorithms in simple simulation scenarios, testing is usually focused on the algorithm itself and not on the integration of such an algorithm in an autonomous multi-UAV system.

3. Methodology

3.1 System Architecture

For autonomous multi-UAV missions using a decentralized coordination approach, each UAV must have its own onboard decision-making capability. Therefore, each UAV must be able to communicate with the other UAVs, plan its own mission and execute the tasks within the mission. These onboard autonomy capabilities are typically located in the mission management system. In prior work [22], we developed a mission management system based on Behavior Trees for a single UAV for dynamic environments and demonstrated its capability of reacting to dynamic events in flight tests. The system is based on a three-layered skill-based architecture, consisting of a deliberation, executive and functional layer, which is commonly used in autonomous systems [23]. Flight behaviors such as taking off or flying to a waypoint are implemented as skills in a skill manager in the executive layer, which sends commands to the autopilot in the functional layer. These skills are available for execution by the mission manager in the deliberation layer responsible for decision-making in the autonomous system. The decision-logic of the mission manager is formally modelled and implemented as a Behavior Tree using the Open-Source library BehaviorTree.CPP. The developed system is very modular enabling easy integration of new skills for autonomous flight in the skill manager or new decisions for planning or mission execution in the mission manager. In this work, we extend this mission management framework presented for a single UAV to a decentralized multi-UAV system and integrate an auction-based task assignment to ensure autonomous coordination of tasks. We use location-based tasks, which provide a foundation for more complex tasks in future work.

Figure 1 displays an overview of the system architecture. An operator provides the multi-UAV system with tasks, which all UAVs can access. Each UAV is equipped with an autopilot, currently PX4, and the autonomy flight stack, consisting of the mission manager and the skill manager. The mission manager of each UAV is responsible for task allocation, plan generation and plan execution, which triggers the respective skills included in the skill manager.

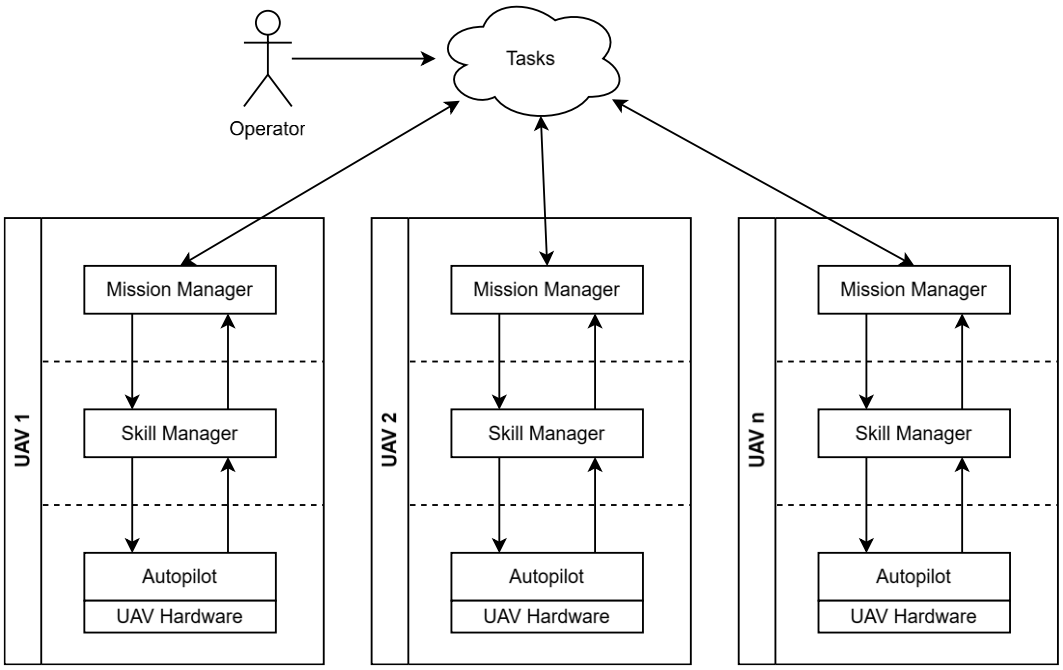


Figure 1: System overview. Each UAV has a mission manager responsible for decision-making, a skill-manager holding the information about flight skills, and an autopilot for execution of the skills. An operator defines the tasks, which are then made available to each UAV.

3.2. Behavior Tree for Task Assignment and Execution

The general high-level control structure for the onboard mission management is a Behavior Tree in the mission manager. In this work, we extend the Behavior Tree in the mission manager developed for a single UAV in prior work [22] to include decentralized coordination between multiple UAVs. Auction-based task assignment algorithms are commonly used for decentralized multi-UAV systems, with CBBA being a popular approach for long-horizon task assignment. CBBA consists of two phases, bundle generation and conflict resolution, which are repeated until consensus over the assignment is reached.

A schematic Behavior Tree for task assignment using CBBA and execution of the mission is shown in Figure 2. The left-hand side of the tree represents the task assignment resulting in an ordered list of tasks to be performed by the UAV. If new tasks are available, the execution of CBBA is triggered. To support missions in dynamic environments with new tasks, the developed Behavior Tree structure for mission management must support re-evaluation of the task assignment. Therefore, the task assignment algorithm is integrated into a reactive node of the Behavior Tree, which allows continuous re-evaluation of the task assignment nodes while executing the mission plan. For CBBA, the tree loops over bundle generation and conflict resolution until consensus is reached and a final task list for execution is produced. The right-hand side of the Behavior Tree shows the mission execution, which iterates through the list of tasks. As this work limits the tasks to location-based tasks, a simple “Waypoint” flight skill is triggered, in which the UAV flies to the commanded waypoint in a direct line. This could be extended with more complex task execution behaviors as more complex tasks are introduced in the future. If the last task is completed, checkLastTask returns failure and ends the mission execution. Before and after this Behavior Tree, takeoff and land nodes must be executed for a full mission.

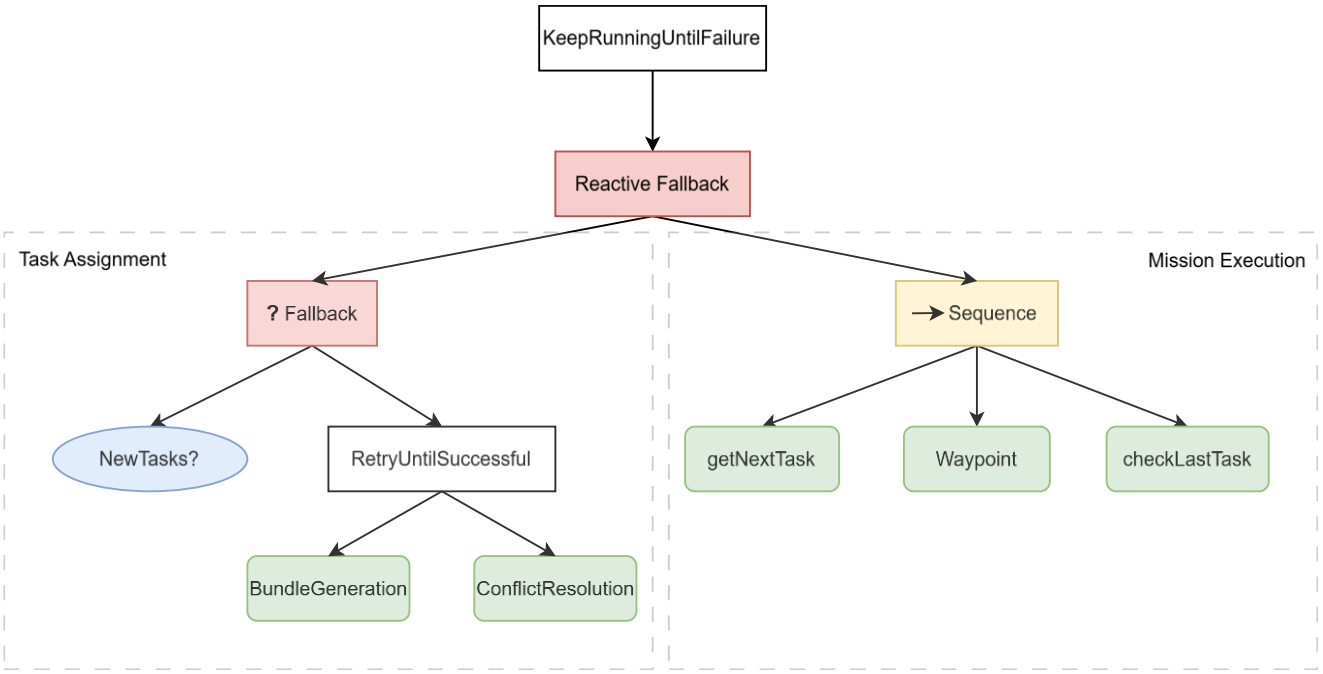


Figure 2: Schematic Behavior Tree for task assignment and mission execution. The left-hand side in the reactive fallback node represents the task assignment including the CBBA with its two phases Bundle Generation and Conflict Resolution, whereas the right-hand side represents the mission execution.

3.3. Task Assignment Algorithm

This section will describe the underlying task assignment algorithm in more detail. The implemented task assignment algorithm chosen for this is the Consensus-Based Bundle Algorithm (CBBA) algorithm, as proposed by Choi et al in [4]. This algorithm was chosen due to its ability to handle decentralized task assignment, scalability and communication efficiency. Many researchers have extended the CBBA tailoring it to the requirements of specific mission and task constraints. Therefore, the basic version of the CBBA also builds a good foundation for future work in decentralized task assignment. Due to the modular nature of Behavior Trees, the task assignment subtree of the mission Behavior Tree can be exchanged easily to implement a more advanced task assignment algorithm. The purpose of this work is to show the general implementation of a completely decentralized coordination approach in Behavior Trees used for UAV mission management.

CBBA consists of two phases: the bundle construction phase and the conflict resolution phase. In the bundle generation, each agent generates a bundle of tasks and updates this bundle later on. Each agent has a bundle, which is a list of tasks ordered based on when they were added and the path list, which is ordered based on when the tasks should be executed. Tasks are inserted into the path list where they generate the best score improvement. The score is calculated based on the scoring function, which is a time-discounted reward function describing the arrival time to a task along the path, as used in the original CBBA. A different scoring function can be used, as long as it satisfies the condition of diminishing marginal gain, meaning the value of a task does not increase as other elements are added to the set before. As this work also deals with tasks in the form of waypoints, the original score function can also be used as follows:

$$S_i^{p_i} = \lambda_j^{\tau_i^j(p_i)} \overline{c_j} \tag{2}$$

$\tau_i^j(p_i)$ describes the time it takes agent i to arrive at task j after following the path specified in its path list. In this work, we estimate this value assuming a constant speed of the UAVs and the length of the path. $\lambda_j < 1$ is the discounting factor for task j, and $\overline{c_j}$ is a static score for performing task j. $\overline{c_j}$ can be used to introduce task priorities. In this work, λ_j = 0.95 and $\overline{c_j}$ = 1 are used for all tasks, as in the original publication of CBBA. According to the score function in Equation 2, the longer an agent needs to move on a path, the lower the score value will be.

In the bundle generation phase, the agents update their path list and their bundle list based on their bundle, as well as two other lists: a winning agent list and a winning bid list. The winning bids list contains the bids for each task, while the winning agent list contains the information on which agent currently hold the highest bid for that task. The bundle generation terminates when all available tasks have been tried to include in the current path. A task is only included in the bundle and path list if it generates a higher score than the current score of that task in the winning bid list. If new tasks were included in the bundle and path, the winning agent list and winning bid list need to be updated.

In the conflict resolution phase, every agent needs to communicate three lists: the winning bids list, the winning agent list and a list containing the last timestamp for the last information from the other agents. When receiving a message from the other agents, the UAV iterates through the tasks and decides if it needs to update the winning bid and agent list upon receiving a higher bid, leave the task in the winning bid and agent list when winning a bid, or reset the task if mixed signals are received. If a task is released from the bundle and path, all following tasks are released. They also need to be released from the winning bid and winning agent lists. If consensus about the task allocation was not reached, the bundle generation needs to run again.

If consensus is reached, the fallback of the Behavior Tree in Figure 2 returns success and the right side of the Behavior Tree can begin executing the resulting mission in the path list. The nature of this particular Behavior Tree ensures that new incoming tasks will trigger the CBBA again resulting in a new bundle and a new path. It also ensures that the CBBA is only executed again when necessary. For the design of this particular Behavior Tree for mission management, modularity, adaptability and interpretability were of high importance. The left side of the Behavior Tree can be exchanged with different task assignment algorithms as long as the output of the task assignment is also a list of tasks to be executed. Breaking down the task assignment algorithm in the nodes according to the phases of the CBBA helps understand what each UAV does during the task assignment process.

3.4. Experimental Setup

We demonstrate our system in simulation, using multi-rotor UAVs equipped with the PX4 autopilot software in the Gazebo simulator, in an exemplary surveillance use case. The general mission is to visit multiple points of interest (POIs), which represent the tasks. The POIs are detected by an external sensor and communicated to the multi-UAV system. To fulfill the mission, the UAVs must distribute the tasks and then execute the mission. While visiting the POIs, several new POIs can become available, requiring new tasks to be integrated into the existing distribution. This will be solved by the task assignment which is integrated in the mission management system onboard each UAV. For that we use the Behavior Tree from Figure 2, but add a takeoff node before and a land node after the subtree for a realistic mission. We assume that all UAVs can communicate with all other UAVs. We use ROS2 for communication between the UAVs. In ROS2, the UAVs are separated using different namespaces. The communication between the skill manager and the autopilot is handled via MAVROS. Mission and skill manager use ROS2 action interfaces for executing flight skills and ROS2 topics for data updates. ROS2 communication is also used for exchanging the necessary messages during the bidding of the CBBA. In the consensus phase of the CBBA each agent publishes the three lists needed for consensus on their respective topic. For this, a custom ROS2 message WinningList was created, which contains the winning bid list, winning agent list and timestamp. Each mission manager has listeners on the other UAVs WinningList topic. Figure 3 visualizes the communication setup. For evaluating our setup we chose two example scenarios, one containing two UAVs and ten tasks, and one containing three additional dynamic tasks where tasks are introduced after a certain period of time.

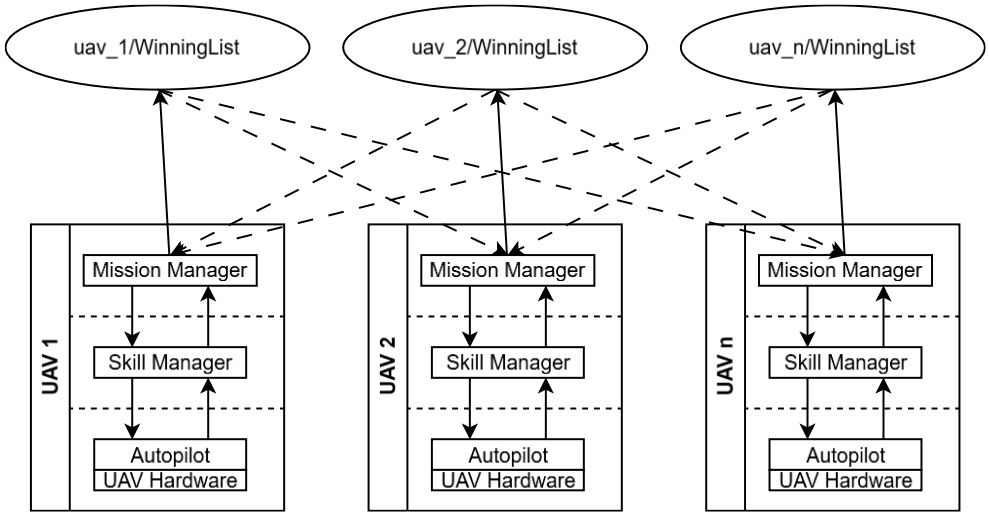


Figure 3: Communication Setup for CBBA in multi-UAV system using a ROS2 topic WinningList with a custom namespace for each UAV.

3.4.1. Static Scenario: Two UAVs, Ten Static Tasks

The static scenario consists of ten tasks and two UAVs. The starting positions for the UAVs are (100,0) and (300,0). The flight speed of the UAVs is 2 m/s. The area in which the tasks are distributed is 500 m x 500 m. The task locations are known and displayed in Table 1. This simple scenario is chosen to highlight the task assignment process and visualize the resulting task assignment.

Table 1: Location of tasks for the static scenario

Task No	Position x[m]	Position y [m]
0	50	70
1	150	100
2	250	150
3	350	70
4	450	150
5	200	350
6	100	250
7	300	480
8	400	310
9	50	340

3.4.2. Reactive Scenario: Two UAVs, Ten Static Tasks, Three Dynamic Tasks

In the reactive scenario, we add incoming tasks to the static mission, which the UAVs then need to distribute. The original tasks from the scenario remain, but we add three new tasks after the mission was started. Table 2 shows the locations of the new tasks to be added, and the timestamps of when tasks are introduced after starting the mission.

Table 2: Location of additional tasks and the respective times of introduction after mission start for the reactive scenario

Task No	Position x [m]	Position y [m]	Time when task becomes available [s]
10	420	420	120
11	100	450	180
12	350	200	240

4. Results

In this section, we show the results of the flight simulations for both the static and reactive scenario highlighting the reaction to changes in the task distribution due to new tasks. The static scenario is used to analyze the task assignment process in more detail.

4.1. Static Scenario: Two UAVs, Ten Static Tasks

The resulting flight paths of the mission are shown in Figure 4. The blue flight path represents UAV-1, while orange represents UAV-2. The task locations are marked in green and with their respective number based on Table 1. The flight path shows that UAV-1 finishes Tasks 0,1,5,6,7,9 while UAV-2 finishes Tasks 2,3,4,8 as assigned in the task assignment process. UAV-1 had a total flight time of 388 seconds from start to Task 7. UAV-2 had a total flight time of 305 seconds from start to final task.

The assignment rounds are displayed in Table 3 with the path list and winning agent list of each UAV after each phase (bundle generation and conflict resolution) of each round of the CBBA. Note that the path contains the numbers of the tasks in planned execution order, while the winning agent list contains the agent that this UAV thinks is best suited for the task. This table explains the task assignment process in this specific scenario. After the first bundle generation, both UAVs have ten tasks in their bundle and path, which is also reflected by their winning agent list. For conflict resolution, they exchange their winning agent and winning bid list. As Tasks 0 and 1 are closer to UAV-1 than UAV-2, UAV-1 wins those tasks for its bundle. As Task 3 is the closest task for UAV-2, UAV-2 wins the task along with Task 2. All tasks after Task 2 for UAV-2 and Task 1 for UAV-1 were released from the respective bundles of the UAVs. In the following bundle generation of round 2, both UAVs create a bundle of eight tasks, as two are already won by each of them. In the conflict resolution phase, UAV-1 wins most of the bids, while UAV-2 has to release most new tasks from its bundle. In the next bundle generation in the third round, UAV-2 can add two more tasks, resulting in a bundle size of four, while UAV-1 has six tasks in its bundle. No tasks were added in the following bundle generation phase, resulting in consensus being reached after four rounds.

Table 3: Task Assignment Process for the static scenario. The path and winning agent list are shown after each phase (BG = Bundle Generation, CR = Conflict Resolution) of the CBBA in the respective rounds that are needed to reach consensus. If the winning agent list is not updated in a round, consensus is reached.

	UAV-1	UAV-2
Round 1		
BG – path	0;1;2;3;4;8;7;5;6;9	3;2;1;0;6;9;5;7;8;4
BG – winning agent list	1;1;1;1;1;1;1;1;1;1	2;2;2;2;2;2;2;2;2;2
CR – path	0;1	3;2
CR – winning agent list	1;1;2;2;0;2;2;0;0;2	1;1;2;2;1;0;0;1;1;0
Round 2		
BG – path	0;1;6;9;5;7;8;4	3;2;6;9;5;7
BG – winning agent list	1;1;2;2;1;1;1;1;1;1	1;1;2;2;1;2;2;2;1;2
CR – path	0;1;6;9;5;7;8;4	3;2
CR – winning agent list	1;1;2;2;1;1;1;1;1;1	1;1;2;2;1;1;1;1;1;1
Round 3		
BG – path	0;1;6;9;5;7;8;4	3;4;8;2
BG – wining agent list	1;1;2;2;1;1;1;1;1;1	1;1;2;2;2;1;1;1;2;1
CR – path	0;1;6;9;5;7	3;4;8;2
CR – winning agent	1;1;2;2;2;1;1;1;2;1	1;1;2;2;2;1;1;1;2;1
Round 4		
BG – path	0;1;6;9;5;7	3;4;8;2
BG – winning agent list	1;1;2;2;2;1;1;1;2;1	1;1;2;2;2;1;1;1;2;1
CR – path	0;1;6;9;5;7	3;4;8;2
CR – winning agent list	1;1;2;2;2;1;1;1;2;1	1;1;2;2;2;1;1;1;2;1

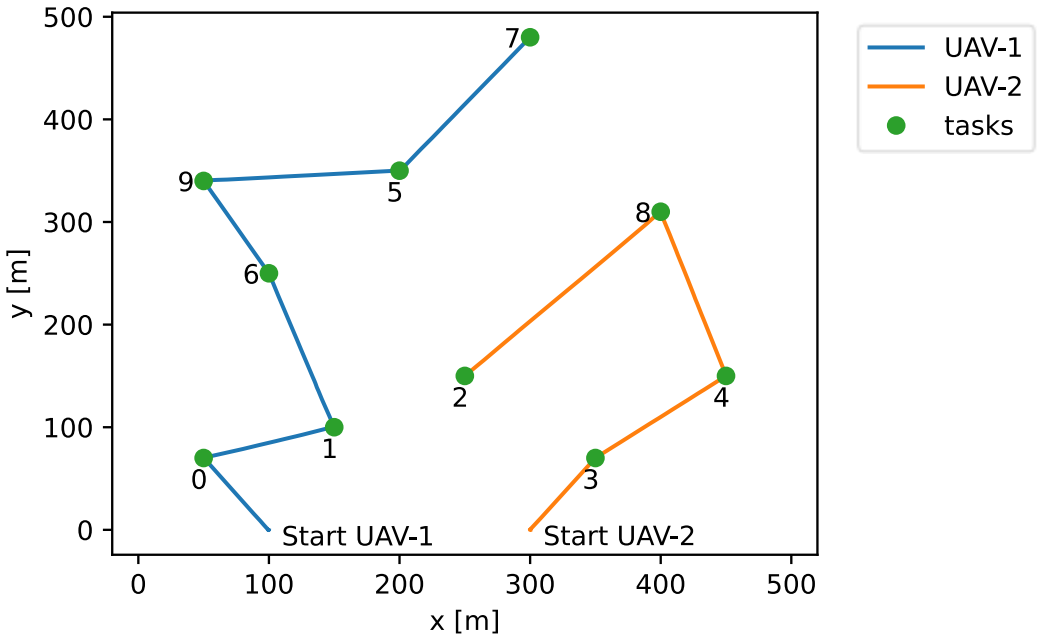


Figure 4: Resulting flight path of both UAVs for the static scenario. The blue line represents the flight path of UAV-1, whereas the orange flight path shows the flight path of UAV-2. The green points represent the task locations, numbered according to Table 1.

4.2 Reactive Scenario: Two UAVs, Ten Static Tasks, Three Dynamic Tasks

The flight path of the two UAVs in the reactive scenario are shown in Figure 5, where the blue flight path represents UAV-1 and the orange flight path UAV-2. The green points show the task locations. The dynamic tasks are introduced after a certain period of time according to Table 2. Their locations are marked in red, while the time when the tasks were introduced is marked on each flight path with a grey square. Each new task arrival retriggers the task assignment process in the left-hand side of the Behavior Tree. The CBBA rounds are conducted using the current position of the UAV, only including the tasks that still need to be performed. For Task 10 it took three rounds to reach consensus, for Task 11 it took two rounds and for Task 12 it took five rounds. The resulting flight paths shows that UAV-1 takes over Task 11 while UAV-2 takes over Task 10 and 12. In comparison to the static assignment of the ten original tasks, UAV-1 takes over Task 2, while UAV-2 takes over Task 7, as these fit better in the updated path with the new tasks. Table 4 shows the resulting path order for both UAVs after task assignment was triggered by the new task arrivals. The table shows that due to the introduction of Task 10, UAV-2 now takes over Task 10 and 7, while UAV-1 takes Task 2, which was initially in the bundle of UAV-2. The simulation with dynamically appearing new tasks demonstrate that the UAVs are able to negotiate on who is best suited for a newly introduced task and agree on a good solution, as the flight paths don't show any obvious detours. UAV-1 had a flight time of 469 seconds from start to final task, while UAV-2 had a flight time of 465 seconds.

Table 4: Task assignment for both UAVs as found in the path list before and after new tasks were introduced in the reactive scenario

	UAV-1	UAV-2
Initial path	0;1;6;9;5;7	3;4;8;2
Task 10 – final path	6;9;5;2	8;10;7
Task 11 – final path	6;9;11;5;2	8;10;7
Task 12 – final path	11;5;2	10;7;12

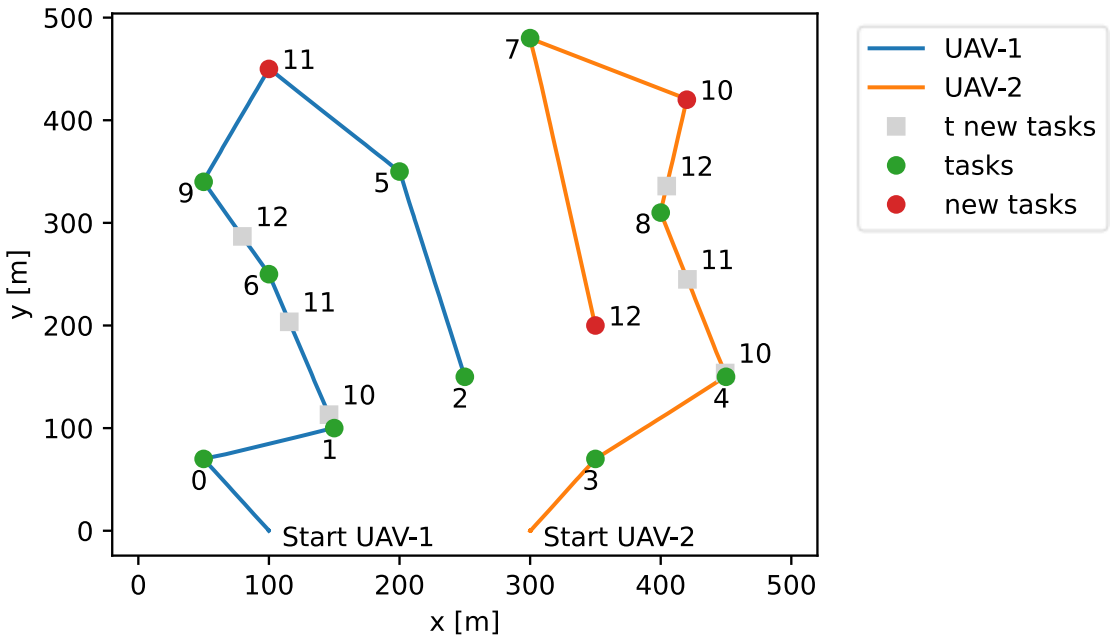


Figure 5: Resulting flight paths of both UAVs in the reactive scenario. The blue line shows the flight path of UAV-1, whereas the orange line displays the flight path of UAV-2. The green points represent the locations of the static tasks (Table 1), while the red points show the dynamic tasks (Table 2). The grey squares show the locations on the flight paths of each UAV when a new task became available. The squares are labeled with their associated task number.

4.3. Evaluation

The experimental results show that the proposed onboard mission management system which included the CBBA for task assignment in a Behavior Tree structure is capable of assigning tasks to the participating UAVs in static and dynamic mission conditions in a decentralized approach without relying on a central controlling unit. Integration of the task assignment in the Behavior Tree enables dynamic reassignment of tasks when new tasks arrived, while maintaining continuous task execution. Reassignment, and the necessary steps to perform CBBA again, was triggered when new tasks arrived to include the new tasks into the paths of the UAVs. Due to the reassignment of all tasks that still needed to be performed, tasks were released from one UAV and taken over by the other to generate a better distribution. This is expected as the CBBA considers the best assignment due to the current positions of the UAVs. As the cost function from the original CBBA optimizes for travel times, those were similar for both UAVs. This still resulted in a similar distribution of travel distances.

Using the Px4-Gazebo simulation framework enhanced the real-world deployment capabilities of our onboard mission management system. Our simulation is a closed-loop software-in-the-loop simulation with a realistic physics simulation, flight controller integration and sensor emulation allowing fast deployment of our system in a flight test.

The current state of the implementation was only tested with two UAVs and requires synchronous communication of the winning bid lists. Therefore, scalability and robustness to large-scale scenarios could not be evaluated. After showcasing the general applicability of CBBA in a Behavior Tree-based mission management in this work, more UAVs and larger numbers of tasks should be tested. As the focus of this work was to demonstrate how integrating the task assignment into the behavior-tree based onboard mission management system can enable task assignment in dynamic environments, different task assignment algorithms and their suitability for different scenarios and applications have not been compared yet.

5. Conclusion and Future Work

In this work, we showed an implementation of a decentralized Behavior Tree-based mission management system for use in multi-UAV missions. We integrated the consensus-based bundle algorithm for decentralized task assignment in the Behavior Tree structure of the mission management system onboard each UAV. The usage of Behavior Trees allows for reactive task reassignment in dynamic environments, while also providing a modular structure for development. We tested our proposed implementation in simulations using multiple multi-rotor UAVs using the PX4 autopilot in the Gazebo simulator. We demonstrated the applicability of our proposed Behavior Tree structure with integrated task assignment and mission execution in two scenarios. The general task assignment process is visualized using a simple scenario with two UAVs and ten tasks, while a second scenario is introduced which uses the simple scenarios as a starting point but adds three tasks dynamically during mission execution. The proposed system was able to integrate these dynamic events into the generated bundles for mission execution demonstrating adaptability to dynamic environments while maintaining continuous mission execution.

The presented decentralized onboard mission management for multi-UAV missions enhances autonomous multi-UAV missions as it enables decentralized coordination in dynamic environments. It serves as a foundation for future work in this field, including testing and improving decentralized task assignment algorithms especially in dynamic environments. The developed system can also be used to improve benchmarking for decentralized task assignment algorithms in realistic simulation environments as modularity supports integrating other task assignment algorithms more easily. With this, bigger scenarios or edge cases can be analyzed in order to find the limitations of this system. This also includes extending the system to deal with more constraints or different tasks, which need to be developed for different scenarios.

6. Contact Author Email Address

Contact Author: Karolin Thomessen, {firstname.lastname}@dlr.de

7. Copyright Statement

The authors confirm that they, and/or their company or organization, hold copyright on all of the original material included in this paper. The authors also confirm that they have obtained permission, from the copyright holder of any third party material included in this paper, to publish it as part of their paper. The authors confirm that they give permission, or have obtained permission from the copyright holder of this paper, for the publication and distribution of this paper as part of the ICAS proceedings or as individual off-prints from the proceedings.

References

[1] G. Skorobogatov, C. Barrado, and E. Salamí, 'Multiple UAV Systems: A Survey', *Unmanned Syst.*, vol. 08, no. 02, pp. 149–169, Apr. 2020, doi: 10.1142/S2301385020500090.

[2] M. Gil-Castilla, A. R. Poma, A. Caballero, and A. Ollero, 'A Cooperative Multi-Uav Framework for Bridge Inspection', in *2025 International Conference on Unmanned Aircraft Systems (ICUAS)*, Charlotte, NC, USA: IEEE, May 2025, pp. 408–415. doi: 10.1109/icuas65942.2025.11007792.

[3] S. Alqefari and M. E. B. Menai, 'Multi-UAV Task Assignment in Dynamic Environments: Current Trends and Future Directions', *Drones*, vol. 9, no. 1, p. 75, Jan. 2025, doi: 10.3390/drones9010075.

[4] Han-Lim Choi, L. Brunet, and J. P. How, 'Consensus-Based Decentralized Auctions for Robust Task Allocation', *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009, doi: 10.1109/TRO.2009.2022423.

[5] M. Colledanchise and P. Ögren, *Behavior Trees in Robotics and AI: An Introduction*. 2018. doi: 10.1201/9780429489105.

[6] N. Dahlquist, S. Nordström, N. Stathoulopoulos, B. Lindqvist, A. Saradagi, and G. Nikolakopoulos, 'Deployment of an Aerial Multiagent System for Automated Task Execution in Large-Scale Underground Mining Environments', *IEEE Trans. Field Robot.*, vol. 2, pp. 465–484, 2025, doi: 10.1109/TFR.2025.3580068.

[7] T. G. Tadewos, L. Shamgah, and A. Karimoddini, 'On-the-Fly Decentralized Tasking of Autonomous Vehicles', in *2019 IEEE 58th Conference on Decision and Control (CDC)*, Nice, France: IEEE, Dec. 2019, pp. 2770–2775. doi: 10.1109/CDC40024.2019.9029554.

[8] J. Song, K. Zhao, and Y. Liu, 'Survey on Mission Planning of Multiple Unmanned Aerial Vehicles', *Aerospace*, vol. 10, no. 3, p. 208, Feb. 2023, doi: 10.3390/aerospace10030208.

[9] G. M. Skaltsis, H.-S. Shin, and A. Tsourdos, 'A Review of Task Allocation Methods for UAVs', *J. Intell. Robot. Syst.*, vol. 109, no. 4, p. 76, Dec. 2023, doi: 10.1007/s10846-023-02011-0.

[10] C. Sampedro *et al.*, 'A flexible and dynamic mission planning architecture for UAV swarm coordination', in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, Arlington, VA, USA: IEEE, Jun. 2016, pp. 355–363. doi: 10.1109/ICUAS.2016.7502669.

[11] J. Autenrieb, N. Strawa, H.-S. Shin, and J.-H. Hong, 'A Mission Planning and Task Allocation Framework For Multi-UAV Swarm Coordination', in *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, Cranfield, United Kingdom: IEEE, Nov. 2019, pp. 297–304. doi: 10.1109/REDUAS47371.2019.8999708.

[12] A. Atyabi, S. MahmoudZadeh, and S. Nefti-Meziani, 'Current advancements on autonomous mission planning and management systems: An AUV and UAV perspective', *Annu. Rev. Control*, vol. 46, pp. 196–215, 2018, doi: 10.1016/j.arcontrol.2018.07.002.

[13] W. Y. H. Adoni *et al.*, 'Intelligent Swarm: Concept, Design and Validation of Self-Organized UAVs Based on Leader–Followers Paradigm for Autonomous Mission Planning', *Drones*, vol. 8, no. 10, p. 575, Oct. 2024, doi: 10.3390/drones8100575.

[14] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, 'A survey of Behavior Trees in robotics and AI', *Robot. Auton. Syst.*, vol. 154, p. 104096, Aug. 2022, doi: 10.1016/j.robot.2022.104096.

[15] S. Macenski, F. Martín, R. White, and J. G. Clavero, 'The Marathon 2: A Navigation System', 2020, doi: 10.48550/ARXIV.2003.00368.

[16] M. Fernandez-Cortizas, M. Molina, P. Arias-Perez, R. Perez-Segui, D. Perez-Saura, and P. Campoy, 'Aerostack2: A Software Framework for Developing Multi-robot Aerial Systems', Sep. 03, 2024, *arXiv*: arXiv:2303.18237. doi: 10.48550/arXiv.2303.18237.

[17] T. G. Tadewos, L. Shamgah, and A. Karimoddini, 'Automatic Decentralized Behavior Tree Synthesis and Execution for Coordination of Intelligent Vehicles', *Knowl.-Based Syst.*, vol. 260, p. 110181, Jan. 2023, doi: 10.1016/j.knosys.2022.110181.

[18] F. Aznar, M. Pujol, and Á. Díez, 'Event-triggered adaptive consensus for multi-robot task allocation', *Comput. Commun.*, vol. 251, p. 108499, Apr. 2026, doi: 10.1016/j.comcom.2026.108499.

[19] B. P. Gerkey and M. J. Matarić, 'A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems', *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, Sep. 2004, doi: 10.1177/0278364904045564.

[20] A. Whitbrook, Q. Meng, and P. W. H. Chung, 'A novel distributed scheduling algorithm for time-critical multi-agent systems', in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany: IEEE, Sep. 2015, pp. 6451–6458. doi: 10.1109/IROS.2015.7354299.

[21] Smith, 'The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver', *IEEE Trans. Comput.*, vol. C–29, no. 12, pp. 1104–1113, Dec. 1980, doi: 10.1109/TC.1980.1675516.

[22] K. Thomessen and P. Nagarajan, 'Reactive Mission and Contingency Management for UAS: Implementation and Flight Testing', in *AIAA SCITECH 2026 Forum*, Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2026. doi: 10.2514/6.2026-2574.

[23] P. Malafosse, A. Albore, J. Guiochet, and C. Lesire, 'Skill-Based Architectures in Autonomous Systems: an Empirical Study', in *2025 IEEE Engineering Reliable Autonomous Systems (ERAS)*, Worcester, MA, USA: IEEE, May 2025, pp. 1–8. doi: 10.1109/ERAS63351.2025.11135549.