Autonomous and Dependable Multi-Agent Systems for the Mission Planning of Multi-UAV Surveillance Missions

Domenico Pascarella, Gabriella Gigante, Salvatore Luongo
CIRA – Italian Aerospace Research Centre
ISVV (Intelligent Systems, Verification & Validation) Laboratory
Via Maiorise, 81043 Capua (CE)
ITALY

d.pascarella@cira.it, g.gigante@cira.it, s.luongo@cira.it

Salvatore Venticinque
Università degli Studi della Campania "Luigi Vanvitelli"
School of Polytechnic and Basic Sciences, Department of Industrial and Information Engineering
Via Roma 29, 81031 Aversa (CE)
ITALY

salvatore.venticinque@unicampania.it

ABSTRACT

Autonomy of Unmanned Aerial Vehicles (UAVs) may improve resilience of fleets when the remote control by human operators is lost because of attacks or failures. Besides, autonomy may be accomplished by agent-oriented approaches, which significantly enhance ability to model, design and build autonomous software systems. Nevertheless, if a fleet of UAVs is equipped with autonomy, it is generally complex to assure that its vehicles are able to guarantee the required level of dependability. This work investigates on agents abilities related to predictability and on design-for-dependability methodology for multi-agent systems. It adopts concepts of multi-agent task allocation, game theory and formal verification to develop a dynamic and decentralized mission planner for a fleet of UAVs, which shall accomplish a persistent surveillance mission. The integrated utilization of formal models and agents programming techniques is proposed to allow a real-time scheduling of the agents behaviours. The framework of Markov games and the Distributed Stochastic Algorithm are used to synthesize a coordination mechanism that governs the interactions between the agents. In the end, some simulation results and an efficiency analysis are discussed.

1.0 INTRODUCTION

In a broad sense, this work is about the contrast and the coupling between two significant system properties: **dependability** and **autonomy**. Dependability is a fundamental property of **critical systems**, for which a failure in the accomplishment of one or more functions that are required to it may imply outcomes with a remarkable severity. On the other hand, autonomy is the ability of independently act, wherein the independence is from human operators. Autonomous systems cover a large range of functions and complexities, from sensor networks to space rovers, including elderly care assistants and unmanned vehicles. These complex artefacts cannot always be tele-operated because of operational constraints or because of a demanding interaction with non-expert humans.

Anyway, complex autonomous systems that are able to choose and execute high-level actions without human supervision are not yet ready in all applications, especially in critical applications. Indeed, one of the major drawbacks in the utilization of such systems is the difficulty to predict and verify their behaviour. While autonomy offers improved capabilities at a reduced operational cost, there are concerns about the verification of such autonomous systems in a reliable and cost-effective manner. Moreover, the introduction of critical autonomous systems has elicited the need to ascertain their dependability, in order to assess the trust that they will successfully and satisfactorily perform their missions and they will not cause any

catastrophes. In detail, autonomous management usually requires both **accurate reactions** and **timely reactions** to environmental events. The former require **intelligence**, which is the capability to react to the greatest number of events in order to allow for the achievement of the mission goals. The latter shall prevent failures with serious outcomes (e.g., loss of life).

Besides, it may turn out to be difficult to impose cooperation and to reach agreements in systems wherein the agents are self-interested (i.e., they have conflicting goals). In fact, their coordination is a key factor for multi-agent problem solving. Indeed, the development of multi-agent systems is complicated by two main factors: the increasing complexity that is associated with their system-level and agent-level properties, and the need to prove their dependability in critical contexts. To ensure the dependability of a **critical multi-agent system**, we need development techniques that allow us to manage the complexity of the design and to formally verify the correctness of agent interactions while performing critical coordination activities. Indeed, a **formal verification process** would provide significant evidence of **safety assurance** (for safety-critical applications) and **mission assurance** (for mission-critical applications), but it might also discover the design restrictions that should be imposed on the behaviour to guarantee the required properties (both for the single agents and for the system as a whole).

The previous considerations motivated the high-level objective of this work, which is to develop a **design-for-dependability** methodology of a critical multi-agent system, i.e., a systematic methodology in order to:

- synthesize and analyse individual and coordinating strategies for multi-agent dependable systems;
- provide (possibly formally) evidence for safety assurance and/or mission assurance;
- guarantee that the system will be predictable, namely, it will work as intended and will be compliant with individual and global properties that are required.

Such a methodology has several application areas, such as intelligent spacecraft, intelligent transportation systems, fly-by-wire systems for aircraft, etc. In this work, it has been experimented on the design of an **online mission planner for** surveillance missions that engage two or more unmanned aerial vehicles (UAVs), i.e., for **multi-UAV surveillance missions**. Such missions are particularly effective if the region of interest (ROI) is large and/or the target areas to check are several. Indeed, the potential advantages of this paradigm are the following:

- Multiple simultaneous interventions – A multi-UAV team may simultaneously collect data from multiple locations.
- Global performance and mission effectiveness – A multi-UAV team may split up in order to efficiently cover a large area, optimizing available resources. A coordinated decision strategy may efficiently allocate the group resources over the multiple targets in order to prevent the over-service or the under-service of the targets. Thus, the surveillance performance of a set of UAVs fulfilling a single mission is expected to exceed the sum of the performance of the individual UAVs.
- Global information – Every UAV may share sensor information with the fleet via a communication network. The entire fleet may act based on a global (or network-centric) situation awareness.
- Reliability – The fleet may reconfigure its distribution architecture in order to minimize the performance degradation that are produced by vehicles failures.
- Cost efficiency – A single vehicle to execute some tasks could be an expensive solution when comparing to several low cost vehicles.

The reference multi-UAV surveillance for this work is a wide-area **persistent surveillance**, that is a special target discovery mission. Persistent surveillance is essentially a set of continuous monitoring tasks, which arise when the guarded region cannot be fully covered by a stationary set of guard entities and which differ

from traditional exploration tasks due to the perpetual need to cover the supervised region.

We have adopted concepts and technologies of multi-agent task allocation, distributed control systems, decision theory, game theory, vehicle routing and formal verification to develop a **dynamic** and **decentralized mission planner** for a fleet of homogeneous UAVs, which shall accomplish a persistent surveillance mission. Firstly, the mission planning problem for multi-UAV persistent surveillance has been formally stated as a stochastic constrained optimization problem.

We have proposed a systematic methodology based on the integrated utilization of formal models and agents programming techniques, such as **BDI** (Belief-Desire-Intention) paradigm. The control loop for the reasoning of the BDI vehicle agents has been formally specified by means of a collection of **concurrent automata**. Such a formalism is potentially apt to a real-time context since a real-time scheduler may use the concurrent automaton model to decide about the current intention to be executed.

Then, we have applied the framework of **Markov Decision Processes** (MDPs) and **Markov games** to model the negotiation amongst the agents of the multi-UAV mission planner and to synthesize a coordination mechanism. The **Distributed Stochastic Algorithm** (DSA) has been customized to implement the distributed negotiation protocol that governs the interactions amongst involved agents.

In the end, the distributed coordination protocol has been implemented in a multi-agent programmable modelling environment in order to execute several simulations in different scenarios and to attain different measures of the numerical quality of the deliberated plans. For example, an efficiency metric has been assessed in terms of surveillance performance deviation with respect to an equivalent centralized mission planner.

2.0 PROBLEM STATEMENT

The reference case study of this work is the problem of **autonomous planning** for multi-UAV persistent surveillance, i.e., a persistent surveillance by means of an unmanned aerial system (UAS), which is composed of a set of homogeneous UAVs. Homogeneity of the vehicles is both for their payload capability and for their dynamic features. We suppose that there are a number of geographically distributed targets in a given ROI and the UAVs are in charge of perpetually visiting these targets. Figure 2-1 graphically illustrates a two-dimensional qualitative description of the required mission. Here, no-fly zones are regions that shall not be crossed by vehicles for any reason, whereas threats mark hazardous space regions that may be transited, but they threaten the reliability (mission success) or even the safety (the vehicle survival).

In addition, each target shall be visited with a certain frequency for the fleet to accomplish the mission. Then, a **joint surveillance strategy** shall be generated for the team to satisfy the visitation frequencies of the target areas. Generally speaking, the planning of this strategy shall process two types of plans: the **path plans**, which allow the UAVs to reach the target areas requested by the remote operators; the **activity plans**, that allow the payloads to execute the necessary activities in a target area for the fulfilment of the mission requested by the remote operators. This work does not concern the planning for the payload activities and examines only the problem of path planning for multi-UAV persistent surveillance. The associated planner is also named **multi-UAV persistent surveillance planner** or just persistent surveillance planner.

2.1 Requirements Analysis

The overall system architecture for multi-UAV persistent surveillance conforms to the classic scheme with the ground segment (i.e., the control station for remote operators) and the flying segment (i.e., the UAVs). The architecture is shown in Figure 2-2. A suitable communication infrastructure includes both a data link between an aircraft and the ground station and the data link amongst the vehicles. Hence, both Ground-To-Vehicle (G2V) and Vehicle-To-Vehicle (V2V) communications are enabled.

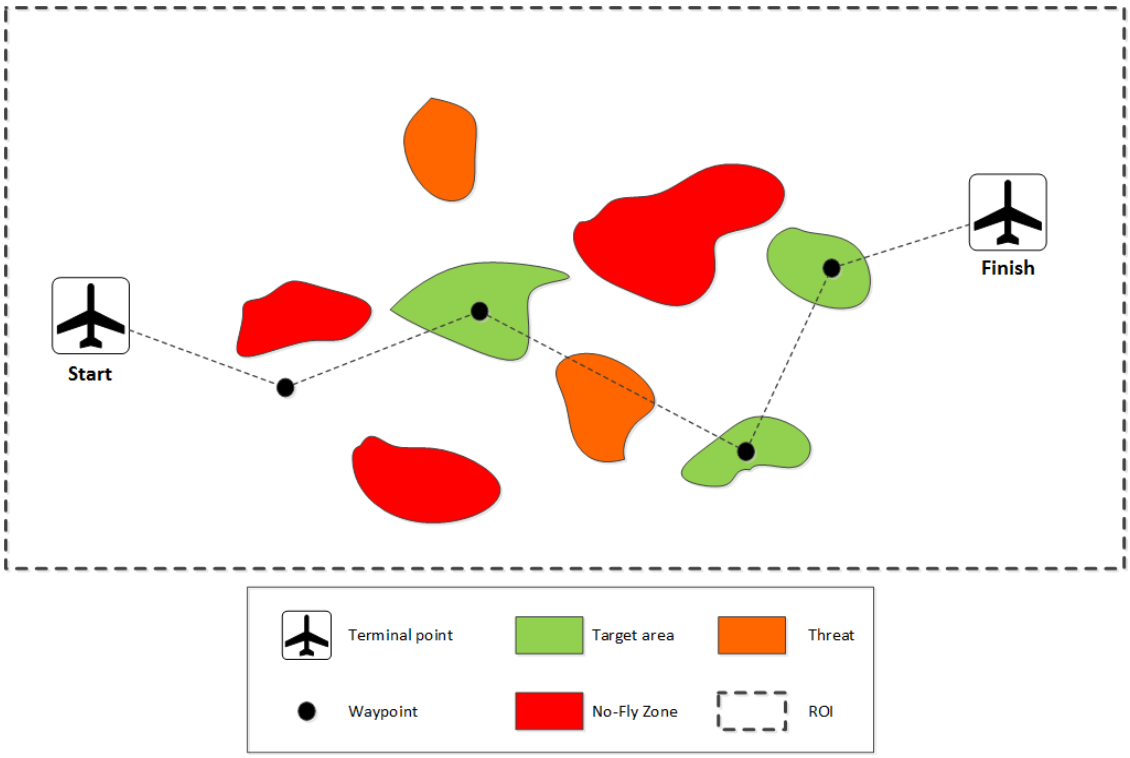

Figure 2-1: Two-dimensional graphical representation of the problem of multi-UAV persistent surveillance planning.

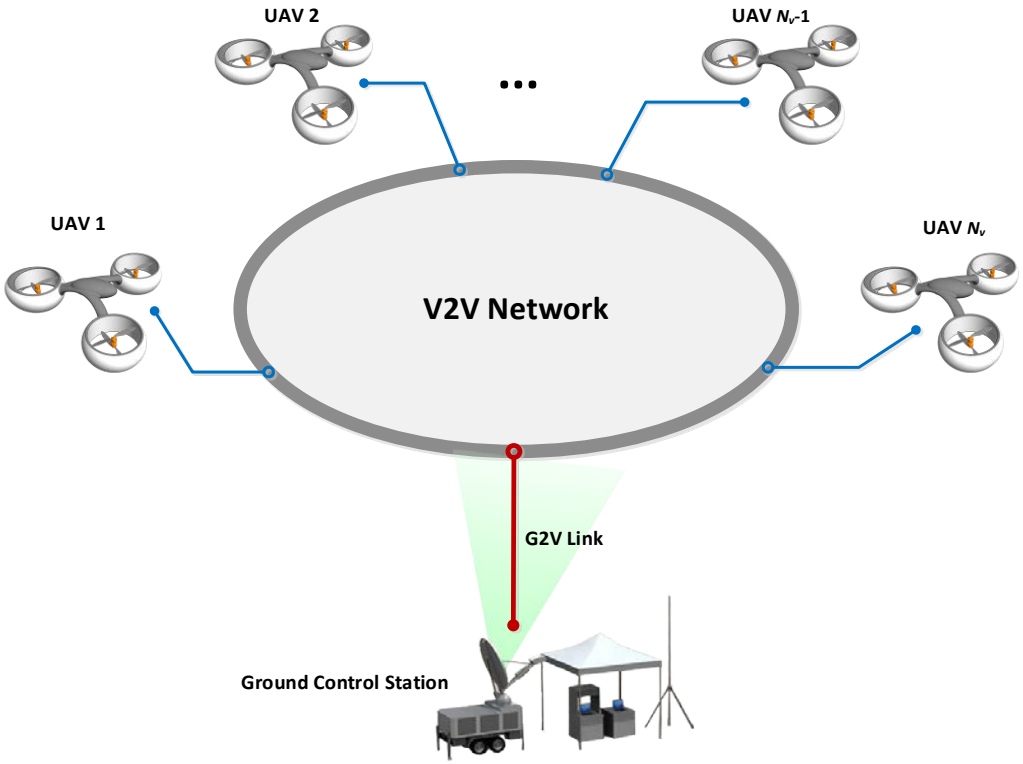

Figure 2-2: System architecture for multi-UAV persistent surveillance.

In particular, the ground operators shall be able to specify high-level missions and to require their fulfilment (strategic control). It stands to reason that the proposed task is an instance of the more general problem of **distributed control for decoupled systems**. Indeed, this problem concerns the control of a **team**, namely, a composite system which is decomposable in controlled subsystems, that are spatially distributed and dynamically decoupled, but have some common objectives and constraints that somehow make them interact between each other [1]. Below, the terms “fleet” and “team” will be used as synonyms.

Two requirements are essential for the persistent surveillance planner:

- **Robustness** – The persistent surveillance planner shall dynamically operate and shall accommodate the changes of the mission definition or of the perceived situation in a graceful manner.
- **Timeliness** – The persistent surveillance planner shall find a solution in a reasonable time frame.

Thus, a dynamic planning strategy for the mission is needed: the global planner shall operate in a dynamic fashion to account any variation within the mission definition or within the perceived situation. Fast and efficient planning is required since the state of the environment may abruptly and radically change during the course of the mission. Moreover, the planner shall exhibit a predictable behaviour due to the potential critical nature of the faced problem, which may be both mission-critical and safety-critical. Consequently, the reference planning problem shall be dynamic and shall be handled as a **real-time planning-and-replanning** problem. The proposed problem has also to be stated as an optimization problem since the global team planning is expected to produce plans which somehow maximize the mission effectiveness.

The necessity to replan in real-time makes such a problem especially challenging from a computational point of view. In principle, its inherent complexity does not allow to guarantee finding the most effective plan in the limited available time for planning in real-time. A possible solution of this issue is the adoption of a **hierarchical decomposition** and to define a **multi-UAV mission planning** problem, which is discussed in the following sections.

2.2 Mathematical Formulation of the Problem

The reference UAS includes a team of N_v homogeneous UAVs, which shall move within the region of interest $\mathbb{W}$. We assume a **point-mass model** of the UAVs, thus, yaw, pitch and roll angles may be ignored in the dynamic state of the i-th vehicle. Besides, we make the following assumptions to discretize the problem:

- the UAVs move only in a plane (i.e., at constant altitude h) at a constant speed v_0;
- the targets $\mathbb{T}_k \in \mathbb{T}$ a coincide with a single representative waypoint $\mathbf{t}_k$;
- the no-fly zones set $\mathbb{W}_{\mathrm{NFZ}}$ and the threats set $\mathbb{W}_{\mathrm{TH}}$ are the union of disjoint straight prisms with a limited or an unlimited height.

The **single-UAV mission plan** $\mathcal{M}_i$ of the i-th UAV is an assigned route that the i-th UAV of the team shall carry out. It may be defined as an ordered sequence of points in $\mathbb{W} - \mathbb{W}_{\mathrm{NFZ}}$. By taking into account the previous assumptions, the discrete-space mission plan $\mathcal{M}_i$ has the following expression

$$\mathcal{M}_i = \left\{ \left(\mathbf{x}_{i_0}, \mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \cdots \right) \middle| \mathbf{x}_{i_a} \in \left\{ \mathbb{T} \cup \mathbb{C}\left(\mathbb{W}_{\mathrm{NFZ}}^{(h)} \right) \right\}, \ \forall a \in \mathbb{N} \right\}, \tag{1}$$

wherein $\mathbb{C}\left(\mathbb{W}_{\mathrm{NFZ}}^{(h)} \right)$ is the contour of projection of $\mathbb{W}_{\mathrm{NFZ}}$ at the altitude h. Clearly, $\mathbb{C}\left(\mathbb{W}_{\mathrm{NFZ}}^{(h)} \right)$ is a polygon. Instead, the **joint mission plan** or **global mission plan** $\mathcal{M}$ of the team is defined as the set of all the single-UAV mission plans in the fleet.

The proposed problem has also to take into account the mission type and its properties in order to solicit a

rational choice of the possible routes. In detail, the first issue in devising surveillance autonomous tasks is to agree what it means to do a good surveillance job and to define a surveillance performance index. The quality of a surveillance service is generally made up of the accuracy, the quantity and the timeliness of the acquired information about the reference phenomenon into the areas of interest. Several performance metrics are possible for persistent surveillance missions [2]. Here, we adopt the **ECI** (**expected cost of ignorance**) operator [3], which is based on the information utility concept. In particular, if the surveillance task has to detect an event E into a target, then the greater is the absence period of the guard from the zone, the higher is the occurrence probability of E in that zone without being discovered. Thus, the not visitation cost of a target in a given interval is related to the occurrence probability and the occurrence cost of E.

If a target $\mathbf{t}_k$ is observed only before the instant t_0 and after the instant t_1, the **ECI** of $\mathbf{t}_k$ in $[t_0, t_1]$ is

$$\mathrm{ECI}_k(t_0, t_1) = \int_{t_0}^{t_1} P_{E,k}(t) \cdot C_{E,k}(t_1 - t)dt, \tag{2}$$

wherein $P_{E,k}(t)$ is the probability density function of E at the instant t in $\mathbf{t}_k$, whereas $C_{E,k}(t_1 - t)$ is the cost for the occurrence of E at the instant t in $\mathbf{t}_k$, given that t_1 is the next observation instant of $\mathbf{t}_k$. If $t^* \in [t_0, t_1]$ is the occurrence time of E in $\mathbf{t}_k$, the sooner E occurs within the non-observation window $[t_0, t_1]$, the more expensive is the cost because of the greater delay $t_1 - t^*$ for its detection. $P_{E,k}(t)$ and $C_{E,k}(t_1 - t)$ are specific mission features and represent the degrees of freedom for the mission modelling. They are a function of $\mathbf{t}_k$, so, they enable a ranking of the targets according to their importance (i.e., their effect on E).

In general, given a mission plan $\mathcal{M}$, we denote with

$$\mathbb{O}_k^{(\mathcal{M})} = \left\{\left(\left[t_{k_0}^{(\mathcal{M})}, t_{k_1}^{(\mathcal{M})}\right], \left[t_{k_2}^{(\mathcal{M})}, t_{k_3}^{(\mathcal{M})}\right], \cdots\right) | t_{k_l}^{(\mathcal{M})} < t_{k_{l+1}}^{(\mathcal{M})}, \forall l \in 2\mathbb{N}\right\} \tag{3}$$

the sequence of the time intervals during which $\mathbf{t}_k$ is not observed by any guard by means of $\mathcal{M}$. If $\mathcal{M}$ begins at $t = 0$, the **ECI** of $\mathbf{t}_k$ for the mission plan $\mathcal{M}$ may be expressed as

$$\begin{aligned} \mathrm{ECI}_k^{(\mathcal{M})} = \mathrm{ECI}_k^{(\mathcal{M})}(0, +\infty) = \sum_{l=1}^{+\infty} \mathrm{ECI}_k\left(t_{k_l}^{(\mathcal{M})}, t_{k_{l+1}}^{(\mathcal{M})}\right), \\ l \in 2\mathbb{N}, \left[t_{k_l}^{(\mathcal{M})}, t_{k_{l+1}}^{(\mathcal{M})}\right] \subseteq \mathbb{O}_k^{(\mathcal{M})}. \end{aligned} \tag{4}$$

wherein $\left[t_{k_l}^{(\mathcal{M})}, t_{k_{l+1}}^{(\mathcal{M})}\right]$ is the generic non-observation window in $\mathbb{O}_k^{(\mathcal{M})}$. Moreover, the **ECI** of $\mathbf{t}_k$ for $\mathcal{M}$ at t is

$$\begin{aligned} \mathrm{ECI}_k^{(\mathcal{M},t)} = \mathrm{ECI}_k^{(\mathcal{M})}(0, +\infty) = \sum_{l=1}^{t} \mathrm{ECI}_k\left(t_{k_l}^{(\mathcal{M})}, t_{k_{l+1}}^{(\mathcal{M})}\right), \\ l \in 2\mathbb{N}, \left[t_{k_l}^{(\mathcal{M})}, t_{k_{l+1}}^{(\mathcal{M})}\right] \subseteq \mathbb{O}_k^{(\mathcal{M})}. \end{aligned} \tag{5}$$

The total expected cost of ignorance of the mission plan $\mathcal{M}$ for the ROI $\mathbb{W}$ and the set of targets $\mathbb{T}$ is the sum of the temporal sums of the expected costs of ignorance of all the N_t targets by means of $\mathcal{M}$, i.e.

$$\mathrm{ECI}_{W,\mathbb{T}}^{(\mathcal{M})} = \sum_{k=1}^{N_t} \sum_{t=0}^{+\infty} \mathrm{ECI}_{k}^{(\mathcal{M},t)}. \tag{6}$$

In the end, the **joint mission planning problem** of a fleet of UAVs for persistent surveillance is the search of a joint mission plan $\mathcal{M}^*$ such that the reference constraints are fulfilled and

$$\mathcal{M}^* = \underset{\mathcal{M}}{\operatorname{argmin}}\, \mathrm{ECI}_{W,\mathbb{T}}^{(\mathcal{M})}. \tag{7}$$

3.0 DESIGN OF THE MULTI-UAV MISSION PLANNER

A decentralized design of a multi-UAV mission planner employs a peer-to-peer team without a team leader. No single entity solves the global problem alone: there are no bottlenecks and the distributed system is fully robust. Besides, this control mechanism may overcome the limited communication constraint since it is able to operate using limited amounts of communication. Every vehicle is equipped with its mission planner, which interacts with the mission planners of the other vehicles over the V2V network.

Hence, the design is focused on a **decentralized** and **dynamic MAS-based mission planner** for the multi-UAV persistent surveillance problem. In detail, the MAS-based model of the system is made up of the following basic entities: vehicle agents or guard agents. These are physically embodied agents: each of them conceptually represents a UAV and is deployed on a smart device within the UAV. The vehicle agents are in charge of the mission planning for their respective UAVs. Thus, they are the distributed components for the global task of mission planning for multi-UAV persistent surveillance.

In the following sections, we report the BDI-based design of the vehicle agents by precisely describing its key components (beliefs, desires, intentions and the functions of the control loop), as addressed in [4].

3.1 Beliefs of Vehicle Agents

The knowledge of the vehicle agents consists of a set of beliefs, which shall change on the occurrence of new perceptions. In our model, perceptions are: data read from vehicle sensors; commands from the remote operator by means of the mission template (i.e., the description of the required mission by the ground operator); messages from other vehicle agents, which control other UAVs in the same team. Table 3-1 points out the designed set of beliefs for the vehicle agents.

Table 3-1: Designed beliefs for the vehicle agents.

Label	Description	Origin
$\mathbb{W}$	ROI	Mission Template
$\mathbb{W}_{\mathrm{NFZ}}$	Set of no-fly zones	Mission Template Navigation Database
$\mathbb{W}_{\mathrm{TH}}$	Set of threats	Mission Template Navigation Database
$\mathbb{T}$	Set of target areas	Mission Template
t_{OBS}	Observation period of targets	Mission Template
v_0	Cruise speed of the vehicles	Mission Template
$\mathbf{x}_i(t)$	Current position of the vehicle	On-Board Sensors

Label	Description	Origin
f_i	Residual fuel of the vehicle	On-Board Sensors
$f_{\min}$	Safety margin for the fuel consumption of the vehicles	Mission Template
$P_{E,k}$	Probability density function of E over the k-th target	Mission Template
$C_{E,k}$	Occurrence cost of E over the k-th target	Mission Template
$\mathbf{x}_{\mathrm{HOME}}$	Final vehicles position	Mission Template
$\mathcal{F}(t)$	Current system failures	Failure & Health Monitoring
$\mathcal{D}_{\mathrm{LAST}_i}$	Last data sent by other UAVs of the fleet	V2V Network

3.2 Events for Vehicle Agents

The vehicle agents shall exhibit a reactive behaviour with respect to some designed events, which are all associated to their perceptions. The designed events for the vehicle agents are listed in Table 4-2.

Table 4-2: Designed events for the vehicle agents.

Event	Description
Threat	Perception of a new threat
No-Fly Zone	Perception of a new no-fly zone
Fuel Level	Perception of a variation of the residual fuel level
Vehicles Message	Perception of a new message from other UAVs in the fleet
Command	Perception of a new command from the GCS

3.3 Goals and Desires of Vehicle Agents

Each vehicle agent shall aim to find an optimal itinerary kin order to solve the joint mission planning problem in definition (7). More generally, by taking into account also the possible minimization of other functions in addition to $\mathbf{ECI}$ (such as the total time spent or the fuel consumed for surveillance and the penalty factor for threats crossing), the potential goals of a vehicle agent shall be:

- minimization of the $\mathbf{ECI}$ of the mission plan;
- minimization of the total time spent (or the fuel consumed) by means of the mission plan;
- minimization of penalties due to the crossing of threats;
- minimization of all the previous indices.

The desires of a vehicle agent coincide with its possible goals. We suppose that the desires cannot be autonomously changed by the agent, but only by the remote operator. Thus, the option generation function is automatically included in the belief revision function as regards the choice of the current goals by the remote operator.

3.4 Intentions of Vehicle Agents

The envisaged intentions for the vehicle agents are the following:

- *Reactive* – It enables the reactive behaviour. It allows the agent to react to the environment changes, i.e., to new perceptions.
- *Planning* – It is the intention for the itinerary planning. Therefore, it does not coincide with the planning of the BDI vehicle agent in the control loop, but it provides the policy or a list of policies for the surveillance itinerary.
- *Safe−Path* – It looks for a safe itinerary into the library of possible itineraries.
- *Broker* – It selects an itinerary amongst the valid ones in the library. It looks for the itinerary that optimally satisfies the activated goal.

Thus, the designed vehicle agents are not standard BDI agents since, although the possible intentions are dependent on the set of options (i.e., the goals), do not directly coincide with them. The behaviours of the scheduled intentions shall be related to the active goals.

3.3 Control Loop of Vehicle Agents

The reasoner engine of a vehicle agent shall be in charge of revising the current beliefs on the occurrence of new perceptions and of starting the required plan according to the revised beliefs and the active goals. Within this section, the term “plan” does not mean the mission plan that shall be provided by every vehicle agent as a policy to be performed for persistent surveillance. On the contrary, it means the policy that the BDI agent determines in order to achieve the intentions. Hence, the two concepts not necessarily will coincide and their relations will depend on the design of intentions.

According to the control loop for the practical reasoning of a BDI agent [4], the design of the i-th vehicle agent has to address the following basic functions:

- the belief revision function $\mathbf{brf}(Bel, \rho)$;
- the option generation function $\mathbf{options}(Bel, Int)$;
- the filtering function $\mathbf{filter}(Bel, Des, Int)$;
- the means-ends reasoning function $\mathbf{plan}(Bel, Int, \Upsilon_i)$;
- the reconsidering function $\mathbf{reconsider}(Int, Bel)$;
- the function $\mathbf{impossible}(Int, Bel)$;
- the function $\mathbf{sound}(\pi_i, Int, Bel)$.

The belief revision function is simply a lookup method defined by a static table, which allows the agent to update the beliefs in Table 4-1 according to the current perceptions. Moreover, since the desires of a vehicle agent coincide with its possible goals, the option generation function is automatically included in the belief revision function as regards the choice of the current goals by the remote operator.

The computational model of the vehicle agents is represented by a set of single finite-state non-deterministic automata, which interact through the values of some common variables. The automata are also asynchronous, i.e., at any time a possible transition may delay for an indefinite amount of time and no hypotheses may be made on when a particular state transition takes place inside an automaton. In detail, every automaton of the set models an intention of the agent.

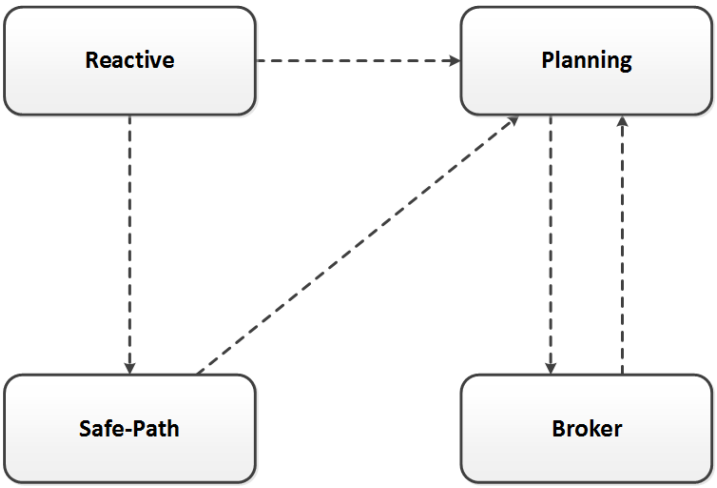

Figure 3-1: Concurrent automata of the intentions of vehicle agents.

Figure 3-1 illustrates the concurrent automata of the intentions and their relationships. As a matter of principle, the automata wait for perceptions, lookup for new beliefs and spawn new automata (i.e., new intentions). Spawn refers to a function that loads and executes a new child process: the current process may wait for the child to terminate or may continue to asynchronously execute. Thus, intentions may spawn and schedule other intentions, whose priority will depend on the active goals. As an example, Figure 3-2 and Figure 3-3 respectively show the structure of *Reactive* and *Planning* intentions.

At the startup, the agent firstly schedules *Reactive* intention, which allows the agent to react to the new perceptions. Moreover, it spawns other automata (i.e., intentions) according to the triggering perceptions and the current state. In case of a new perception, the belief revision function is executed and several behaviours are possible. For example, if a new threat or a new no-fly zone is notified, *Safe−Path* intention is spawn if the threat or the no-fly zone is critical (i.e., if the new no-fly zones or threats are violated by the current itinerary), otherwise *Planning* intention is spawn. Instead, if the amount of fuel is notified as below the safety threshold, the itinerary to go back home is activated.

Planning intention computes the list of itineraries that optimize the achievement of active goals for the current mission and for the current beliefs and insert them in a library of available itineraries. Besides, *Planning* intention may recursively spawn a child process if an alternative candidate is available for the final brokering. On the occurrence of a new candidate itinerary, a new planning algorithm shall be launched and scheduled as an intention itself. Each spawned child shall wait on a common synchronization barrier at the end of its workflow. As a consequence, the other intentions have to “push down” the synchronization barrier in order to spawn *Planning* intention whenever needed.

Broker intention looks for and enables the itinerary that optimally satisfies the activated goal. If a valid itinerary is not found, it starts a replanning by spawning *Planning* intention, namely, by closing the synchronization barrier. Instead, *Safe−Path* intention is spawn whether new no-fly zones or threats are violated by the current itinerary. Through this intention, the agent looks for a safe itinerary amongst the last ones found. It notifies the result and activates the itinerary that satisfies the new constraints. If no safe itineraries exist, it spawns *Planning* intention for a replanning in order to search new available itineraries.

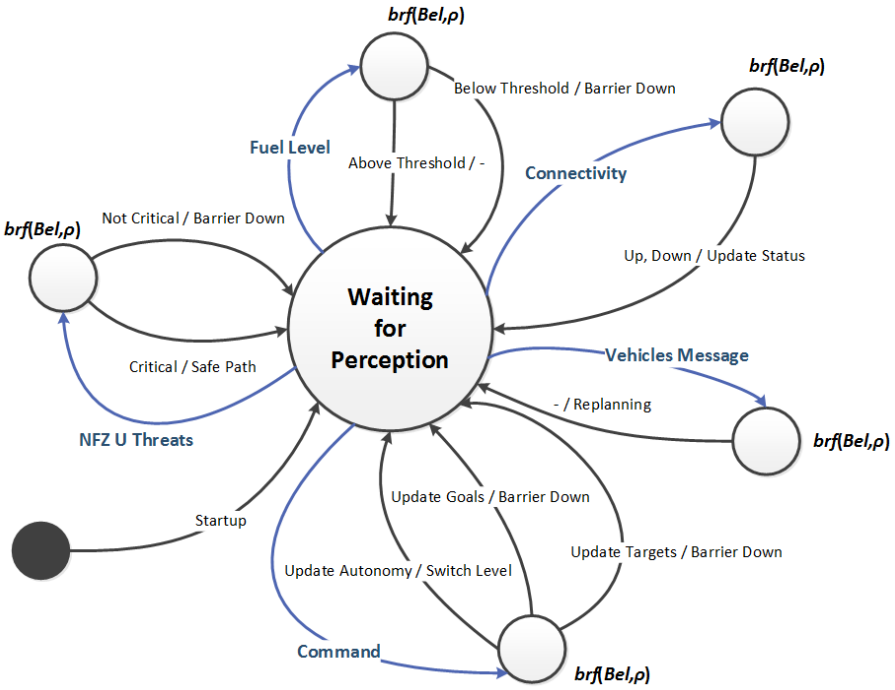

Figure 3-2: Concurrent automaton of *Reactive* intention of vehicle agents.

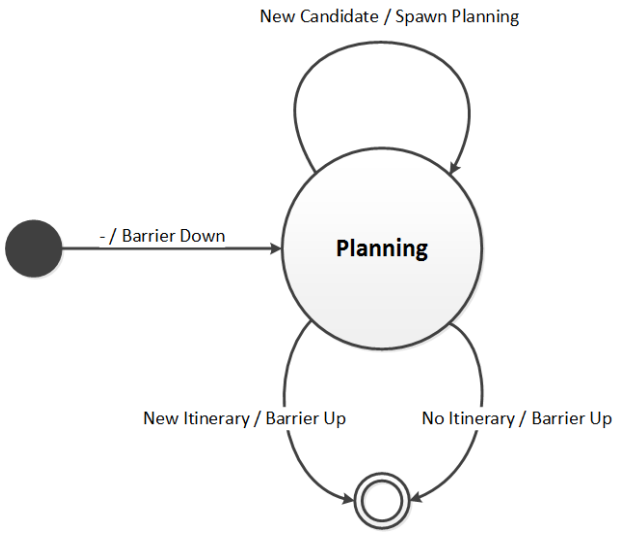

Figure 3-3: Concurrent automaton of *Planning* intention of vehicle agents.

The concurrent automata directly provide the implementation of the filtering function $\mathbf{filter}(Bel, Des, Int)$ and the means-ends reasoning function $\mathbf{plan}(Bel, Int, Y_i)$. Moreover, they implement all the other required functions for the control loop of a BDI agent since:

- they address the scheduling of new intentions when a change in the beliefs occurs;
- the function $\mathbf{impossible}(Int, Bel)$ is true when the *Planning* intention is not able to find new itineraries with the current beliefs;
- the function $\mathbf{sound}(\pi_i, Int, Bel)$ is true when the current itinerary satisfies the new constraints (i.e.,

the new beliefs).

The proposed vehicle agent may be viewed as a **hybrid BDI** agent. Indeed, the possible intentions and the plans to achieve them are established at design time by means of the set of component automata and their mutual relations. Nevertheless, each vehicle agent shall execute a suitable planning algorithm in *Planning* intention in order to update the library of possible itineraries at runtime. Hence, in contrast with "pure" BDI systems, a vehicle agent shall partly perform planning according to first principles.

In the end, the proposed design of the vehicle agents is potentially apt to a real-time context. For instance, a real-time scheduler may use the concurrent automaton model to decide about the current intention to be executed: the available actions of the vehicles agent are represented by the intentions scheduling.

4.0 FORMAL VERIFICATION OF THE MULTI-UAV MISSION PLANNER

In our reference scenario, a real-time scheduling of the intentions of the vehicle agents is needed. In other words, the intentions shall be the possible tasks that a real-time scheduler of the target platform shall accept. Consequently, the multi-UAV mission planner shall be predictable, i.e., the termination time of each intention shall be deterministic and shall be known in advance. The real-time scheduler shall choose for the intention to be scheduled or to be terminated according to a pessimistic (i.e., worst-case) estimation of the durations of each task, including *Planning* intention. The latter naturally represents the most complex task from a computational point of view and requires the greatest amount of resources and execution time. Hence, the basic design principle for the timeliness of the vehicle agents shall be the following: if *Planning* intention exceeds its assigned deadline, it shall be cut and the partial itineraries that have been processed until the cut time shall be kept and added to the library of available itineraries. For this reason, we have to infer the maximum allowable time to search the itineraries for persistent surveillance.

We employ timed automata to implement the timed BDI model of the vehicle agents. A timed automaton is a finite-state machine extended with clock variables. As regards the tool, we have used the UPPAAL (http://www.uppaal.org/) framework. It is an integrated toolbox for the modelling, the simulation and the verification of real-time systems [5]. In UPPAAL, a system is modelled as a network of timed automata, i.e., a composition of timed automata (also named processes) that evolve in parallel.

4.1 Definition of the Timed BDI Model of the Vehicle Agents

The concurrent automaton model of the BDI vehicle agents has been translated into a network of parallel timed automata. Thus, every intention of the agent is represented by an UPPAAL process. As an example, Figure 4-1 points out the graphical representations of *Reactive* process.

In addition to the physical intention processes, a *Perceptor* process has been specified. It is in charge of the generation of the external stimuli for the vehicle agent and triggers the intention processes according to the stimulus by means of channel synchronizations. It is parametrized with respect to an integer constant *deadline*, which sets the global limit within the vehicle agent has to thoroughly react to the stimulus. If this limit is crossed without the agent reaction, the process enters in the *Error* state, from which no recovery actions are feasible (i.e., a hard deadline miss occurs). Otherwise, if the agent computes a reaction in time, the *reaction_completed* channel is activated and the *Perceptor* process is led into the *Idle* state.

Reactive process consumes the external stimuli produced by *Perceptor* process. It carries out the belief revision task and selects the most suitable action to handle the beliefs variation. It is parametrized with respect to the duration *BR_time* of the belief revision task. We suppose that this duration is a constant, coherently with the design of the belief revision function as a lookup method. According to the external stimulus and to the related beliefs change, *Reactive* process triggers the other intentions by activating the proper channel. The processes of the other intentions are all parametrized with respect to their own predetermined duration. As regards *Broker* process, it is supposed to trigger *Planning* intention at most one

time, namely, *Planning* process cannot be turned on again if a valid itinerary has not been found.

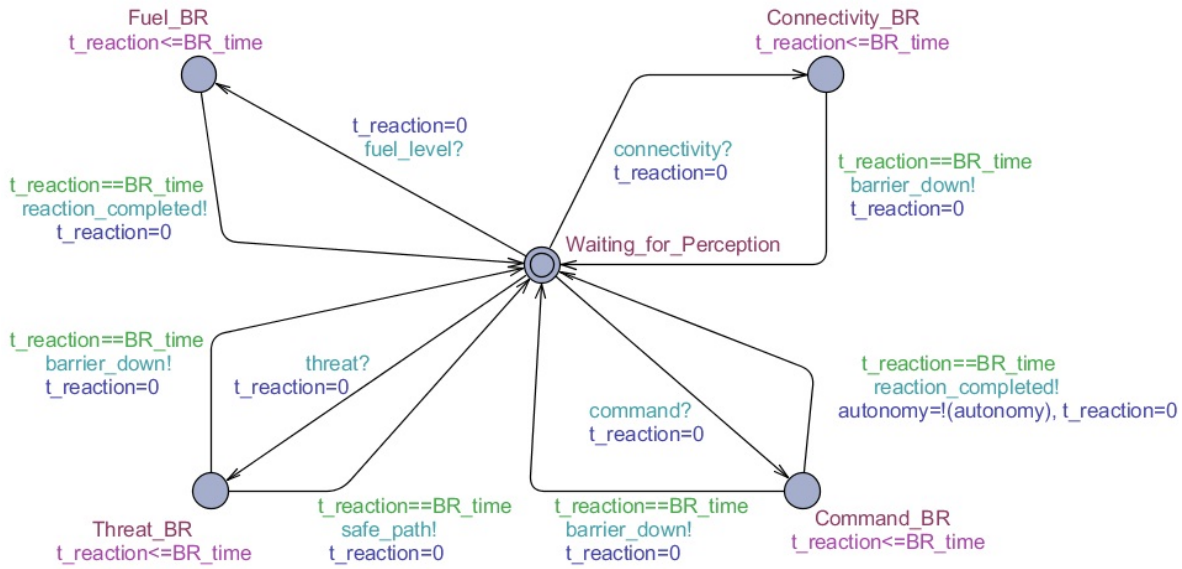

Figure 4-1: Timed automaton of *Reactive* process in UPPAAL.

4.2 Formal Analysis of the Timed BDI Model of the Vehicle Agents

The formal analysis of the timed BDI model has been accomplished by means of UPPAAL query language, which is a subset of TCTL (Timed Computation Tree Logic) and formally specifies the properties to check. Given that UPPAAL environment supports only integer variables, the global parameter deadline is set to 10 and the task durations (except the task duration of *Planning* process, that is the analysis target) are set to 1.

The real-time requirement that the vehicle agent shall necessarily satisfy is the invariable completion of the reaction to an external stimulus within the hard deadline. This constraint is equivalent to the condition that the error state of the *Perceptor* process is never reached, so, the agent always meets the deadline. This is a safety property, i.e., a property of the form “something bad will never happen”. Thus, the property to check for the real-time correctness of the vehicle agent is the following safety property

$$A\square(\mathrm{not}(Perceptor.Error)) \tag{8}$$

The formal analysis by means of UPPAAL model checker has proved that the maximum granted duration for the task is 2 in the above mentioned context. Greater values cannot guarantee the compliance with the property specified in equation (8). Figure 4-2 shows an instance of the UPPAAL simulation trace with a deadline miss of the vehicle agent. Such a trace refers to a simulation with a duration of 4 for *Planning* process.

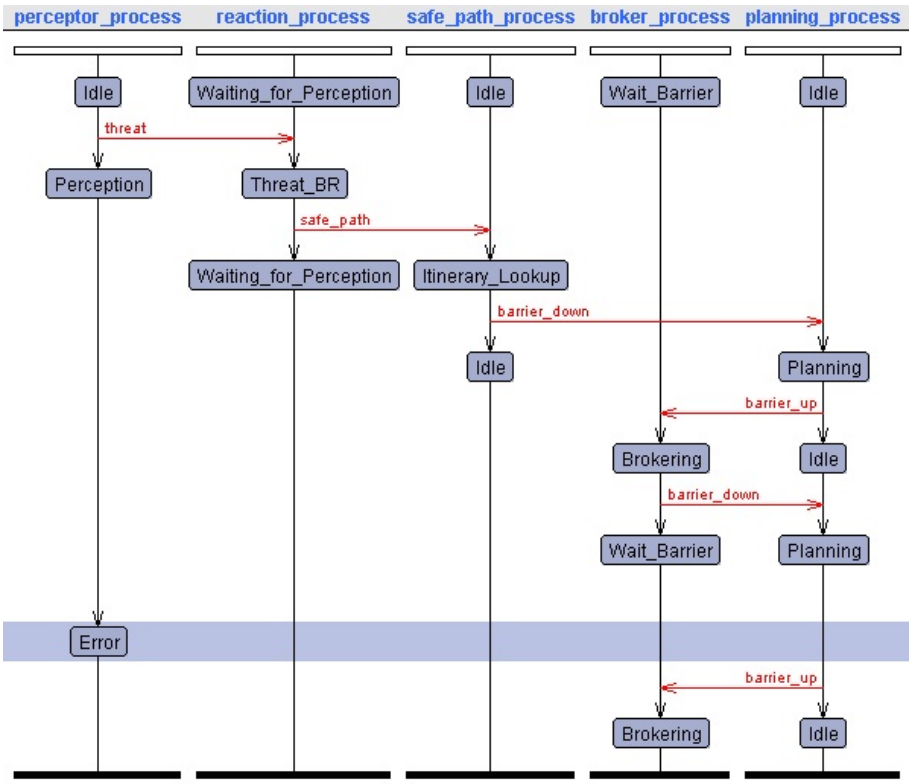

Figure 4-2: Instance of UPPAAL simulation trace with a deadline miss by the vehicle agent.

5.0 DESIGN OF THE COORDINATION OF MULTI-UAV MISSION PLANNING

The design of the multi-UAV mission planner is completed by the macro design. This consists in addressing a coordination mechanism amongst the vehicle agents, which shall ensure that agents' activities retain the desired relationships (i.e., that are compliant with system requirements) and shall maximize the global effectiveness by avoiding destructive or unhelpful interactions and by exploiting any positive interaction. Given that a decentralized architecture has been selected for the multi-UAV mission planner, a **competitive** paradigm is adopted. Indeed, competition fits with a decentralized and scalable method. Thus, vehicle agents shall be antagonistic, even if they shall not be real opponents because their interaction shall achieve a global goal, namely, the minimization of the mission ECI.

In detail, starting from the formal statement of the problem, a utility-based framework is needed and vehicle agents shall optimize both an internal (i.e., private) utility function and a global (i.e., shared with all the other vehicle agents) utility function. For this purpose, the coordination of vehicle agents shall occur by means of a **negotiation protocol**, in order to process an acceptable agreement on the division of resources (i.e., the targets to guard) amongst the vehicle agents starting from their proposed individual strategies. In other words, negotiation is the means by which the goals and the private utilities of single vehicle agents are aligned with the global objectives. In this way, the competitive MAS is able to efficiently control the decentralized mission planning for multi-UAV persistent surveillance.

In addition, the environment entails a decision-theoretic approach for the global mission planner and for each vehicle agents. Hence, the global mission planner may be represented by a Markov Decision Process (MDP). Then, the control over the decision variables of the problem (i.e., the allocation of surveillance tasks in the joint mission plan) is given to the set of self-interested vehicle agents, each one responsible for the mission planning of its own vehicle. For this purpose, game theory may be efficiently used to model sequential decision-making problems amongst rational and competitive players (i.e., decision makers) since it may be

seen as an extension of decision theory to multi-agent environments. Thus, a game-theoretic approach is used for the design of the proposed negotiation protocol.

The following steps have to be performed in order to attain an engineering solution to this problem:

- design of the formal model of the game (e.g., identification of players and their utility functions);
- development of a negotiation protocol, that governs the interactions between the players, defines which proposals are allowable by the negotiating agents on the basis of prior negotiation history and dynamically leads the game to converge to a desired rational outcome, such as a Nash equilibrium.

5.1 Formal Model of the Coordination Game

Given that the game to state shall coordinate a set of agents to control the decision variables of an MDP, it shall be a stochastic or Markov game [6]. Indeed, stochastic games represent the direct extension of MDPs to multi-agent environments. Thus, the problem of multi-UAV mission planning is a distributed dynamic task allocation problem, that is defined as a stochastic game Γ. This game may be also represented as a sequence of stage games $\gamma(\bar{t})$, one for each decision epoch $\bar{t}$. Every stage game is a one-shot game and represents the static game that arises in a certain state of Γ. The stage game has its own strategies spaces and utilities functions, which are the enabled portions of the strategies spaces and utilities functions in the state of Γ at $\bar{t}$.

At a given decision epoch $\bar{t}$, we assume that the vehicle agents observe the current state in order to retrace the full history of the game. This implies that every agent totally knows what the other agents did at the previous time steps, including their accomplished surveillance routes and the last previous actions (i.e., the outcomes of their last stage games). Starting from the past routes, the agents are completely informed about the utilities that the other players have gained up to the current time step. On the contrary, the vehicle agents do not know what the other agents are going to do at the current decision epoch (i.e., their selected current actions by the outcomes of their current stage games). This is coherent with the hypothesis of strategic form for the games Γ and γ.

The formal model of the proposed coordination game Γ shall be designed to point out all the basic elements of a Markov game, as addressed in [7]. Firstly, the temporal utility function of the target $\mathbf{t}_k$ at the time step t by means of the joint plan $\mathcal{M}$ is set as

$$\mu_{\mathbf{t}_k}^{(t)}(\mathcal{M}) = \beta^{\mathrm{ECI}_k^{(\mathcal{M})}(0,t)}, \tag{9}$$

wherein $\beta \in (0,1]$ is a discount factor that weights the payoff of observing earlier the target.

As regards the agents utility functions, note that there exists a strategy for every stage of the stochastic game, hence, for every possible state of the environment. Given the large number of possible states, the processing of a strategy may be a computationally expensive process and may not comply with the timeliness requirement. Indeed, the solution of a general stochastic game by means of exact techniques is impractical if the state space is large. This is even more so due to the infinite horizon nature of the process, which prescribes an undefined-length plan for every single-agent strategy of each stage game. To reduce computational complexity and to allow a real-time coordination, the infinite horizon objective of the single stage games is restricted to a fixed interval in order to compute the optimal joint strategy only over a limited temporal horizon, named **planning horizon** (or decision window). Thus, the original stochastic game Γ is approximated with a sequence of finite horizon static games $\bar{\gamma}$, which are a windowing of the stage games γ of Γ. This type of approximation is suitable since we expect all environment states to be quiescent, namely, changes in the state do not significantly affect the long-run payoffs to the agents.

In particular, if t_w is the length of the planning horizon, the game $\bar{\gamma}$ is designed to have the following global

utility function

$$\mu_{\bar{\gamma}(\bar{t})}(\pi)=\sum_{\mathbf{t}_k\in\mathbb{T}}\sum_{t=\bar{t}}^{\bar{t}+t_W}\mu_{\mathbf{t}_k}^{(t)}(\pi). \tag{10}$$

For the design of the utilities of the vehicle agents, we apply the **marginal contribution protocol** [8]. In detail, we define the marginal contribution of the vehicle agent i to a target $\mathbf{t}_k$ by means of $\mathcal{M}=(\mathcal{M}_i,\mathcal{M}_{-i})$ as

$$\Delta\mu_{i,\mathbf{t}_k}^{(t)}(\mathcal{M}_i,\mathcal{M}_{-i})=\mu_{\mathbf{t}_k}^{(t)}(\mathcal{M}_i,\mathcal{M}_{-i})-\left(\mathcal{M}_{i_0},\mathcal{M}_{-i}\right), \tag{11}$$

wherein $\mathcal{M}_{i_0}$ is the null plan for agent i, i.e., a plan by means of which i does not observe any target. Then, starting from the marginal contribution, the agent utility function $\mu_{i,\bar{\gamma}}$ by means of π is designed as

$$\mu_{i,\bar{\gamma}(\bar{t})}(\pi_i,\pi_{-i})=\sum_{\mathbf{t}_k\in\mathbb{T}}\sum_{t=\bar{t}}^{\bar{t}+t_W}\Delta\mu_{i,\mathbf{t}_k}^{(t)}(\pi_i,\pi_{-i}). \tag{12}$$

Thus, the designed utility function of vehicle agents in the lookahead stage games is the sum of its marginal contributions for all the targets. Such a choice is meaningful since it may be proved that the static game $\bar{\gamma}(\bar{t})$ is an exact potential game [9] and an exact potential function of its is its designed global utility function $\mu_{\bar{\gamma}(\bar{t})}$. Thus, a joint plan $\mathcal{M}^*$ that maximizes $\mu_{\bar{\gamma}(\bar{t})}$ is also a Nash equilibrium of the game $\bar{\gamma}(\bar{t})$ and a possible protocol for the proposed coordination of multi-UAV mission planning shall find a Nash equilibrium of $\bar{\gamma}(\bar{t})$ at each decision epoch $\bar{t}$.

5.2 Negotiation Protocol

The coordination of multi-UAV mission planning is a distributed negotiation protocol since the coordination is totally distributed (i.e., the control is decentralized). At each decision epoch, the protocol shall process a Nash equilibrium of the game $\bar{\gamma}(\bar{t})$ a distributed fashion.

One possible approach is to treat this problem as a **Distributed Constraint Optimization Problem** (DCOP). A DCOP is a formalism wherein distributed agents, each with control of some variables, have to optimize a global objective function characterized as the aggregation of distributed constraint utility functions, i.e., the utilities for satisfying or violating the constraints [10]. In addition, a DCOP game is a formulation that explicitly models the strategic dependencies between the decision variables that are controlled by each agent in a DCOP [11]. It is shown that a DCOP game is potential. Thus, a solution of a DCOP is a Nash equilibrium of the related DCOP [11]. As a consequence, an algorithm that is able to solve a DCOP may be used to compute a Nash equilibrium of a generic potential game and is able to represent a negotiation protocol for such a game.

Within this work, we have used the Distributed Stochastic Algorithm (DSA). It is not a complete algorithm: it does not always find an optimal solution, but it finds a sub-optimal solution of a DCOP. Nevertheless, although DSA does not guarantee the optimality of solutions that it provides, it is generally suitable to find good approximate solutions in distributed real-time environments, even if the objectives frequently change over time. Indeed, the control flow of DSA is really simple. After an initial phase in which the agents select random values for their decision variables, a negotiation loop starts. In each iteration, every agent i sends its current state information (e.g., its current selection of its decision variables $\mathbf{d}_i$) to its neighbouring agents if it changed its value in the previous iteration and similarly receives the state information $\mathbf{d}_j$ from the

neighbours. Then, it stochastically decides to keep its current value or change to a new one. The objective of this change is typically to reduce the possible violation of constraints, by simultaneously maximizing the individual utility of the agent. Instead, no computation of the global utility is required: this function is totally ignored by the agents.

A unique feature of DSA is to adopt randomness to select the next decision value. Depending on if it is able to select a new value that reduces the conflicts, the agent changes to the new value according to a stochastic scheme. Since each agent changes its variables with $p \in [0,1]$ probability, the likelihood of two neighbours changing at the same time is p^2. As long as p is conveniently selected in the given domain of the problem, the protocol iteratively climbs to a better state. The p parameter is also named degree of parallel executions. Moreover, if the DCOP is a potential game (just like in the case of the lookahead stage game for the coordination mechanism of multi-UAV mission planning), no agent will leave a Nash equilibrium after the first time it is played and the probability that the agents play a Nash equilibrium goes to 1 as time progresses.

procedure MISSION_PLANNING_COORDINATION($\mathbf{x}_i(t), t, t_w, \mathbb{T}, \pi_{i,\bar{\gamma}(t-1)}, M, N_{\text{neg}}$)
  $\pi_{i,\bar{\gamma}(t)} \leftarrow \text{commitment\_filter}\left(\pi_{i,\bar{\gamma}(t-1)}\right)$ ▷ commitment filtering of last outcome
  for $k \leftarrow 1, N_{\text{neg}}$ **do** ▷ negotiation loop
    send $\left(\pi_{i,\bar{\gamma}(t)}, j\right)$ ▷ send proposal to other vehicle agents
    $\pi_{j,\bar{\gamma}(t)} \leftarrow \text{receive}(j)$ ▷ collect proposals from other vehicle agents
    $\mathbb{T}_{\text{conflict-free}} \leftarrow \mathbb{T}$ ▷ initialization of reachable targets set
    $\text{C} \leftarrow$ **false**
    for all $j \in (1, \ldots, N_v), j \neq i$ **do** ▷ checking loop for strategy collisions
      $(\text{C}_j, \mathbb{T}_{\text{conflict-free}}) \leftarrow \text{check\_strategic\_collisions}\left(\pi_{i,\bar{\gamma}(t)}, \pi_{j,\bar{\gamma}(t)}, \mathbb{T}_{\text{conflict-free}}\right)$
      $\text{C} \leftarrow \text{C}$ **and** C_j
    end for
    $r \leftarrow \text{random}(0,1)$ ▷ generation of a random number in $(0,1)$
    if (C **or** $r < p$) **then**
      $\pi_{i,\bar{\gamma}(t)} \leftarrow \text{Micro-Level\_Mission\_Planning}\left(\mathbf{x}_i(t), t, t_w, \mathbb{T}_{\text{conflict-free}}, M\right)$
    end if
  end for
  return $\pi_{i,\bar{\gamma}(t)}$
end procedure

Figure 5-1: Negotiation protocol of the game $\bar{\gamma}(t)$, executed by the i-th vehicle agent

Figure 5-1 reports the design of the coordination algorithm for the MAS-based multi-UAV mission planning. It shall be iteratively invoked at each decision step of the control cycle of the vehicle agent. It performs the negotiation in a totally distributed fashion and is a customization of the DSA algorithm for our application. Firstly, in contrast with the traditional DSA, each vehicle agent does not start from a random value of the decision variable $\pi_{i,\bar{\gamma}(t)}$, but it uses the outcome of the lookahead stage game $\pi_{i,\bar{\gamma}(t-1)}$ at the previous decision epoch as an initial condition for the current solution. This choice is really useful since it is expected to significantly speed the convergence of the protocol. Besides, the initial condition $\pi_{i,\bar{\gamma}(t-1)}$ is filtered by checking a commitment constraint, which states that if a vehicle is executing a surveillance task of a target $\mathbf{t}_k$ (i.e., it is moving towards $\mathbf{t}_k$ or it is observing it), it shall not be able to abort it. Thus, the **commitment_filter** function partially builds the planned strategy by "blocking" the initial committed portion. The negotiation loop shall plan only the residual negotiable portion of the individual strategy.

At each iteration of the negotiation loop, the vehicle agents exchange their own proposals for the windowed strategy. Then, every vehicle agent performs a check of possible strategic collisions amongst its strategy proposal and the strategy proposals of the others. This check provides a flag that signals the presence of collisions and the set $\mathbb{T}_{\text{conflict-free}}$ of reachable targets that have to be considered for the planning of the next

strategy proposal. In particular, this set is computed in a way that: only the targets that may be reached within the current planning window have to be considered; if two vehicles strategically collide on a conflicting target, this shall be kept by the vehicle agent that reaches the target in the shortest time by means of its strategy proposal. The former criterion requires sufficiently large values of the length t_w of the planning window.

If a collision is detected or if the generated random number $r \in (0,1)$ is smaller than the parallel degree p, the negotiation process of vehicle i calls for the process of single-UAV mission planning. This may be any graph-based routing algorithm for a single vehicle. As regards the termination condition of the negotiation loop, a fixed number of iterations N_{neg} is employed. This choice is mandatory in order to equip the protocol with a deterministic behaviour from a temporal point of view, which complies with timeliness property.

5.3 Simulation Results

The distributed negotiation protocol has been implemented in NetLogo (http://ccl.northwestern.edu/netlogo/). It is a multiagent-programmable modelling environment, that is particularly well-suited for modelling complex systems developing over time. Figure 5-2 shows the graphical user interface of the NetLogo implementation of the multi-UAV mission planner.

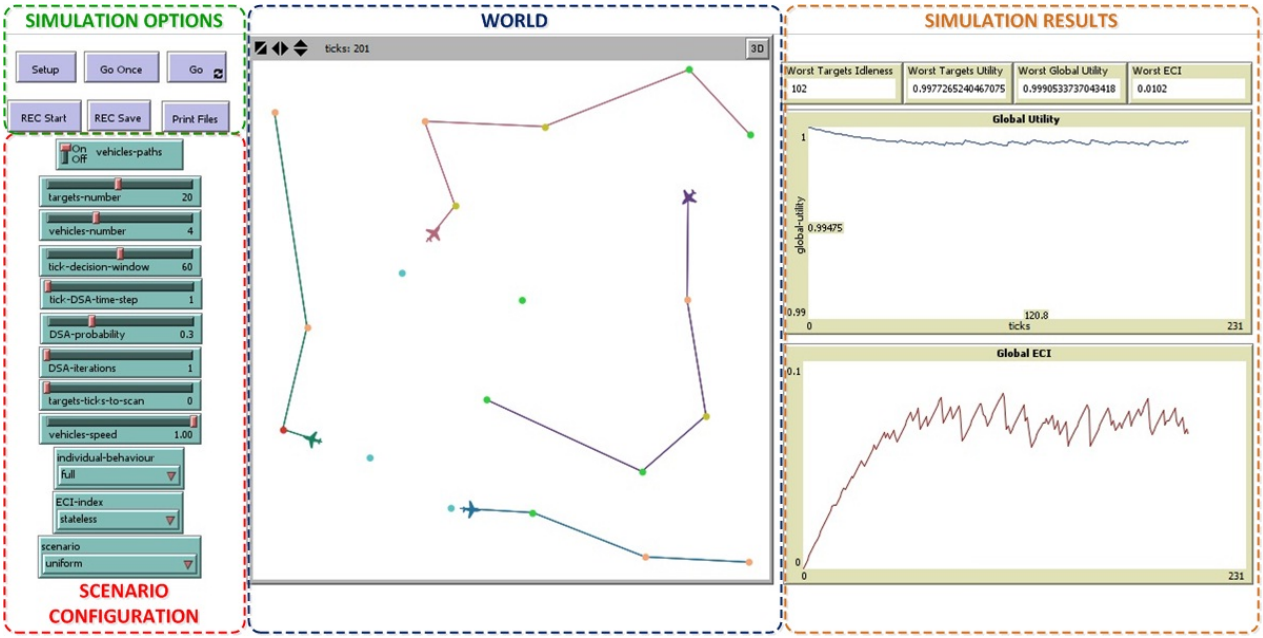

Figure 5-2: NetLogo interface of the multi-UAV mission planner for persistent surveillance.

The following metrics have been measured for each simulation run:

- t_{exp} – the exploration time, i.e., the number of time steps to visit once all targets;
- $idle_{\text{worst}}$ – the worst idleness, i.e., the maximum number of time steps that a target has remained unvisited;
- $\overline{\text{ECI}}$ – the sample mean of the ECI index;
- $\text{ECI}_{\text{worst}}$ – the worst value of the ECI of a target;
- $\bar{\mu}_T$ – the sample mean of the average target utility;

- $\mu_{T_{worst}}$ – the worst value of average target utility.

$\mathbf{ECI}_{worst}$, $\mu_{T_{worst}}$, t_{exp} and $idle_{worst}$ are provided only for completeness. They are not included in the objective of the optimization problem for mission planning, therefore, their values are not explicitly indicative for the performance, in contrast with the sample means $\overline{\mathbf{ECI}}$ and $\bar{\mu}_T$.

In order to provide a measure of the numerical quality attained by means of the distributed negotiation protocol, a comparison has been made with respect to a centralized mission planner. This comparison is useful since the negotiation protocol may generically find a local optimum of problem (7). The deviation of such a local optimum from the global optimum represents the loss of quality of our game-theoretic decentralized solution with respect to a centralized solution. In detail, a central best-first algorithm has been implemented in NetLogo to solve the multi-UAV mission planning problem. Such an algorithm accesses the total environment state at every time step and centrally deliberates the mission plans of all the vehicles. In order to attain a consistent measure, the comparison has been made between the centralized best-first algorithm and the distributed negotiation protocol with a best-first individual planning of the vehicles.

The adopted simulation scenario is a uniform one, wherein all targets have the same the probability density $P_E(t)$. This is a constant P_E and is equal to 10^{-4} per time step. In addition, the following conditions have been employed for the simulation scenario:

- the number of targets is equal to 20;
- the targets positions are randomly generated for each run and a uniform random distribution $\mathcal{U}(0,40)$ is used for both coordinates;
- the number of vehicles is equal to 4;
- the DSA degree of parallel executions p is equal to 0.3;
- the number of iterations of the negotiation loop is equal to 3;
- the cost $C_E(t_1 - t)$ of the event E is unitary;
- the speed of the vehicles is equal to 1 meter per time step;
- the length of the decision window is equal to 60 time steps;
- constraints about fuel consumption, threats and no-fly zones are not considered;
- the discount factor β of target utility is equal to 0.8;
- the number t_{OBS} of time steps to inspect a target is equal to 1.

Table 5-1 reports the estimated values for the simulations results of the comparison by means of 150 simulation runs. The values of the utility functions are not reported here since they are not considered by the centralized algorithm, which exclusively takes into account the $\mathbf{ECI}$ function.

Table 5-1: Simulation performances of the centralized best-first and the decentralized best-first algorithms for multi-UAV mission planning

Algorithm	t_{exp} [steps]	$idle_{worst}$ [steps]	$\overline{\mathbf{ECI}}$	$\mathbf{ECI}_{worst}$
Centralized	94	138	0.0923	0.0138
Decentralized	86	113	0.0944	0.0113

Starting from these simulations, an analysis of the efficiency of the distributed negotiation protocol may be

performed. Such an efficiency has to be interpreted from a numerical point of view, i.e., it has to quantify the loss of quality of the decentralized strategy with respect to the quality of the optimal strategy that is planned by the centralized mission planner. The instantanous efficiency of the designed protocol is defined as

$$E_{\text{neg}}(t) = \frac{\text{ECI}_{W,T}^{(\mathcal{M}^*_{\text{cen}},t)}}{\text{ECI}_{W,T}^{(\mathcal{M}^*_{\text{neg}},t)}}, \tag{13}$$

wherein $\mathcal{M}^*_{\text{neg}}$ is optimal mission plan of the distributed negotiation protocol, whereas $\mathcal{M}^*_{\text{cen}}$ is the correspondent optimal mission plan of the centralized algorithm. We also denote with $\overline{E_{\text{neg}}(t)}$ the sample mean of the instantaneous efficiency and with $E_{\text{neg}_{\text{min}}}(t)$ the mimimum value of the instataneous efficiency. In detail, the following results have been obtained for the efficiency analysis:

- $\overline{E_{\text{neg}}(t)} = 0.9896$;
- $E_{\text{neg}_{\text{min}}}(t) = 0.6063$.

Thus, in the adopted simulation scenario, the efficiency is on average 98.96%, with a minimum peak of 60.63%.

6.0 CONCLUSION

The high-level objective of this work has been to develop a design-for-dependability solution of a critical autonomous MAS. The case study has been represented by the design of a dynamic and decentralized mission planner for a fleet of autonomous UAVs, that shall accomplish a persistent surveillance mission in a cooperative fashion.

We have proposed a systematic methodology, which combines the BDI paradigm and the formal verification of timed automata. The related time-based model may be used for a scheduling of the agents behaviours in a real-time context.

The coordination mechanism has been designed by means of a distributed negotiation protocol, which relies on the framework of Markov Decision Processes and Markov games. A formal model of the coordination game has been built in order to assure the optimal agreement of the involved agents in the mission planning. The Distributed Stochastic Algorithm has been customized to implement the protocol, which has been implemented in a multi-agent programmable modelling environment in order to perform simulation and efficiency analyses.

The transition of the proposed system to real world requires: the integration of other functional and non-functional properties in the design-for-dependability methodology; the implementation of the proposed design on a target embedded platform, with a specific real-time scheduler; the extension of the distributed negotiation protocol to detect possible non-nominal situations (e.g., vehicle failures, V2V network failure, etc.) and to autonomously recover from them.

7.0 REFERENCES

[1] Chandler, P. and M. Pachter (2009). “UAV Cooperative Decision and Control: Challenges and Practical Approaches”. In: ed. by T. Shima and S. Rasmussen. Philadelphia, Pennsylvania, USA: Society for Industrial and Applied Mathematics. Chap. 2, pp. 15–35.

[2] Nigam, N. (2014). “The Multiple Unmanned Air Vehicle Persistent Surveillance Problem: A Review”.

In: *Machines* 2.1, pp. 13–72.

[3] Freed, M., R. Harris, and M. Shafto (2004). "Comparing methods for UAV-based autonomous surveillance". In: *Proceedings of 2004 National Conference on Artificial Intelligence*.

[4] Bordini, R. H., J. F. Hübner, and M. J. Wooldridge (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology. Chichester, England: John Wiley & Sons Ltd.

[5] Behrmann, G., A. David, and K. G. Larsen (2006). *A Tutorial on UPPAAL 4.0*. Department of Computer Science, Aalborg University. Aalborg, Denmark.

[6] Littman, M. L. (1994). "Markov games as a framework for multi-agent reinforcement learning". In: *Proceedings of the 11th International Conference on Machine Learning (ML–94)*. New Brunswick, New Jersey, USA: Morgan Kaufmann, pp. 157–163.

[7] Chapman, A. C. (2009). "Control of Large Distributed Systems using Games with Pure Strategy Nash Equilibria". PhD thesis. School of Electronics and Computer Science, University of Southampton.

[8] Wolpert, D. and K. Tumer (1999). "An overview of collective intelligence". In: *Handbook of Agent Technology*. Ed. by J. M. Bradshaw. Cambridge, Massachussets, USA: AAAI Press/MIT Press.

[9] Monderer, D. and L. S. Shapley (1996). "Potential Games". In: *Games and Economic Behavior* 14, pp. 124–143.

[10] Maheswaran, R. T., J. P. Pearce, and M. Tambe (2004). "Distributed Algorithms for DCOP: A Graphical-Game-Based Approach". In: *Proceedings of Distributed and Parallel Computing Systems (PDCS)*, pp. 432–439.

[11] Chapman, A. C., A. Rogers, and N. R. Jennings (2011). "Benchmarking Hybrid Algorithms for Distributed Constraint Optimisation Games". In: *Autonomous Agents and Multi-Agent Systems* 22.3, pp. 385–414.