Management of Several Unmanned Aerial Vehicles Towards Several Objectives

Michel Barat, Raphael Cuisinier, Franck Dietrich, Jean-Loup Farges, Guillaume Infantes, Alain Michel, Stephanie Prudhomme and Pascal Taillandier

ONERA
29 avenue de la division Leclerc
92322 CHATILLON Cedex

Abstract

The work presented in this paper proposes a dynamical hierarchical architecture for controlling a system which includes several robots and several objects to be detected, identified, localized and treated during the course of the mission. The architecture presents three levels: The upper level is devoted to the control of areas, the intermediate level to the control of objects and the lower lever to the control of robots. Modules of all levels include planning and execution control sub-modules. Specific issues arising during the development of the architecture are presented: database management, observation management, situation tracking, planning, execution and reaction to disruptive events and communication management. Finally, the first validation step foreseen for the architecture will use a simulation platform. The framework, actors and metrics of the simulation are presented.

Keywords

Control ; Architecture ; Robot ; Multi-robot ; Simulation

1 INTRODUCTION

ONERA is developing an Orientation and Decision Airborne System (ODAS) with the objective of increasing decisional autonomy of Unmanned Aerial Vehicles (UAV). ODAS of different UAV have to cooperate through a communication system in order to be able to decide and to implement actions that fulfil the goals of several missions. The control system resulting from the composition of several ODAS is a multi-robot control architecture.

Work is led in the frame of an internal ONERA federative research project called Autonomy of Offensive Airborne Missions (AMAO), which involves specialists from various domains: sensors, system, database management, simulation, perception algorithms and of course decision algorithms. The context of operations is set to an asymmetrical conflict. Several UAV are supposed to perform detection, reconnaissance, identification as well as strike tasks. The missions considered of most interest belong to Close Air Support (CAS) and Suppression of Enemy Air Defence (SEAD) class. This context as defined here implies constraints on time or accuracy. It conditions also the choices made for the elaboration of final ODAS architecture.

Previous works have addressed the issue of defining an architecture for the control of several robots. Those works bring solutions in terms of execution control and planning. In [1] a decentralized architecture allows robots to receive new goals during the mission. Robots exchange their plans and when a robot receives a new goal it increments its plan in order to achieve it. This new plan is compatible with the activities of its current plan and with the resource usage of the activities of the plans of other robots. The approach is demonstrated in an encumbered environment, where the edges of a navigation graph are resources to be shared

among robots, by the absence of collision and deadlock. Alternatively, the anti-collision is not handled by planning but managed locally in a reactive way, for instance by stopping the robot with lower priority [2]. In this work the assignment of tasks to robots is performed by a central module which make requests to the robots to compute the cost of a path to a task destination or of a visit of several task locations. The answers to the requests are used by the central module for optimizing the assignment. However the task assignment is not necessarily performed thought atomic planning requests. For instance, it can be performed using rules on the state of the robots such as assigning the task to the nearest robot with the capability to perform it [3]. At the opposite, each robot can compute the travel times of several paths, each path allowing to perform a sequence of tasks and a central module selects for each robot one of proposed path and sets travel times in order to synchronize arrivals on locations of linked tasks [4]. Synchronization can also be performed after assignment by generating, for each robot, a set of good path to the location where it has to perform the task and then to find among those sets the best feasible instant for task achievement [3]. The models of robot navigation used in the assignment and synchronization process are generally rough, leading each robot to a final step of trajectory optimization under constraints on successive locations and times.

The traditional approach for developing control architectures [5] distinguish multilayer control and multilevel control. The multilayer control consists in decomposing the problem into simpler problems. For instance the multi-robot control problem can be decomposed in collision avoidance, task assignment, synchronization and trajectory optimization sub-problems. A control function is then associated to each sub-problem. Higher layer control functions then serve to integrate the individual sub-problems, so that the overall problem is adequately handled. The multilevel control consists in decomposing the controlled system in sub-systems and in associating to each sub-system a controller. Then in order to obtain consistent behaviours for the controllers, upper level controllers are devoted to the coordination of immediate lower level controllers leading to a hierarchical structure. It should be noted that an architecture integrates generally both aspects. Thus, each subsystem controller of the multilevel hierarchy might itself be realized in terms of a multilayer structure. The distinction between multilayer and multilevel architectures relies mainly on the type of the first decomposition made for solving the control problem.

The work presented here is an attempt to solve the multi-robot control problem using a multilevel approach. The main problem for applying this approach to a set of mobile robots acting in an open environment is that the controlled system is not known *a priori* and is discovered by robots in the course of the mission. Thus, the work presented here proposes a dynamical hierarchical architecture for multi-robot systems in a partially unknown environment. Another challenging specificity of the application of the multilevel approach to multi-robot and environment control is that all subsystems belonging to the environment have no associated sensor and no associated actuators.

In the next section of this paper, the main hypotheses about the controlled system are given. The section 3 presents the derivation of the architecture from the hypothesis made about the system. The issues of database management, observation management, situation tracking, planning, execution and reaction to disruptive events are addressed in the scope of the architecture. This is the topic of section 4. Such an architecture has to be demonstrated by simulation of scenarios. The development of the simulation platform is presented in section 5. Finally some conclusions are given.

2 HYPOTHESES ABOUT THE CONTROLLED SYSTEM

2.1 Robots

The process to be controlled includes a set of mobile robots. Those robots are equipped with sensors and actuators, allowing them to perceive and act on their immediate environment. More precisely, the nature of perception and possible action of a robot depends on the location and attitude of the robot. Moreover, some sensors are not permanently active and have to be controlled. For instance sensors can be controlled using the following actions:

- Heating sensor,
- Turning on or off sensor,
- Changing the mode of sensing and
- Pointing sensor in a given direction.

The rough information given by some sensors has to be processed to deduce relevant features of the environment. Examples of this kind of processing are:

- Image processing,
- Stereo vision by motion and
- Processing of Synthetic Aperture Radar (SAR) sensor data.

Finally, robots may exchange information with other robots and operators through a communication system.

2.2 Environment

The process to be controlled includes not only robots but also a set of mission relevant Objects of Interest (OI) located in several area of interest of the environment. Those objects belong to a given set of categories. The mission defines the type of treatment that has to be applied to an object in function of its category and eventually of its area. However, the number, the locations and the categories of the objects present in the environment are not initially known by the control architecture.

As shown on figure 1, the evolution model of an object from the point of view of the architecture describes initially the evolution of the knowledge of the architecture about the object and latter the evolution of the object itself. Each arrow of the graph corresponds to actions performed by robots that achieve the transition from one state to the next state.

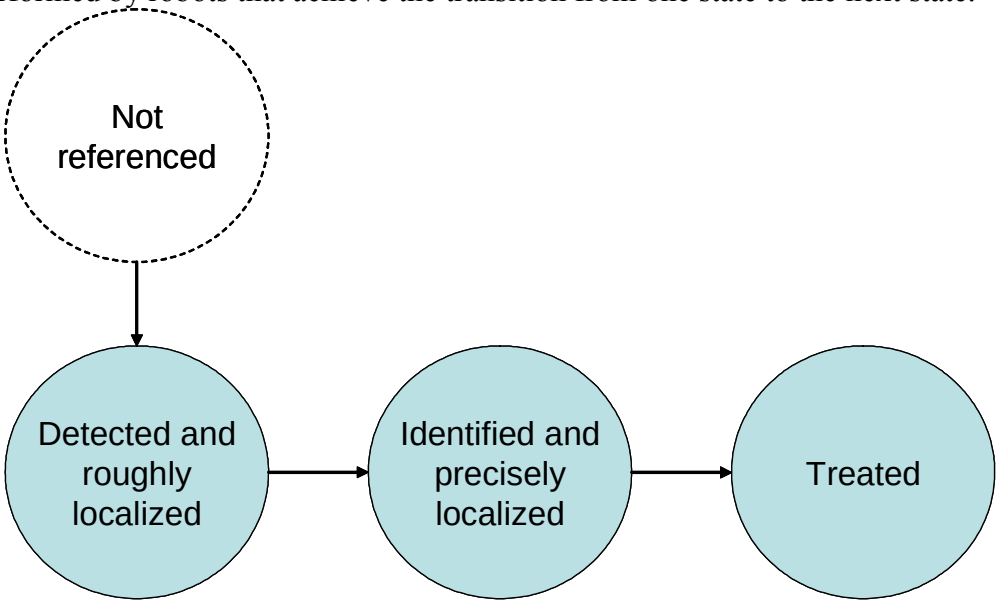

FIG. 1 – Evolution model of an object from the point of view of the architecture.

3 PROPOSED ARCHITECTURE

3.1 Principle

The architecture for controlling the set of robots in their environment is defined following the principles of hierarchical architectures. The system to be controlled is decomposed in subsystems and a control module is associated to each subsystem. The hypothesis made about the system to be controlled leads naturally to define:

- A robot control module for each robot.
- An object control module for each known object.

The fact that the objects are found by the architecture during the course of the mission leads to a dynamical structure where object control modules are created when necessary. Moreover, the object control modules are not directly connected to sensors and actuators. They have to control the object through robot control modules, implying the existence of communication links between each pair of robot control and object control modules. The creation of object control modules is performed by area control modules devoted to the different area of interest for the mission. Area control modules have to control the area on one hand directly through robot control modules and on the other hand indirectly thought object control modules. This implies communication links on one hand between each pair of robot control and area control modules and on the other hand between each area control module and all the object control modules in charge of objects already detected in the area. Figure 2 gives a graphical presentation of the architecture together with the controlled system.

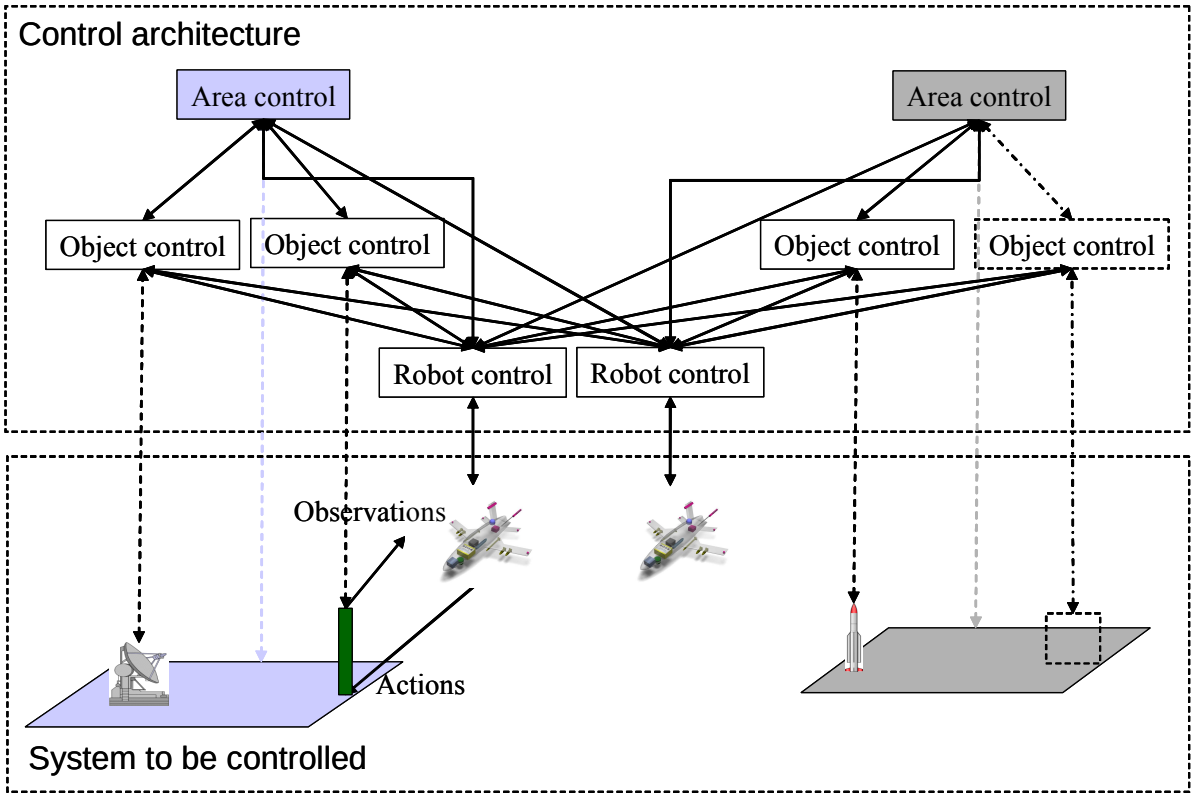

FIG. 2 – The system to be controlled and the proposed control architecture.

The architecture aims at giving to the set of robots an autonomy at the level of objective management. This specification leads to control modules including not only executive and

reactive behaviours but also a planning behaviour. As shown on figure 3, this implies two types of dialogs between modules :

- Task planning requests and answers to those requests and
- Task execution clearance and task execution report.

Besides those short messages, each module of the architecture has to be aware of the parameters of the mission, of state of the controlled system and of the intentions of other modules. This feature is provided by a database with shared elements.

From the hardware point of view, the architecture includes a computer inside each robot. The robot control module of a given robot is naturally resident in the computer of the robot. Area control modules are assigned to robot computers at the mission preparation phase. Object control modules are created in the computer making the processing that allows the corresponding object discovery. This hardware distribution induces a distribution of the database on the different computers with mechanisms to maintain consistency of shared elements.

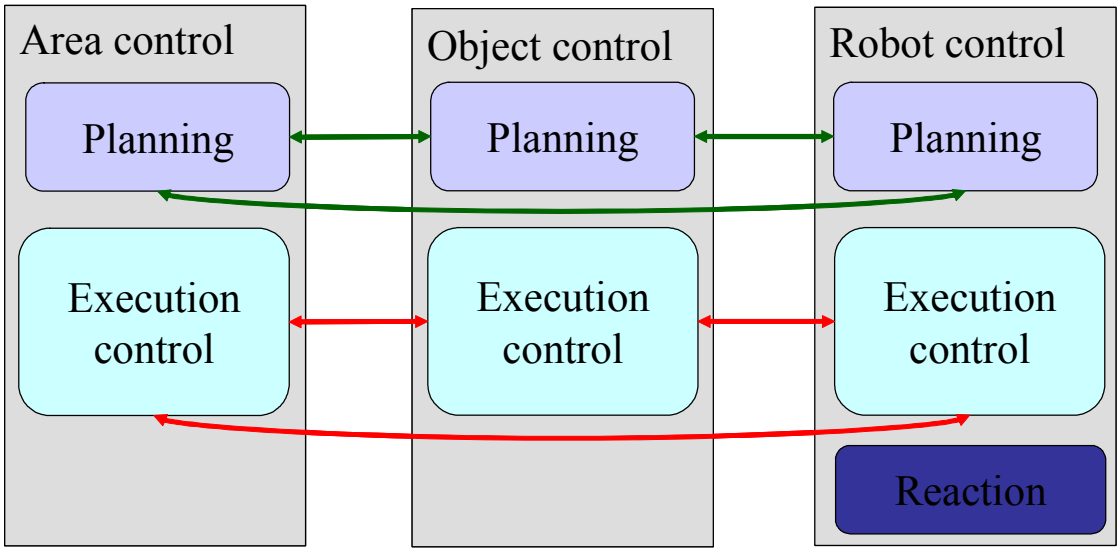

FIG. 3 – Module decomposition and exchange between modules. In green task planning requests and task acceptance or rejection. In red task execution clearance and task execution report.

3.2 Robot control

3.2.1 Planning

The planning sub-module of the robot control module may receive task planning requests from several object control modules and from several area control modules. The fact that some of those requests may be conflicting leads to use an oversubscription planning method; some tasks may not be planned in the produced solution.

The robot control module itself is another source of planning requests. Requests are emitted when:

- execution deviates strongly from current plan or
- current plan computation hypothesis is invalidated by changes in environment knowledge.

In order to be able to cope with different kind of requests possibly arriving while the planning algorithm is executing, the planning sub-module includes not only the algorithm but also a request manager.

Moreover a successful plan computation does not mean systematically an immediate application of this plan. Indeed, when the task planning request for a first robot is linked to a task planning request for another robot which is not able to satisfy it, the task planning request for the first robot may not be valid. In such cases, plan application needs some validation from the sender of the request.

3.2.2 Execution control

The sub-module dedicated to control of execution is dependant of clearances for next task execution. This implies to express in the plan structure the actions the robot should perform when the clearance is late.

3.2.3 Reaction

When the robot is submitted to an unexpected a risk, there are two cases:

1. The risk will be effective at a time far from current time.
2. This risk is currently occurring.

In the first case, the reaction consists in requesting a new plan computation taking into account the new risk. For the second case, specific robot behaviours are associated to each, previously identified, kind of risk. Those behaviours are implemented as mini plan structures that can be temporally substituted to the plan under execution. The deviations from the original plan induced by those behaviours may also imply a new plan computation request.

3.3 OI control

3.3.1 Planning

The goal of the object manager plan is to obtain enough information about the object and then to obtain effects on the object in conformance with procedures and object identification. The solution plan is composed of tasks to be submitted to the robots. This leads to the following features:

- The planning process includes a robot assignment process.
- The final feasibility of a plan found by the object manager is function of task acceptance by robot planning.
- There are at least two planning phases: before and after object identification.

In order to avoid a blind search the object manager shall have a simplified model of task acceptance by robots.

The commitment of a robot to perform a task in a time window cannot be unconditional and full because:

1. robots are submitted to disruptive events and
2. there are other object managers competing for the usage of the robots.

This partial and conditional commitment leads to implement a re-planning capability for the object manager planner.

3.3.2 Execution control

The main purpose of object level control of execution is to ensure coordination between tasks performed by different robots on the same object. This coordination deals with precedence and synchronization relations between task executions. For this purpose the object execution controller gathers task execution reports from the controllers of the robots and deduces the clearances for robots awaiting for next task execution.

Another purpose of the control of execution is to produce a planning request when the object identification and localization is completed.

3.4 Area control

3.4.1 Planning

The goal of the area manager plan is in a first phase to explore the area in order to detect objects. In a second phase its goal is to synchronize the treatments of linked objects. Those two phases may overlap. Similarly to the object manager, the area manager planning includes assignment of robots to tasks, but those tasks are exploration tasks. The feasibility of plans depends not only on acceptance of exploration tasks by robot managers but also on acceptance of time windows by object managers. Finally, the area manager requests to object managers are not competing with requests from other area managers because objects belong to only one area.

3.4.2 Execution control

All robot clearance and execution report features of object manager execution control apply to area manager execution control. Moreover, the area manager have to take under control the object managers when they are created. The coordination between object managers performed by an area manager is similar to the coordination between robot managers performed by an object manager.

4 SPECIFIC ISSUES

4.1 Database management

The aim of the ODAS database system is:

- To store and to manage the slow-growing or perennial data about environment, mission, equipment and capabilities of platforms, planning, execution control and tactical situation (observation, perception, decision).
- To contribute to the cooperation between ODAS systems by distributing and synchronizing the respective databases via an automatic mechanism (Figure 4) for exchanging objects with a generic system of communication named SysCom. This mechanism is based on a partitioning of objects in two classes: shareable object and not shareable object.

The database covers six main areas of ODAS and is based on standard NATO JC3IEDM [6].

- Environment: Weather maps, terrain maps, etc.
- Mission: Defining missions, areas of operation, etc.
- Equipment knowledge: Knowledge of the onboard equipment and capabilities of the robot
- Behaviour specification:
  - Rules for creating a mission (tasks, actions, tactics, procedures, instructions).
  - Behaviour consequence : the impact of the current plans of the robots on future situation.
  - Meta plan : the plans of area and object controllers.
- Perception: Information on observation, estimation and decision about object types.
- Tactical picture: Knowledge of the tactical situation, objects of interest, area of interest, etc. Who does what, where, when, how and why.

The database was build using a derivation of a database conceptual model into an implementation model. The automatic generation process produces C++ object classes and xsd schema. Then using a form generator, like InfoPath, and the xsd schema, the initial content of ODAS database coded as an xml file is produced. A library code, automatically generated, allows to create, update and delete objects of the xml file representing the database.

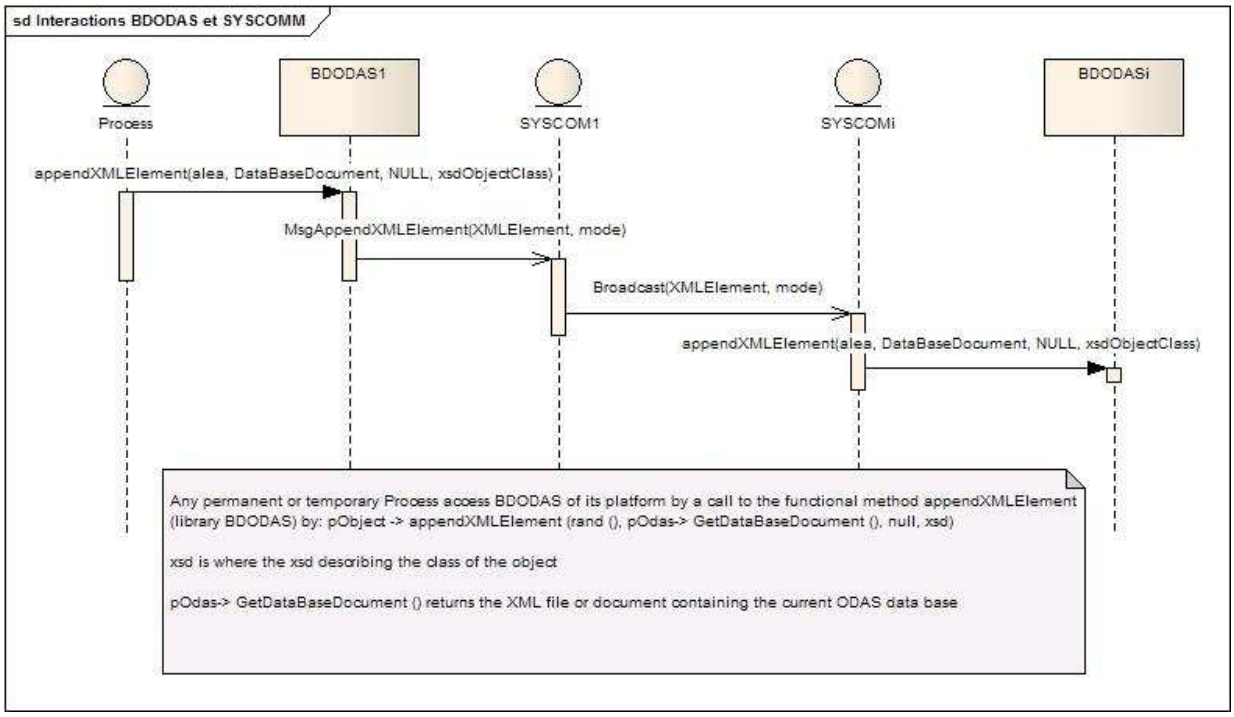

FIG. 4 – Sequence diagram for database synchronization.

4.2 Observation management

Following the result of an acquisition the perception system has to perform the following tasks:

- detection,
- recognition,
- identification,
- fusion,
- decision about object classes made
  - automatically or
  - by an human operator.

The aim of the perception package is to increase the autonomy of the system by proposing fully automatic functionalities for images understanding at sensors output. Figure 5 presents a simple example of control of a detection process by the execution control of a robot after a SAR acquisition.

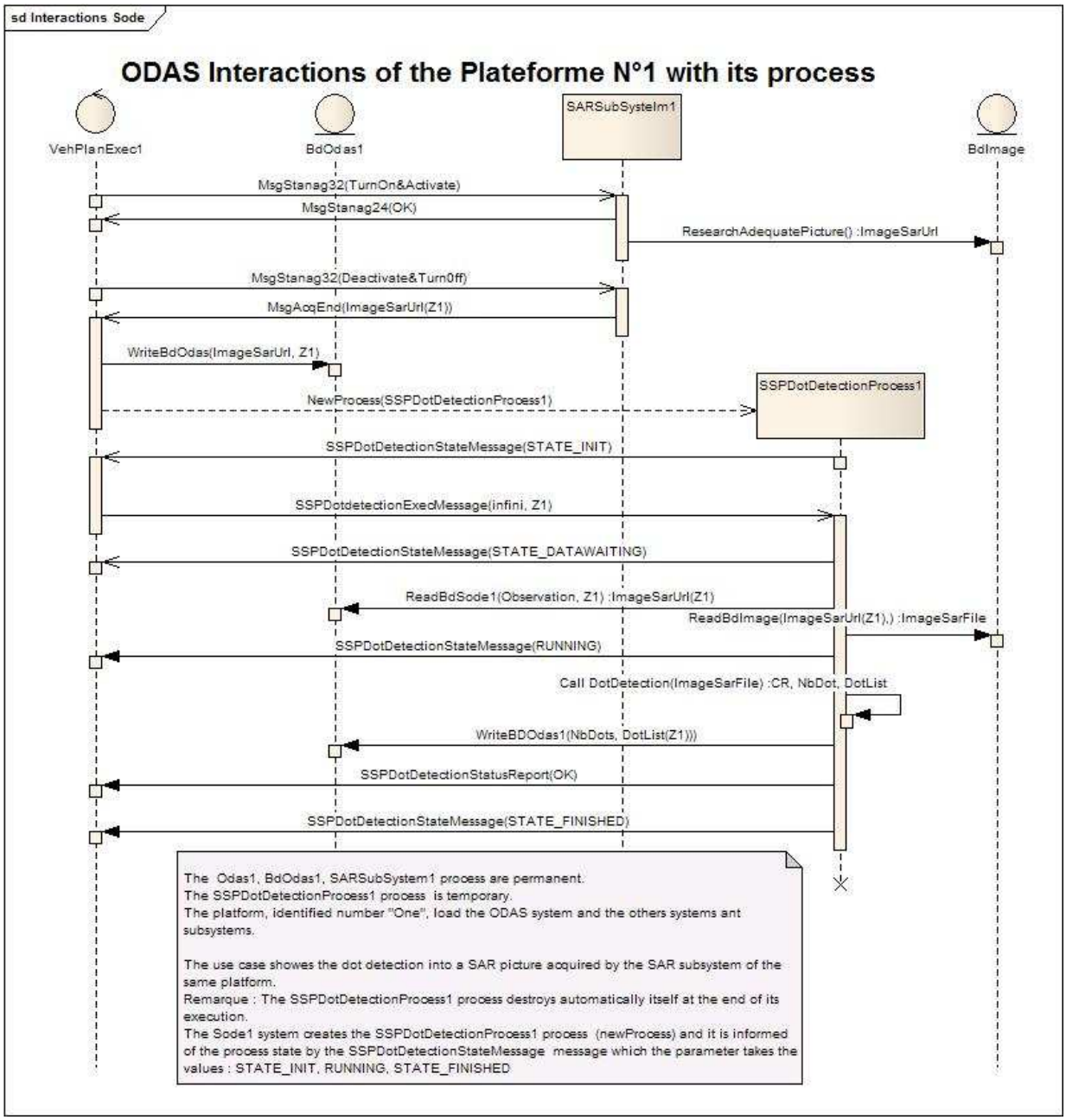

FIG. 5 – Sequence diagram for acquisition and detection.

The intervention of an operator is envisaged, through a graphical interface, only to validate the strike or whenever the algorithms would take uncertain decisions. The implemented functionalities are "automatic target detection" resulting from "fusion of the detections in the 3 SAR images" (Figure 6) and Automatic Target Recognition (ATR) from optronic images (Figure 7).

Thanks to its stand-off acquisition, its wide field, and its all weather capacities, the SAR imagery is particularly well adapted to detecting metallic objects in a natural environment. The method of detection is based mainly on the technique of clutter suppression [7], a target being regarded as an anomaly compared to its close environment.

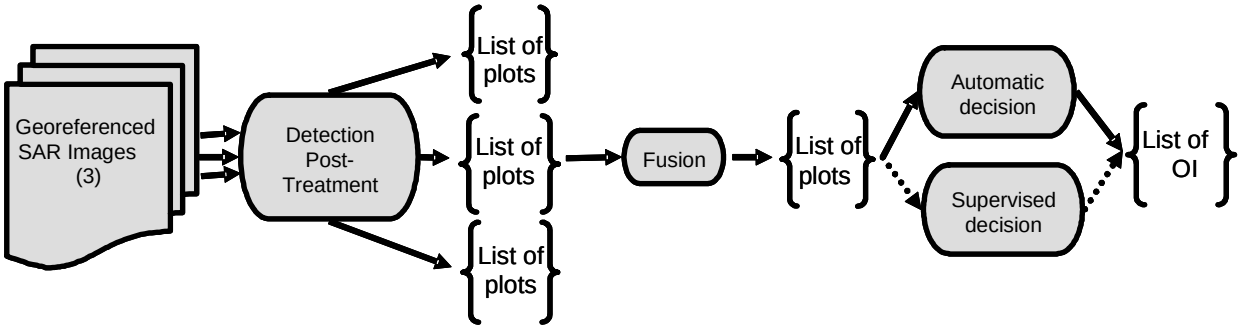

FIG. 6 – Block diagram of baseline SAR detection system.

In order to limit the quantity of information coming through the data links, the result of the detection of each SAR image is transmitted as a list of plots located by their geographical positions. The fusion of detections is then carried out in a decentralized scheme; producing a list of OI characterized by a confidence index providing the plausibility of their existence.

The recognition of the vehicles is carried out on the basis of triplets of pairs of visible and infra-red high-resolution images. As the current performance of the identification process with SAR images is not sufficient to consider automation. Each triplet consists of an image acquired at nadir and two images acquired using an oblique optical axis at ±45°.

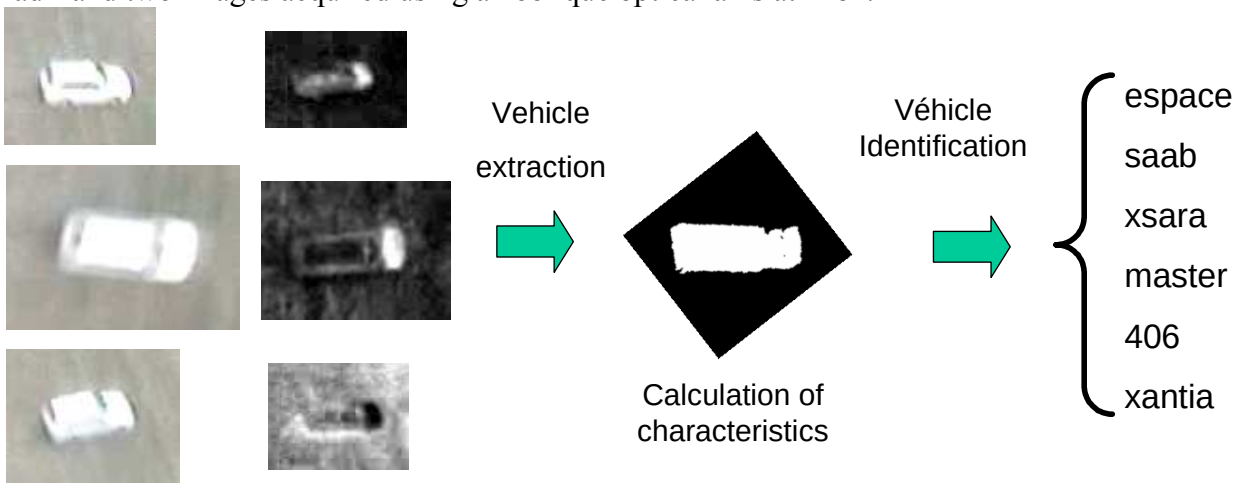

FIG. 7 – Block diagram of baseline ATR system.

At the end of the identification process the "target" is attributed by an identity associated with a confidence index (plausibility). If the value of this index is insufficient, triplets are presented to the human operator who confirms or invalidates the proposed identity.

4.3 Situation tracking

The tactical situation contains information about OI or areas of interest resulting from processing chains of the perception system. The uncertainty information concerns not only the location, the shape geometry of OI, but also of the information produced by the recognition process such as class, function and identity of OI. The situation is build by the ODAS system and can be propagated to other systems by synchronizing the databases.

The objects of the tactical situation are discovered during collection, fusion and decision treatments and enrich the database. In the example shown on figure 5, a detection process add new information to the database.

4.4 Planning

The main issues about planning in this architecture are convergence and optimality. Convergence problems may arise when two OI controllers are expecting answers to their requests sent to two robot controllers. The first robot controller may be waiting for plan validation by the first OI controller, while the second robot controller is waiting for plan validation by the second OI controller. In such case, if the first OI controller has sent a request to the second robot controller and reciprocally the second OI controller has sent a request to the first robot controller, there is a deadlock. The solution chosen to avoid such a situation is based on:

- Interruption of planning and validation processes by request with more priority.
- Definition of a strict priority order on planning requests.

The strict priority order is based on one hand on the type and context of OI or area associated to the request and on the other hand on the identities of the process and node of the architecture emitting the request.

Global optimality is not provided by the architecture because it solves a problem that is not formalized. A local optimality can be ensured at robot level by ensuring that a requested task cannot be rejected by choosing instead a set of tasks with less priority. It is possible to use an optimization algorithm such as A* with the following criterion:

$$J = \sum_{i=0}^{i=N} 2^{-i} T_i$$

J is the criterion to be minimized, T_i a variable with a value of one if the task i is in the plan and zero otherwise. The index 0 is assigned to the task with the highest priority, the index 1 to the second one.. the index N to the task with the lowest priority. As:

$$\sum_{i=0}^{i=N} 2^{-i} = 1 - 2^{-N} < 1$$

it is ensured that T_0 will never be rejected to choose $T_1,\ldots,T_N$. This property apply recursively to T_k with respect to $T_{k+1},\ldots,T_N$. The local planning problem can also be solved by an heuristic which try to plan first the task with the highest priority and then try to integrate recursively in the current plan tasks with lower priorities. It has be observed experimentally that the sub-optimality of this heuristic is acceptable.

4.5 Execution and reaction to disruptive events

The mechanism of execution clearance of tasks by OI controllers do not presents deadlocks because at a given time a robot may execute only one task for a given OI or area controller among all OI or area controllers. Reaction to a disruptive event on a given UAV may lead to giving up tasks. Then the OI or area controller may have to interrupt associated tasks for other UAV and start a new planning cycle.

4.6 Communication management

A generic model of communication system is proposed. This model processes by exchange messages and can measure the quantity of data transit by the channel and evaluates the transmission delay of each data. The operative characteristics of the model are :

- Exchange of information by message
  - Order and control
  - Summery of execution
  - States components
  - Updates element BDODAS (synchronization).

The assessment parameters of assessment are :

- Mode: Broadcast, point to point.
- Characteristics of a generic channel (connection between two SysCom)

- Range
- Rate of flow
- Latency of transmission
- Opening time of the channel
- Repetition rate

5 SIMULATION

5.1 Framework

The framework relies on generic tools and can be used to simulate different kind of scenarios. We will give an example of such a scenario, especially to demonstrate the complexity of the ODAS operational environment. Then, we will discuss the overall simulation solution.

5.1.1 Example of a scenario

In the proposed scenario (Figure 8), UAVs belonging to the "blue force" should destroy a mobile armed camp belonging to the "red force".

- The blue force consists of UAVs, an AWACS, a refueler, and a C2 system with human operators. All those systems communicate through tactical data links. UAVs are equipped with radar and electro-optical sensors, and air-to-surface missiles.
- The red force consists of SA-7 manpads, SA15 vehicles (an armored vehicle equipped with an acquisition radar, a tracking radar and 8 surface-to-air missiles), ZSU23 (a radar-guided anti-aircraft weapon system with four 23 mm autocannons), shelters and tents.
- At the beginning of the mission simulation, UAVs are given a geographical zone where the enemy should be. The UAVs thus have the required initial data to start simulating.

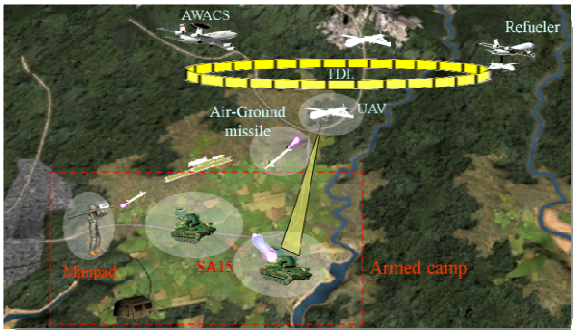

FIG 8: Attack on a mobile armed camp – example of a scenario

5.1.2 Overall solution

The simulator involves a C++ engine, a scenario preparation tool and a 3D visualization software. Those tools can be deployed in different modes. You can gather all models in a single software or you can distribute them across a standard HLA distributed architecture (Figure 9).

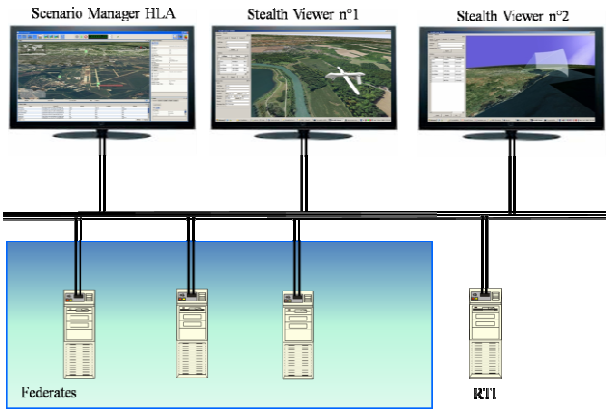

FIG 9 : HLA Federation

The framework components are detailed hereafter.

5.1.2.1 Simulation engine

The C++ simulation engine (Figure 10), which is provided as an Application Programming Interface (API), serves as a backbone across the different steps of the simulation process.

Basically, our simulation entities (platforms, sensors, and weapons) are all considered as actors. Actors exhibit attributes, which have a type, a value, and systematic getter/setter functions. They communicate inside simulation through messages. A manager directs the actors, and is responsible for the whole simulation process (including time management and scenario loading).

The engine architecture is completed by high level components, to be plugged as needed to provide extra functionalities. For example, we have a SimHLA component, responsible for the HLA interface.

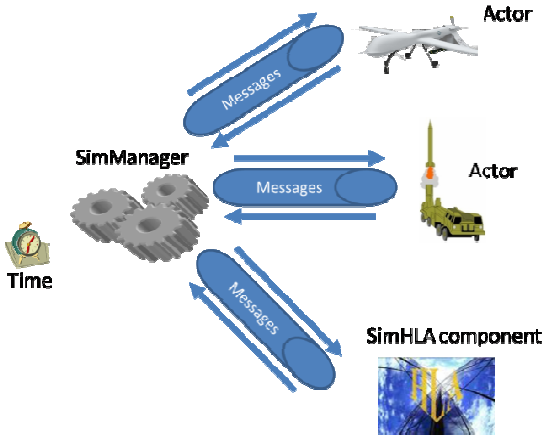

FIG 10 : Simulation engine

The engine also comes with a standard library of actors. In order to enforce RPR-FOM compliance, we have developed some actors which can be inherited from. This is especially useful for platforms: we ensure that actors publish kinematical attributes in a standard way, know how to take damage from different kinds of weapons, understand damage states, can do dead reckoning for both local and remote vehicles, and have the infra-red and radar cross-section attributes useful for sensor models.

5.1.2.2 Simulation tools

Given the complexity of the scenarios, we designed a tool to prepare the different actors (platforms, sensors, weapons) of the simulation, with their technological parameters, their force affiliation and their initial behavior. A GIS helps us to precisely locate the actors in the world. We used World Wind Java SDK (WWJ) as a geographic virtual environment for scenario preparation (Figure 11), tactical situation visualization during simulation execution and replay of trajectory during models development.

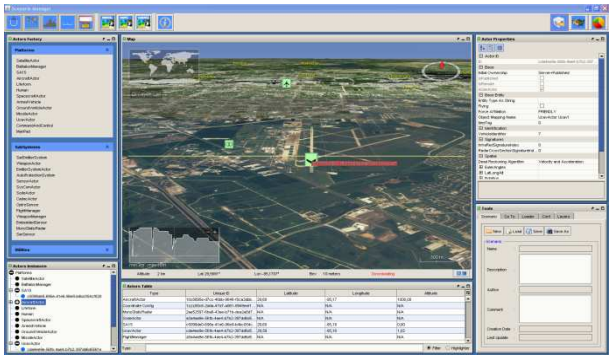

FIG 11: Scenario preparation tool

Another component of the overall framework is the HLA 3D stealth viewer, which provides a 3D visualization of the scene, as seen per the robot.

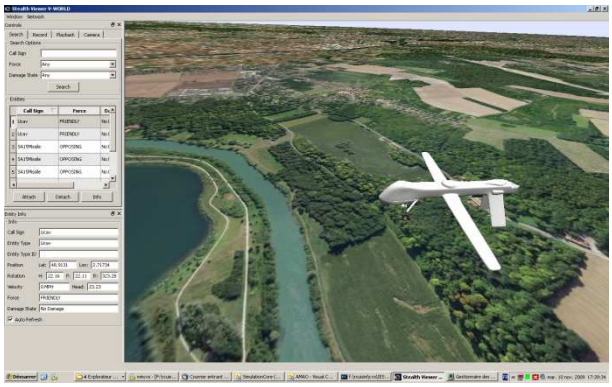

Figure 12: HLA 3D Stealth Viewer

5.2 Actors

5.2.1 Robots

The trajectory of each UAV is simulated by a set of FORTRAN programs based on vehicle aerodynamic data and engine characteristics. This set, named CADAC [8], is encapsulated in an actor of the simulation framework.

Each sensor or actuator sub-system of the robot is also a separate actor whose position and attitude is linked to the position and attitude of the trajectory actor. Actuators sub-systems include weapon and auto-protection sub-systems. Communication is also implemented as a separate actor.

5.2.2 Environment actors

Several environment actors simulate the behaviour of friendly and hostile entities. The hostile entities are:

- Enemy control and command centre.
- Targets.
- Enemy missiles.

The friendly entities are:

- AWACS/JSTAR manned aircrafts.
- Human operator.
- Supply aircraft.

Finally an actor supplies data about terrain configuration and meteorological and atmospheric conditions.

5.2.3 Architecture nodes

Each architecture node corresponding to an individual ODAS embedded in one UAV is implemented as an actor. This actor communicates with the flight management system, sensor systems and weapon systems of its UAV using messages of the framework. Those messages present the same structure than those of the north Atlantic treaty organization standardization agreement 4586 [9]. When exchanging those messages the ODAS actor takes the place of an UAV control system which is usually a ground station.

The ODAS actor manages several permanent and temporary processes. Permanent processes are used for robot control and area control sub modules. Temporary processes are used for OI control sub modules and for perception package functions.

5.3 Metrics

The purpose of the first validation trials will be to validate the technical quality of the functions structuring the architecture. Thus technical metrics are oriented to the technical validation of planning, execution control, perception and communications. In a second step the simulation tests will validate the behaviour of the whole system in operation using mission and operation oriented metrics.

5.3.1 Planning and execution control

The planning function is distributed among area control, OI control and robot control modules. This function is efficient if the tasks and time windows proposed by area control and OI control modules to robot control modules are accepted and executed. From this definition of efficiency several metrics can be defined:

- For each robot, the ratio of the number of requests leading to a new task in a new plan by the number of received requests.
- For each robot, the ratio of the number of requests leading to a new task in a new plan with this task eliminated in a further plan by the number of received requests.
- For each robot, the ratio of the number of requests leading to a new task in a new plan with this task actually executed by the number of received requests.

Those metrics can be aggregated over the set of robots present in the scenario. Symmetrical metrics can be defined for each area or OI controller:

- The ratio of the number of request sent to robot controllers and accepted by them by the number of request sent to robot controllers.
- The ratio of the number of request sent to robot controllers and accepted by them but rejected latter by the number of request sent to robot controllers.

- The ratio of the number of request sent to robot controllers and accepted and executed by them by the number of request sent to robot controllers.

Those metrics can be aggregated over the set of all OI controllers of an area, over the set of all OI controllers and the area manager of an area and over the set of all OI controllers and all area controllers. Finally, some metrics characterize the efficiency of the synchronization of OI controllers by area controllers:

- For each OI controller, the ratio of the number of time windows accepted by the number of time windows requested.
- For each OI controller, the ratio of the number of time windows executed by the number of time windows requested.

Those metrics can be aggregated over all OI controllers of an area to obtain the value for the area controller and over all OI controllers to get an overall value.

5.3.2 Perception

The perception function performs detection and identification. For the detection function the metrics are:

- The ratio of the number of undetected OI at the end of each treatment shown of figure 6 by the number of relevant objects in the input SAR images.
- The ratio of the number of wrongly detected OI at the end of each treatment shown of figure 6 by the number of, wrongly and correctly, detected OI.

Those metrics can be aggregated over several treatments of SAR images.

For the identification the metric is:

- The ratio of the number of correct class assignment by the number ATR function invocations.

5.3.3 Communication

To measure the channel activity and the impact of this activity on the transmission delay.

The metrics of interest are:

- The instantaneous load,
- The minimum load during the simulation or between two events or moments
- The maximum load
- The average load,

For each message type, the minimum delay, maximum and average applied.

During a simulation, it is possible to represent the form of timing diagrams of the different measures: instantaneous or average load, transmission delay, etc.

5.3.4 Mission and operation metrics

Several metrics are scheduled to predict performances at mission and operation levels. They are based on multi-criteria analysis. The key aspects are: resource allocation, time reactivity, survivability, reliability, efficiency, and cost. All these criteria have their own definition which depends on the level of the viewpoint : platform, mission or operation.

6 CONCLUSION

The specification and design of a hierarchical architecture for controlling several robots have been performed. The resulting architecture presents three levels. Usually the control architectures with levels present tree structures. The specificity of the problem of the control of an environment with several robots leads to a more complex structure where lower level modules are not connected to a single module at the immediately higher level. Indeed, robot control modules are connected to all higher level modules. This complex structure leads to

conflicts between planning requests leading to potential deadlocks. The problem of avoiding deadlocks is treated by assigning different priorities to different requests.

However the proof of the correct behaviour of this multi-robot architecture seems difficult to obtain. For this reason, beside the specification and design of the architecture, the specification and design of a simulation tool able to validate it experimentally have been performed. This simulation is based on an actor concept and is able to compute metrics that are valuable for technical and operational assessment.

The on going implementation of the architecture and the simulation will lead to an integration phase. The software resulting from this integration phase will allow the simulation of scenarios that will provide indications about the advantages and disadvantages of hierarchical architectures for multi-robot control.

References

[1] R. Alami, F. Ingrand and S. Qutub *"Planning Coordination and Execution in Multi-robots Environment"*, Proceedings of the 8th International Conference on Advanced Robotics, pp. 525-530, 1997.

[2] B.L. Brumitt and A. Stentz *"GRAMMPS : A Generalized Mission Planner for Multiple Mobile Robots In Unstructured Environments"*, Proceedings of the IEEE International Conference on Robotics and Automation. (3), pp. 2396-2401, 1998.

[3] R.W. Beard, T.W. McLain, M.A. Goodrich and E.P. Anderson *"Coordinated target assignment and intercept for unmanned air vehicles"*, IEEE Transactions on Robotics and Automation, 18(6), pp. 911-922, 2002.

[4] Y. Kuwata *"Real-time Trajectory Design for Unmanned Aerial Vehicles using Receding Horizon Control"*, Master of Science dissertation, Massachusetts Institute of Technology, June 2003.

[5] W. Findeisen and I. Lefkowitz *"Design and applications of multilayer control"*, 4th Congress of the International Federation of Automatic Control, Technical session 42, pp. 3–22, 1969.

[6] NATO Multilateral Interoperability Program *"JC3IEDM-Browse-Representation-DMWG"*, Edition 3.0.2, May 2009, http://www.mip-site.org/publicsite/04-Baseline_3.0/JC3IEDM-Joint_C3_Information_Exchange_Data_Model/HTML-Browser/index.html.

[7] L. Gagnon, H. Oppenheim, P. Valin *"R&D Activities in Airborne SAR Image Processing/Analysis at Lockheed Martin Canada"*, Proceedings SPIE#3491, pp. 998-1003, 1998.

[8] P.H. Zipfel *"Modeling and Simulation of Aerospace Vehicle Dynamics"*, AIAA Education Series, 2000.

[9] NATO Standardization Agency *"Standardisation Agreement – Subject : Standard interfaces for UAV Control System (UCS) for NATO UAV interoperability"*, Edition 2.5, February 2007.