Auton Robot
DOI 10.1007/s10514-016-9603-z

A distributed architecture for supervision of autonomous multi-robot missions

Application to air-sea scenarios

Charles Lesire[1] · **Guillaume Infantes**[1] · **Thibault Gateau**[2] · **Magali Barbier**[1]

Received: 8 December 2014 / Accepted: 23 July 2016
© Springer Science+Business Media New York 2016

Abstract Realizing long-term autonomous missions involving teams of heterogeneous robots is a challenge. It requires mechanisms to make robots react to disturbances or failures that will arise during the mission, while trying to successfully achieve the mission in cooperation. This paper presents HiDDeN, a distributed deliberative architecture that manages the execution of a hierarchical plan. This plan has initially been computed offline, ensuring some military operational constraints of the mission. Each robot's supervisor then executes its own part of the plan, and reacts to failures using a hierarchical repair approach. This hierarchical repair has been designed with the sake of ensuring operational constraints, while reducing the need of communication between robots, as communication may be intermittent or even nonexistent when the robots operate in completely separate environments. HiDDeN's robustness and scalability is evaluated with simulations. Experiments with an autonomous helicopter and an autonomous underwater vehicle have been realized and are presented as the defining point of our contribution.

Keywords Multi-robot cooperation · Autonomous vehicle · Software architecture

✉ Charles Lesire
charles.lesire@onera.fr

Guillaume Infantes
guillaume.infantes@onera.fr

Thibault Gateau
thibault.gateau@isae.fr

Magali Barbier
magali.barbier@onera.fr

[1] ONERA – The French Aerospace Lab, 2, av. Edouard Belin, 31000 Toulouse, France

[2] Institut Supérieur de l'Aéronautique et de l'Espace (ISAE), 10, av. Edouard Belin, 31055 Toulouse, France

1 Introduction

Submarine operations offer a difficult environment for human-beings, and robots have became a natural way to operate in immersion. Of various size, weight, ability and power, they meet both military and civil needs. Communication with marine robots is made difficult by the environment itself, especially if the robots are mobile or not well localized. Increased autonomy abilities for autonomous underwater vehicles (AUVs) allow to tackle a wider range of complex tasks, like search and rescue (Kurowski et al. 2012), pollution surveillance (Mukhopadhyay et al. 2012), infrastructure inspection (Hollinger et al. 2012), collection of oceanographic data (Das et al. 2013), study of wildlife and plants (Pinto et al. 2013b), and so on.

Lots of work are performed on software architectures to increase AUV autonomy so that robots could achieve complex missions while evolving in a dynamic environment and reacting to events. For 10–15 years, new works have been performed on the field of cooperation of unmanned vehicles: flotilla of underwater vehicles (Brinon-Arranz et al. 2014; Xiang et al. 2010; Chatzichristofis et al. 2012; Antonelli et al. 2012) or teams of underwater, aerial and/or surface vehicles (Ferri and Djapic 2013; Shkuri et al. 2012; De Cubber et al. 2012). Multi-robot cooperation has been developed with in mind the added value that a team of robots with complementary skills can achieve more complex and demanding missions, and assist each other to increase the survival time; the drawback being the increased deployment and asset management complexities.

In this paper, we present a distributed deliberative architecture that manages the execution of a plan for a team of heterogeneous and autonomous robots. This work addresses two major constraints that are relevant for long-term missions in operational contexts. First, operational (military)

Published online: 18 August 2016

constraints and procedures are taken into account within a hierarchical representation of tasks. Second, execution and failure management are inherently considered, using a distributed execution scheme as well as a hierarchical repair process adapted to the available communication.

This papers first describes the air-sea scenario in Sect. 2 along with the requirements the scenario implies on the multi-robot architecture. The contribution of this paper is then introduced in Sect. 3: a multilayer distributed deliberative architecture that has been evaluated on simulations and validated on the field. A global view of the architecture is given in Sect. 4, and the underlying models and processes are further detailed in the following sections (Sects. 5, 6). Simulation results are analyzed in Sect. 7, showing the robustness and value of hierarchical repair within a distributed architecture. Section 8 then presents the experiments that have been realized, including some hybrid simulations. Section 9 finally presents some related works, concludes on the operational value of HiDDeN, and discusses some feedbacks learned from experiments.

2 Air-sea securing mission

This section describes the air-sea scenario demonstrated in the ACTION project (see Barbier et al. 2009, for the project challenges) with two experimental platforms: an autonomous aerial vehicle (AAV) operated by ONERA and an autonomous underwater vehicle (AUV) implemented by the Naval French Defense Procurement Agency (DGA TN).

2.1 Military context

The objective of this military scenario (Fig. 1) is to survey a friendly coastal area with an AAV and an AUV in order to detect mines lying on the seabed. This scenario is part of the mine warfare; an application is the securing of an area for the exit/return of a submarine from/to its base. The coastal area is a sensitive area due to boat traffic. The role of vehicles are complementary: the AAV surveys the area to localize boats, whereas the AUV surveys the seabed to localize mines. Due to strict military procedures, the mission area is split into several sub-areas: each sub-area must be secured at the surface before being secured at the seabed.

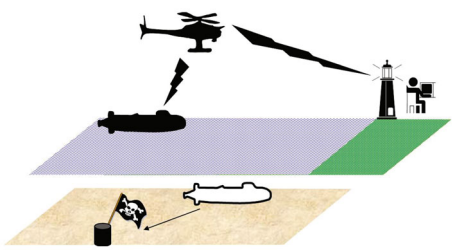

Fig. 1 Securing mission scenario with one AAV and one AUV (the AUV is drawn in white when underwater)

2.2 Operational constraints

Prior to executing the mission, a human operator selects the main area and the sub-areas (according to military procedures, this part has to be done by a military field expert and was not supposed to be automated). The constraints set on the mission by military operators are:

- the initial mission plan must be known by the operator (even if the plan is subject to further changes due to failures) prior to the mission start;
- a sub-area must be scanned by the AAV first, and cleaned of boats, before the AUV can survey the sub-area;
- at the end of each survey, the vehicles must report to the operator; the AAV reports pictures and position of detected boats (the AAV has a camera-based detection and localization function); the AUV reports pictures and position of detected mines (the AUV has a sonar, and associated data processing, to detect and localize mines);
- the AUV never communicates directly with the operator; communications must be relayed by the AAV;
- at the end of each sub-area survey, the operator can stop the mission if they decide that the area is secured enough.

Two main disruptive events have to be managed by the team of robots:

- the AUV detects a mine (in the demonstration, a noisy sensor is simulated) and has to inform the operation center as fast as possible;
- the AUV takes too much time to scan a sub-area, making the AAV wait too much time for the AUV at the end of its own scan.

Furthermore, autonomous management of disruptive events also has to be as predictable as possible in such a military context. Ideally it should consist in parametric procedures with as few parameters as possible.

2.3 Requirements for an autonomous multi-robot mission management system

The constraints raised by the operation of autonomous robots in cooperative air-sea scenarios demand for a mission management system that must fulfill the following requirements:

- An **initial plan** for the whole mission must be computed off-line, in order to be validated by human operators; the plan must then be dispatched to every vehicle.

- As no communication is possible when the AUV is underwater, and even communication with the AAV is uncertain (due to the range of communication means, the presence of waves, …), the system must be **distributed**, i.e., each vehicle must be able to manage the plan execution on its own.
- Several failures could occur during the mission, and they have to be managed as smoothly as possible; a **local management of failures** is then required, as it may not be possible to communicate neither with an other vehicle, nor with the operator.
- The overall system must be **scalable**, in several senses:
  - it must be able to cope with missions involving large areas, with long-term missions, and therefore manage a possibly high-number of failures;
  - it must be easy to maintain and modify, in order to be applied to other missions, with numerous and different vehicles.

3 Contribution

To deal with such demanding autonomous missions, we have developed a multi-layer, distributed architecture for multi-robot plan management, named HiDDeN (Gateau et al. 2013). In this paper, we present HiDDeN in a deeper way, including algorithms for planning and plan distribution, as well as a detailed description of the execution and repair processes.

We also present an application of HiDDeN to the air-sea securing mission. We claim that the local execution and repair processes of HiDDeN are suited to long-term autonomy missions, especially involving submarine robots. Such missions indeed demand for a smooth integration of communication constraints, as communications are highly constrained in marine environments. They also demand for the management of the unavoidable failures that occur during such missions. HiDDeN deals with these failures with a hierarchical repair approach, that tries to minimize the impact of the repair on the mission, while adapting to the communication availabilities. We illustrate these features with simulations that show that the hierarchical repair approach of HiDDeN scales well with the number of failures, and we compare with a *full replan* approach. Another major contribution of this paper is the implementation of the developed deliberative software architecture on several vehicles and experimentations on the field.

4 The HiDDeN architecture

In the operational context of the mission, facing disturbing events and real communication constraints, the choice has been logically made to distribute the deliberative architecture on each vehicle. In that purpose, we have developed a specific layer for the management of the team, called HiDDeN (High-level Distributed DecisioN layer, Fig. 2).

HiDDeN is made of a set of HiDDeN supervisors (one per vehicle) that:

- manage the plan execution in a distributed way (including communication tasks),
- send action requests by interfacing with robots' control architectures,
- trigger plan repairs when needed, possibly orchestrating multi-robot repairs when communication is (partially) possible.

To implement these functionalities, the HiDDeN supervisors are structured into several modules (Fig. 3). While these modules have been implemented as separate threads within the HiDDeN supervisor, the HiDDeN supervisor, as a whole process, has been implemented using standard robotic middleware depending on the platform (e.g. as a YARP or ROS node). HiDDeN modules are briefly presented in this section. The underlying algorithms and procedures contained in the several modules are detailed in the following sections.

4.1 `DataManager`: providing mission-related information

The **`DataManager`** is responsible for updating and providing information about the mission. It relies on Koper (Gateau et al. 2012), an instance of an ontology that contains information about the vehicles' states, the current plan, the vehicles' actions, the mission goal, etc. It distinguishes local data (i.e. managed by the vehicle on which the Koper instance is embedded on), that can be get/set from the vehicle *control architecture*, and remote data (i.e. managed by other vehicles) that are updated when a communication is made with teammates.

4.2 `ExecMessenger`: interfacing with local control architectures

The **`ExecMessenger`** module represents the interface of the *HiDDeN supervisor* with the *control architecture* of the vehicle. In general, autonomous robots have a specific communication protocol allowing interaction with an operator or an internal supervision system that indicates which task has to be achieved. The current task selected by the **`MissionManager`** is then translated by **`ExecMessenger`** into a form understandable by the *control architecture*, using the appropriate protocol (e.g., using YARP bottles or ROS topics). Execution results

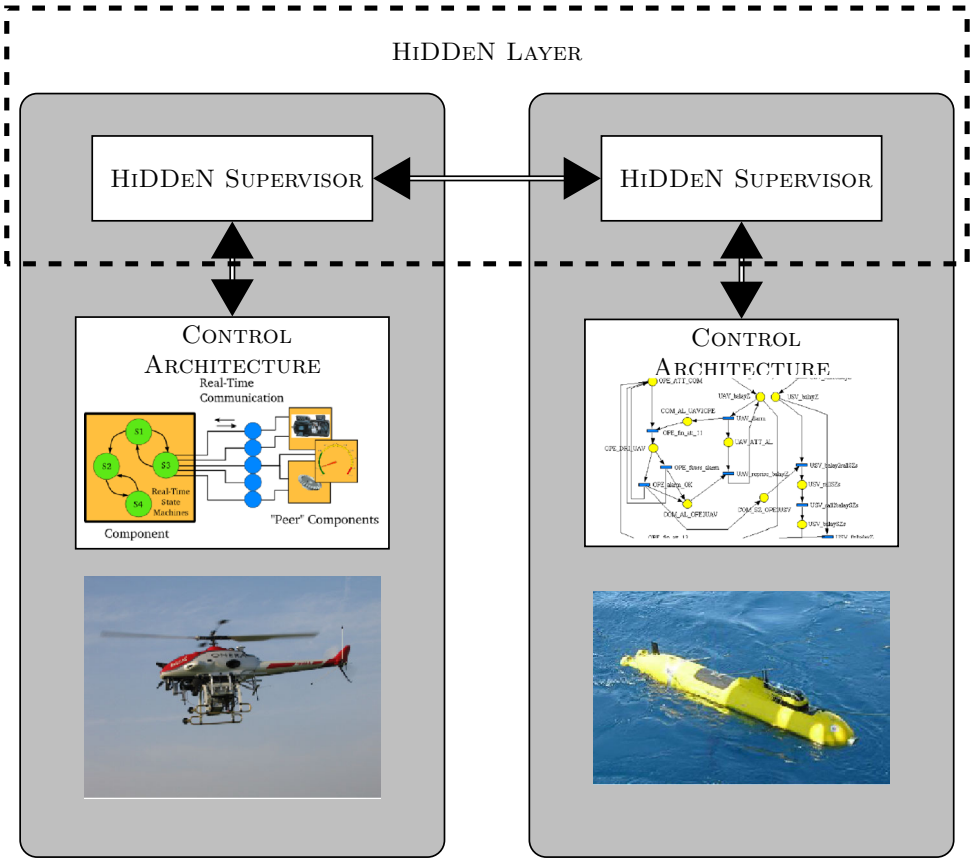

Fig. 2 HiDDeN layer on a team of two robots

are then translated from the *control architecture* to the **`MissionManager`**.

4.3 `MissionManager`: managing plan execution

The **`MissionManager`** manages plan execution, detects failures, and triggers synchronization tasks between vehicles and the repair process. Mission execution is detailed in Sect. 6.

4.4 `PlannerManager`: repairing hierarchical plan

The **`PlannerManager`** takes care of the interaction between the *HiDDeN supervisor* and the embedded planning function implemented with one or several planners. These planners allow a repair or replan of the mission plan according to the encountered failures and to the current situation. Hierarchical plan repair is detailed in Sect. 6.2.

4.5 `CoopMessenger`: communicating and synchronizing

The **`CoopMessenger`** deals with synchronization and is compulsory for team cohesion. It allows *HiDDeN supervisors* to interact, using communication means provided by the autonomous vehicles. These communications are twofold: on one hand, they concern nominal communications that are included in the plan. On the other hand, a specific communication protocol has been implemented to synchronize the robots when a multi-robot repair is necessary. This protocol is detailed in Sect. 6.3.

5 Planning and plan representation

Before the mission start, an initial plan is computed off-line. For that purpose, the planner algorithm takes into account the operational constraints raised by the human operators who monitor and control the mission. These constraints concern both some topological constraints (definition of the mission area, sub-areas, flight altitude for the AAV, immersion of the AUV, …), and some operation organization between tasks. For instance, it is asked by the operators to first scan a sub-area using the AAV, then launch the AUV when the AAV has finished its scan.

The problem and its operational constraints are modeled as a Hierarchical Task Network (see Sect. 5.1). Unlike classical planners that produce a totally or partially ordered set of tasks as a solution plan, our planning algorithm (detailed in Sect. 5.3) produces a plan represented as an instantiated HTN

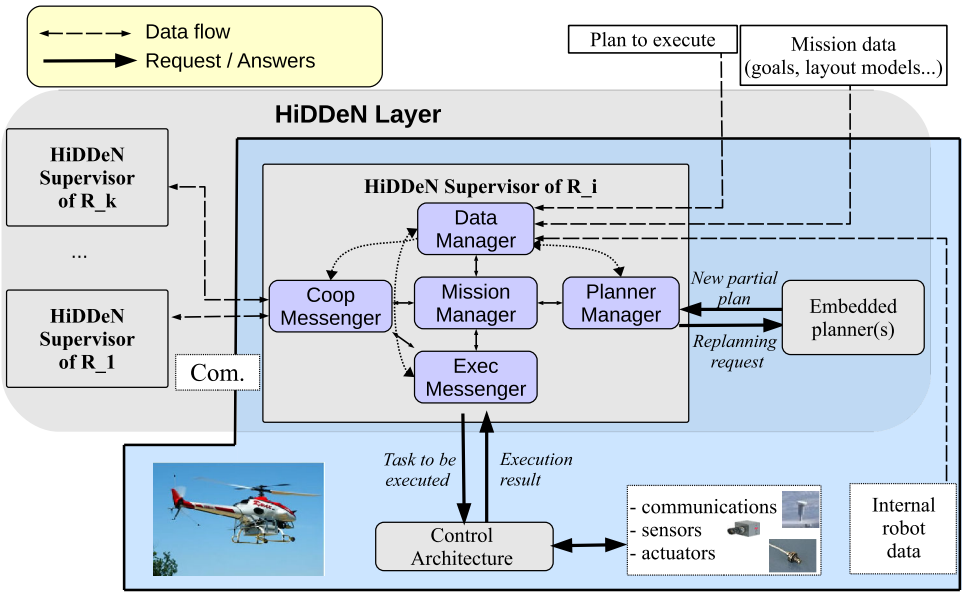

Fig. 3 Overview of the internal modules of an *HiDDeN supervisor*

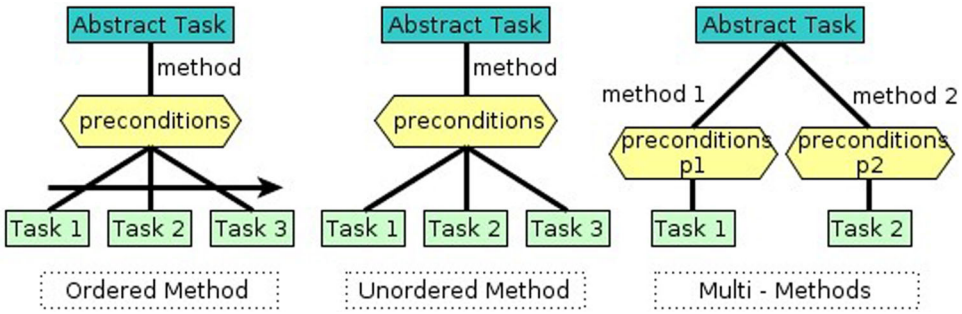

Fig. 4 Types of methods for the HTN formalism

(Sect. 5.2) that explicitly keeps the hierarchical causality in order to ease plan repair over complete replanning.

Finally, this plan is distributed to every vehicle performing the mission (Sect. 5.4). This distribution step is useful for, first, giving to each robot which task it has to do, and second, ensuring some communication constraints between the vehicles when synchronization is mandatory to fulfill the mission.

5.1 Problem formalization framework

We use Hierarchical Task Networks (HTN, Erol et al. 1994), a formalism used to represent planning problems. An HTN is a set of tasks that are either abstract or elementary. Abstract tasks have to be decomposed into (sub-)tasks using methods (Fig. 4). At the lower level, elementary tasks are the actions that will be performed by the *control architecture* of a vehicle. A method has preconditions to be satisfied before it can be selected. The sub-tasks associated to one method (if more than one sub-task) can be either ordered (to be performed in sequence) or unordered (no constraint between them).

Two main advantages can be stated for the use of the HTN formalism in deliberative architectures:

- The human expertise can be inserted in the definition of the mission: operator and experts know the high level tasks to be performed and some relationships between these tasks; they have some ideas about the strategy of reaction to disturbing events; the planning function work is then further to instantiate variables on tasks such as which vehicle or which temporal interval should be associated to a given task.
- The hierarchical structure is well suited for the repair of the plan during the execution process; when a task cannot be performed, the principle is to find a solution that minimizes the number of communications and by extension the number of vehicles: the repair is as local as possible.

5.2 Plan representation

HiDDeN uses an *instantiated HTN* of the mission plan (noted iHTN in the following) as the core model of plan execution monitoring. An iHTN $\mathcal{H}$ is a tuple $(E, V, Pre, Post, T_E, T_A, \Re, M, t_r)$ such that:

- E is the set of *labels* of tasks and methods;

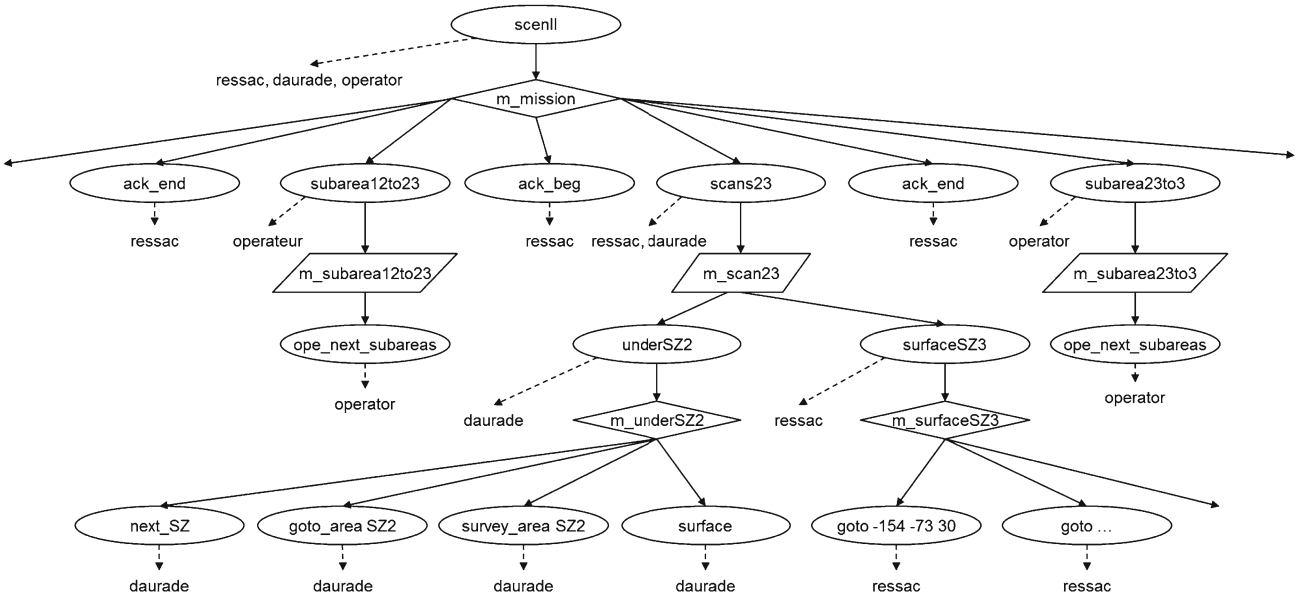

Fig. 5 Part of the iHTN for the securing mission for the AAV (ressac), the AUV (daurade) and the operator

- V is the set of *instantiated variables*;
- Pre is the set of *preconditions*;
- $Post$ is the set of *postconditions*;
- $T_E \subset 2^{Pre} \times 2^V \times 2^{Post}$ is the set of *elementary tasks*. An elementary task t_e is a triplet $(Pre(t_e), V(t_e), Post(t_e))$ with $Pre(t_e)$ a list of preconditions, $V(t_e)$ a list of parameters, and $Post(t_e)$ a list of post-conditions;
- $T_A \subset 2^V \times 2^M \times 2^{Post}$ is the set of *abstract tasks*. An abstract task t_a is a triplet $(V(t_a), M(t_a), Post(t_a))$ with $V(t_a)$ a list of parameters, $M(t_a)$ a list of methods, $M(t_a) \neq \emptyset$, and $Post(t_a)$ a list of post-conditions;
- $\Re$ is a set of partial ordered relations applied to the set of tasks $(T_A \cup T_E)$. We consider that $rel \in \Re$ is either sequential (or ordered, noted $rel = \prec$) or unordered (noted $rel = \sim$);
- $M \subset T_A \times 2^{Pre} \times 2^{(T_A \cup T_E)} \times \Re$ is a set of *methods*. A method m is a quadruplet $(t_m, Pre(m), st, rel)$ with t_m the abstract task to which the method is applied, $Pre(m)$ a list of preconditions, st a set of tasks, rel a sequential or a unordered relation between the st elements. rel provides a mean to define an execution order for tasks of the st set;
- t_r is the *root* abstract task that must be executed; this is the highest level of abstraction in the $\mathcal{H}$ tree.

For each task t, $V(t)$ contains, among other variables, the set of vehicles involved in the task. Note that this definition is similar to the general HTN definition: an iHTN is a subset of an HTN where the planner has previously decided the methods and the variables to use. With deterministic planners (as the one we actually use), the plan contains one method per abstract task.

5.3 Air-sea securing mission planner

The algorithm we have designed for the air-sea securing mission (described in Sect. 2) is a specific algorithm that:

- ensures the hierarchical constraints given by human operators; these constraints have been expressed in natural language, and we formalized them using an HTN representation;
- computes each vehicle trajectory to ensure the survey of each sub-area.

This computation is specific to the mission, but has been made as generic as possible regarding the characteristics of the mission (number of sub-areas, geometry of areas, …) or the robots skills (velocities, turning radius, …) The resulting plan is represented as an iHTN.

Figure 5 shows a part of the iHTN built by the planner for the scenario. This figure must be read from left to right and from top to bottom. Ovals represent tasks. Leaf tasks of the tree are elementary tasks. The agents (vehicles and mission operator) involved in a task are linked with a dotted arrow. Methods name begins with *m_*; methods are ordered when represented by a diamond and unordered when represented by a parallelogram. The part of the iHTN shown in Fig. 5 highlights that the operator has to validate the survey of the first sub-area (abstract task *subarea12to23*) and give a clearance to the vehicles to go on with the next

sub-area. Task *ack_beg* specifically indicates that communications between the operator and the vehicles (AAV and AUV) must go through the AAV. Exploration of the next sub-area (task *scan23*) will be performed by the two vehicles in parallel, the AUV scanning SZ_2 (task $underSZ_2$) and the AAV scanning SZ_3 (task $surfaceSZ_3$). Task *ack_end* indicates that communications between the AUV and the operator must go through the AAV.

We can notice that the plan contains elementary tasks that have a different level of detail depending on the vehicle. Indeed, the skills of each vehicle are different: the AUV can perform *surveys* by its own, while the AAV can only perform *goto* actions and then needs that the mission planner decomposes its survey into elementary goto. The actions available at the robot level, that are implemented within the control architectures of each robot, are described by *services* within the Koper ontology (Gateau et al. 2012). In the air-sea securing mission, only one vehicle of each type is available (AAV and AUV), therefore we have not implemented some selection or allocation process within the planner. Such description is however used to select a vehicle that provides the requested services in air-ground multi-robot missions (Gateau et al. 2013), in which we use the HTN SHOP2 planner (Nau et al. 2003).

Figure 6 shows the planned trajectories of the vehicles computed for the successive surveys of the sub-areas SZ_1, SZ_2 and SZ_3, defined for the experimental demonstration. When the AUV surveys the last sub-area SZ_3, the AAV has a waiting pattern. Trajectories take into account the velocities of the vehicles, the field-of-view of their sensor, as well as the maximal turning radius of the AUV.

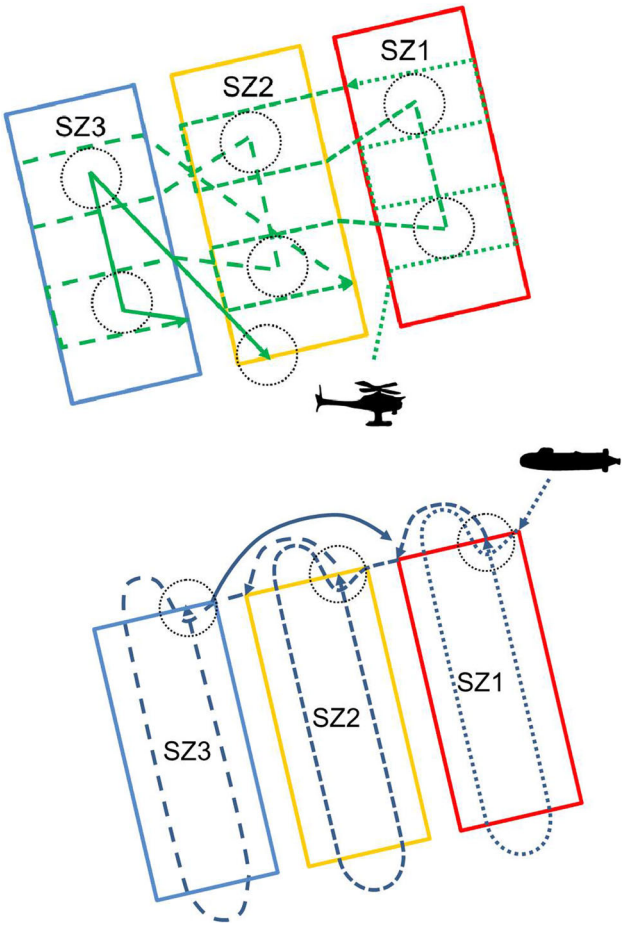

Fig. 6 Scans computed for each vehicle to survey the three successive sub-areas for the securing mission (sub-areas are spaced for better view)

5.4 Distribution of the global iHTN

As the HiDDeN supervisors are distributed among the robots, the initial mission plan has to be given to each robot. We propose a distribution process that will help minimizing communication needs by:

- Removing tasks that are not relevant to that robot from the plan, leading to a local plan adapted to the robot; this cleaning process helps maintaining plan consistency during plan repair: modifying the plan of a sub-team has no impact on the plan of robots that are not involved in the repaired task.
- Inserting necessary communication tasks that are implicit but mandatory in the mission plan, determined by dependencies between tasks achieved by different robots.

More precisely, the global iHTN $\mathcal{H}$ is distributed to the team of robots, providing each robot r_i with a local plan $\mathcal{H}_i$. This local plan is computed using the recursive Alg. 1, starting from the root task t_r of $\mathcal{H}$.

Each elementary task (line 26) is simply either kept if robot r_i is concerned (line 28), or removed otherwise (line 30). Regarding abstract tasks (line 1):

- when a method is unordered, the sub-tasks in which r_i is not involved are removed (lines 4 to 8);
- when a method is ordered (line 11):
  - if r_i is involved in task t_k but not in t_{k+1}, r_i has to warn that it has finished doing t_k: a communication task is inserted (line 17);
  - if r_i is involved in task t_k but not in t_{k-1}, r_i has to wait that t_{k-1} has been finished: a communication task is inserted (line 20);
  - otherwise the sub-tasks are removed (line 23).

Figure 7 shows the local plan for the AUV distributed from the global plan of Fig. 5. Some communication tasks have been added to the plan: the AUV must wait for the AAV message before beginning to survey sub-area SZ_2 (task *?com(ack_beg)*); then it informs the AAV that it has ended

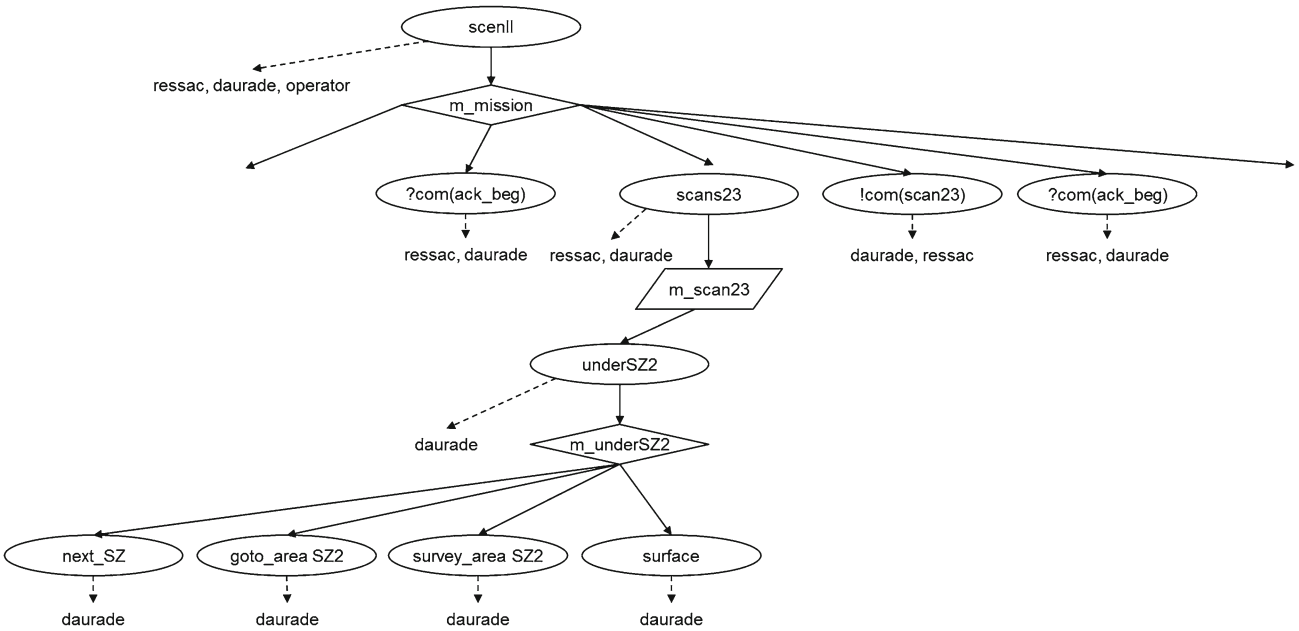

Fig. 7 Part of the iHTN for the securing mission scenario on-board the AUV. Inserted communication tasks are written with a question mark for waiting tasks, and an exclamation point for sending tasks

Algorithm 1: DISTRIBUTE(t, r_i)

```
if t = (V(t), M(t), Post(t)) ∈ T_A then
    foreach m_j ∈ M(t) do
        let m_j = (t, Pre(m_j), subtasks, rel)
        if rel = ∼ then
            foreach t_k ∈ subtasks do
                if r_i ∉ V(t_k) then
                    subtasks → subtasks \ {t_k}
                else
                    DISTRIBUTE(t_k, r_i)
        else
            // rel =≺
            let subtasks = (t_k)_{1≤k≤K} such that
            ∀i ∈ [[1..K − 1]], t_k ≺ t_{k+1}
            for k in 1..(K − 1) do
                DISTRIBUTE(t_k, r_i)
                if r_i ∈ V(t_k) then
                    if r_i ∉ V(t_{k+1}) then
                        τ = SendMessage(t_k)
                        subtasks ← subtask ∪ {τ} such
                        that t_k ≺ τ ≺ t_{k+1}
                    else if r_i ∈ V(t_{k+1}) then
                        τ = WaitMessage(t_k)
                        t_k = τ
                    else
                        subtasks → subtasks \ {t_k}
                DISTRIBUTE (t_K, r_i)
else
    // t ∈ T_E
    if r_i ∈ V(t) then
        do nothing // task executed as is
    else
        t ← ∅ // task is removed from H_i
```

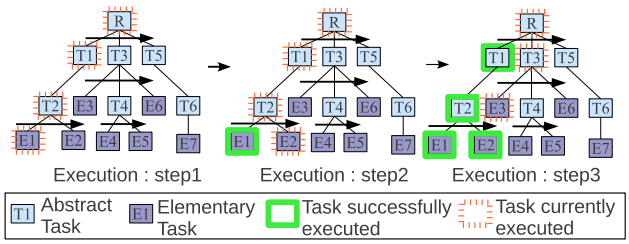

Fig. 8 Execution of an iHTN: HiDDeN processes root task R, and goes down to task E_1. When E_1 is achieved, the following task is E_2. Then abstract tasks T_2 and T_1 are considered achieved, and the next executed task is E_3

this task (task *!com(scan23)*), and finally waits again before going to survey the following area (task *?com(ack_beg)*). Tasks that only concern the AAV have been removed by the distribution algorithm.

6 Mission execution and repair

Plan execution is carried out by the **`MissionManager`**. It executes the robot local plan by scanning it in a depth first (i.e. descending to elementary tasks) and left first (i.e. enforcing the ordered relations between tasks) manner (an example is given on Fig. 8).

When an elementary task (a leaf) is reached, the corresponding action is executed. It is delegated to the **`ExecMessenger`** if the task is a vehicle task (result-

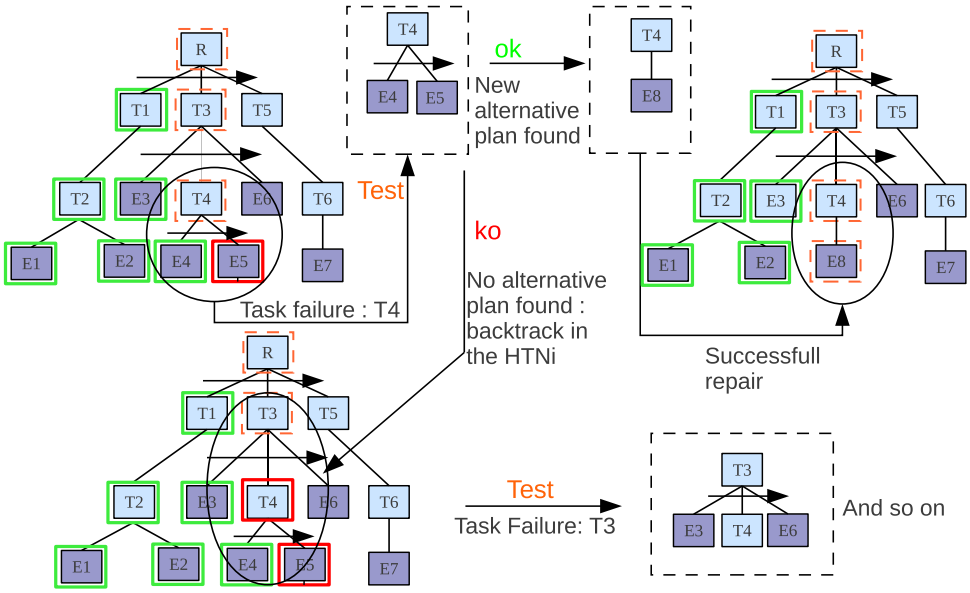

Fig. 9 The repair process: elementary task E_5 fails, which implies a repair of task T_4 (*top*, *left*). If a new plan is found, the iHTN branch is replaced (*top*, *right*); otherwise, the repair process is applied to T_3 (*bottom*)

ing in a request sent to the *control architecture*) or to the **`CoopMessenger`** if the task is a communication task.

6.1 Fault detection

During the execution of the mission plan, faults or disturbances may occur. We introduce several means to detect such failures in HiDDeN. For each task t to be executed, the following failures can arise:

- **invalid preconditions:** before executing task t, the **`MissionManager`** checks that the task preconditions $Pre(t)$ are met, i.e. that the action can be executed in the current state; otherwise, it raises an *invalid preconditions* failure;
- **error:** when the request to perform t is sent to the robot control architecture, an execution report is expected; this report can be either a success report or a failure report; in the later case an *error* is raised;
- **invalid effects:** when task execution reports successfully, the **`MissionManager`** checks that the system state is consistent with the expected effects $Post(t)$ of the task; otherwise, it raises an *invalid effects* failure;
- **timeout:** if a maximal duration is defined for task t, the **`MissionManager`** raises a *timeout* failure if the request has not reported before the duration expired.

When one of these failures appears, the plan has to be repaired.

6.2 Hierarchical plan repair

The basic idea of plan repair in HiDDeN is that when a task fails (whatever the failure rationale), the priority is to replace it (and only this one) by an alternative task that could allow to achieve the mission. If such an alternative does not exist, we take advantage of the hierarchical structure of the iHTN plan to replan only a sub-tree of the plan that has a valid alternative, not the entire mission.

More precisely, when an elementary task fails, the supervision function asks the planning function to solve a new local planning problem. This new local planning problem is generated as a *planning request* by the **`PlannerManager`** according to the current system state and the failure status. The current system state includes the position of robots, as well as the status of the tasks in the current plan (i.e., completed tasks, ongoing tasks, failed tasks). This *planning request* is then sent to the embedded planner. If the planner returns a new plan, this plan then replaces the failing task in the iHTN. Otherwise, the process goes up in the iHTN structure to repair the parent task in an iterative manner, until a valid alternative is found or the root task is reached (Fig. 9).

Figure 10 shows the iHTN that replaces the *survey_area*SZ_2 task of Fig. 5 in case of failure. When a mine is localized during the scan of sub-area SZ_2, the control architecture indeed reports a failure of the current task and adds the mine information in the current system state. The new repairing iHTN consists in having the AUV go to the surface, communicate with the AAV in order to send the mine information to the operator, and finally resume the survey of SZ_2.

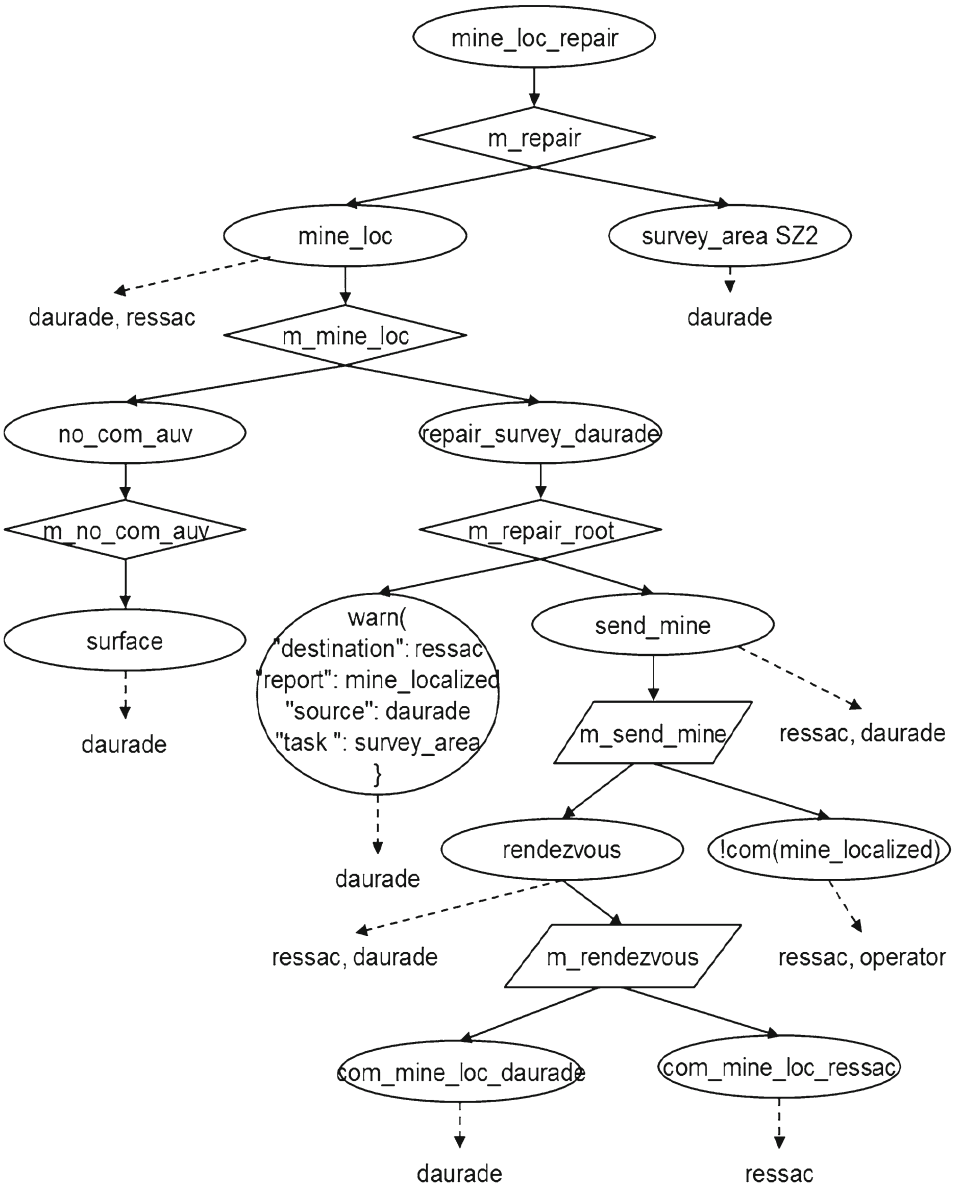

Fig. 10 iHTN repair task when a mine is localized by the AUV

6.3 Multi-robot plan repair

The repair of a task (an elementary task or an abstract task that "failed" because of the bottom-up repair process) will affect all the vehicles involved in its achievement. The repair of a multi-robot task will therefore need a synchronization among the involved vehicles, carried out by their **`CoopMessenger`** modules. This synchronization is enforced by a protocol described into predefined iHTN schemes. This process is illustrated with the sequence diagram of Fig. 11. In this example, robot R_1 has to repair the failing task E_4. As no local plan is available to repair E_4, the repair process goes up to the parent task T_2, that involves both R_1 and R_2. The cooperative repair process is applied, that is modeled as an iHTN and inserted into the current plan, replacing E_4.

The execution of this iHTN is illustrated by the *RepairingPattern* block of the sequence diagram:

- the HTN-FAIL call warns R_2 that a repair is needed;
- R_2 sends its database so that R_1 can update the current state (message BDD-UPDATE);
- R_1 proceeds to the repair of task T_2, resulting in a new task T_6;
- R_1 sends T_6 to R_2 (message PLAN-UPDATE);
- finally T_6 is inserted into the plan, replacing T_2, and the plan execution can go on for both robots.

If the HTN-FAIL call fails, meaning the communication between R_1 and R_2 is not possible at the moment, a similar *CommunicationPattern* is applied to help R_1 establish

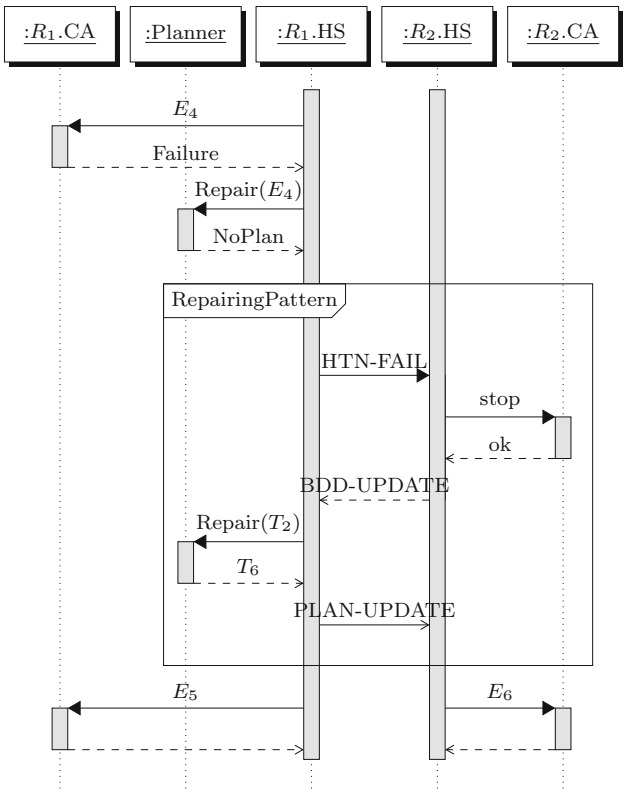

Fig. 11 Sequence diagram of synchronization of two robots for a plan repair. R_i.CA represents the *control architecture* of robot R_i, R_i.HS the *HiDDeN supervisor* of robot R_i

the communication with R_2. This pattern tries several predefined recovering strategies: communication relaying via other robots, area exploration to find R_2, and so on. In the air-sea securing mission, the *CommunicationPattern* that have been implemented consist in:

- for the AUV (when a communication with the AAV failed), to go to surface if under water, or to stay at the same position (hold on a GPS fix), and try to communicate again;
- for the AAV (when a communication with the AUV failed), to go to another planned point of the area, and try to communicate again.

7 Simulation results

In this section, we describe some simulation results that first compare the HiDDeN hierarchical repair process to a *full replan every time* strategy, and second evaluate the robustness of the HiDDeN architecture when facing numerous failures. Simulations use MORSE (Modular OpenRobots Simulation Engine, Echeverria et al. 2012), a versatile simulation infrastructure for robotic architectures. MORSE simulates both the dynamics of robots (in our case with simplified kinematics) and standard robotic sensors (camera, lasers, GPS, inertial measurement unit, …) MORSE then helps the migration from simulations to real experiments in which software must be embedded on the platforms.

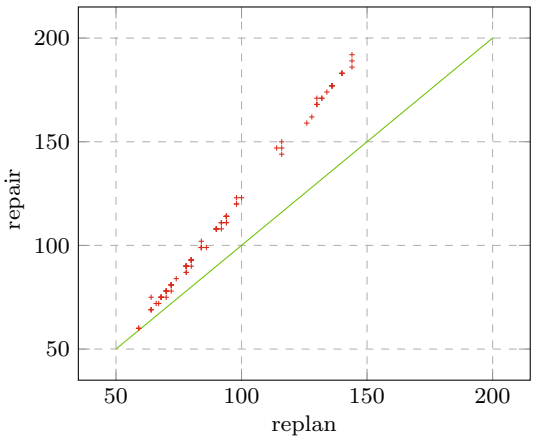

Fig. 12 Number of executed tasks per mission (repair vs. replan)

7.1 Hierarchical repair versus full replan

We have made a hundred of simulations to evaluate the hierarchical repair process implemented in HiDDeN (Sect. 6.2) in comparison with a strategy that would systematically replan all the sequel of the mission when a failure occurs.

To realize these simulations, we only use one kind of failure, and play with other characteristics of the mission to have some variability in the mission duration and the number of failures that occur. We then defined a simulation environment with no mine, but we modified the number of sub-areas as well as the relative velocities of the vehicles. The only failure being the communication failure (after a timeout), modifying the relative velocities of the vehicles indeed creates some high variability in the number of timeouts that occur for a given number of sub-areas.

The results of these simulations are presented in the following using scatter plots: for each simulation, we plot the measured metric value of the HiDDeN *repair* process with respect to the value of the *replan* strategy.

Figure 12 shows the number of tasks that have been executed on each mission. As the repair strategy just inserts some new tasks in the plan, its number of tasks is clearly greater (around 1.5 times the number of tasks of the replan strategy).

Figure 13 shows the total planning time for both strategies. The cumulated planning time of the repair strategy is around 10 times shorter than the one of the replan strategy.

The implemented repair strategy behaves as expected: it inserts a small set of tasks in the plan for each failure, then having more executed tasks greater than in a full replan approach, while having a shorter planning time. However, it

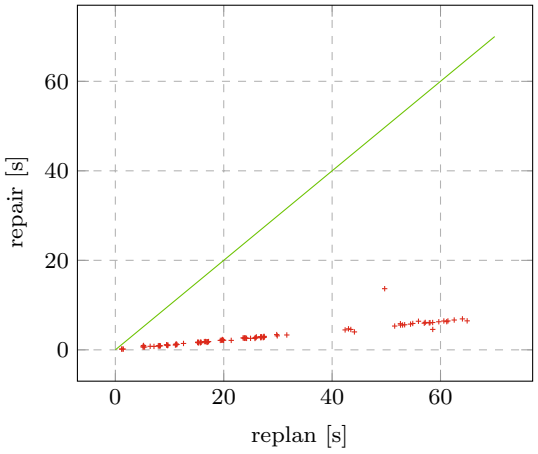

Fig. 13 Total planning time per mission (repair vs. replan)

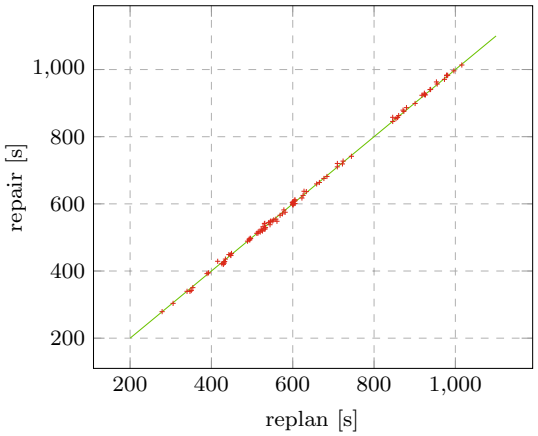

Fig. 14 Mission duration (repair vs. replan)

is interesting to notice that the total mission duration, shown in Fig. 14, is almost identical for both strategies. When some communication timeout occurs, it indeed means that one vehicle is slower than the other, and then the duration of the mission is mainly dependent on the time the slowest vehicle will take to survey all the area.

Whereas the mission durations show no clear advantage of using HiDDeN, it must be noticed that HiDDeN: (1) requires less computational power when repairing, (2) scales better to mission constraints. It is easier to define parametric procedures for repairing at several hierarchical levels (for instance procedures for one vehicle when no communication is possible; procedures for several vehicles when a communication is established) than defining a global planning algorithm that implement such procedures. Moreover, implementing local repairing procedures will ask for less communication.

7.2 Robustness to failures

In order to evaluate the robustness of the HiDDeN architecture, we have made a hundred of simulations in which the

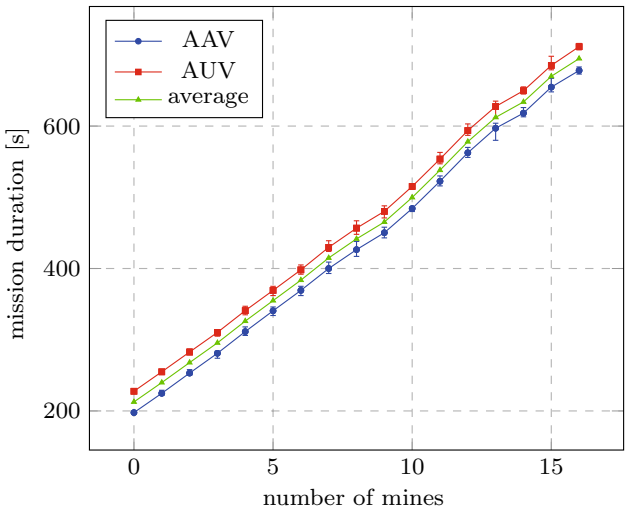

Fig. 15 Mission duration according to the number of mines

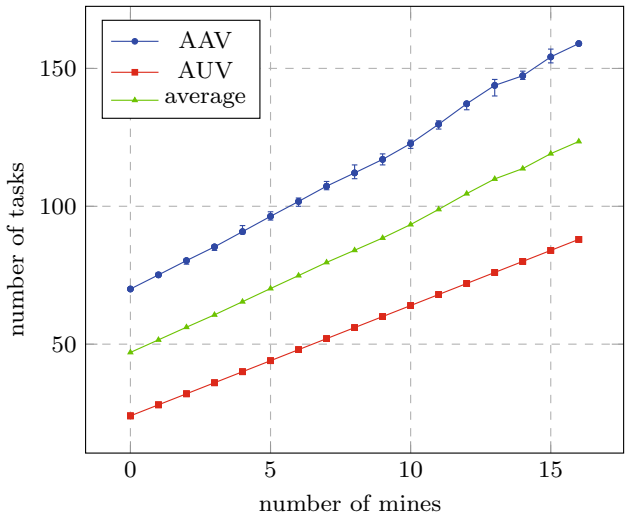

Fig. 16 Number of tasks according to the number of mines

number of mines in the environment increases from 0 to 16. In such simulations, two kind of failures can occur: a mine is found by the AUV, and a communication timeout occurs on the AAV when it waits for a communication with the AUV.

Figures 15 and 16 show the evolution of the mission duration and the number of executed tasks according to the number of mines. These two metrics scale almost linearly with the number of mines.

Figure 17 shows the number of repairs according to the number of mines. The AUV detects mines in the seabed, and consequently its number of repairs is linear in the number of mines. It is nevertheless interesting to notice that the number of repairs of the AAV, i.e. communication timeouts with the AUV, also scales linearly.

The idle and repair ratios, are respectively shown on Figs. 18 and 19. The idle ratio (i.e. time spent waiting for com-

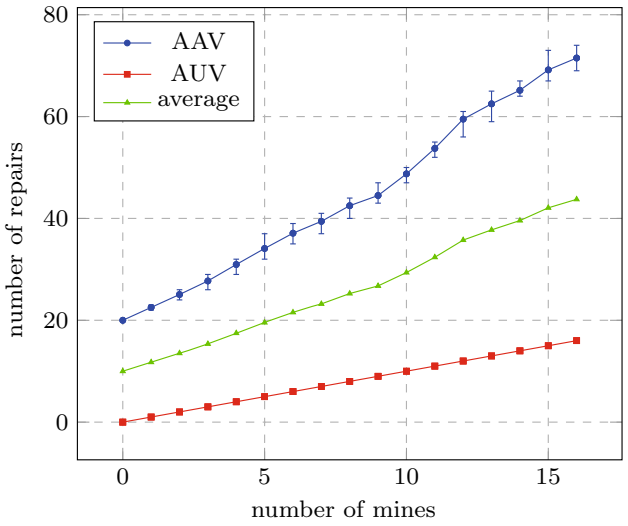

Fig. 17 Number of repair according to the number of mines

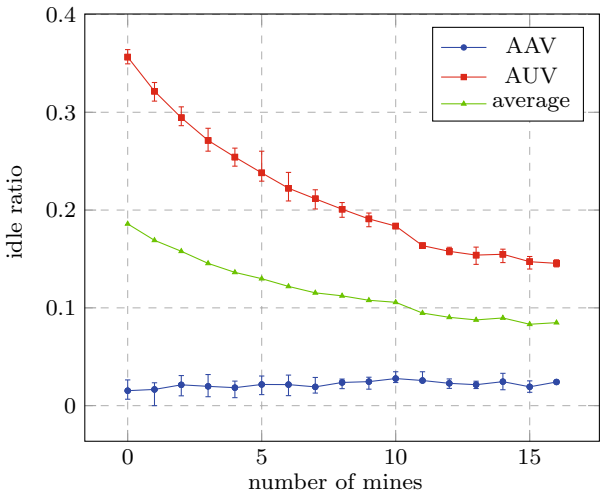

Fig. 18 Idle ratio according to the number of mines

munication over total mission time) of the AUV decreases when the number of mines increases, which is expected as more time during the mission is spent in repairing, and then not in waiting communications. Anyway, it is interesting to see that the idle ratio of the AAV is almost constant, which emphasizes that HiDDeN perfectly manages concurrent disturbances: mine detection in the AUV, communication misses in the AAV.

The repair ratio (i.e. time spent in recovering from a failure over total mission time) has a surprising but logical evolution. In a first reasoning, we might have expected a linear evolution, as the mission duration and the number of repairs have a linear evolution. Nevertheless, the survey of a sub-area that contains a lot of mines is almost only made of repairs (i.e., the AUV detects a mine at every simulation step), and

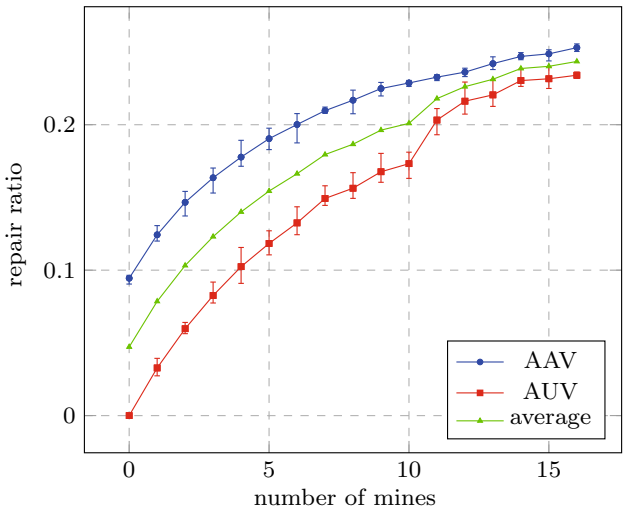

Fig. 19 Repair ratio according to the number of mines

it is then logical to see that the repair ratio increases while asymptotically reaching a ceiling value.

As a conclusion, these last results prove that the HiDDeN architecture, along with the proposed hierarchical repair process, scales correctly with respect to the complexity of the environment, which will lead to a potentially high number of disturbances.

8 Field experiments

We implemented and applied all the above software architecture for planning, execution and plan repair on the air-sea securing mission. We were able to run at least one complete experiment of the scenario on the chosen experimental field. We also demonstrate the robustness of our architecture by adopting an evaluation process made of *software architecture in the loop* hybrid simulations (in which some robots are simulated while others are real). In this section, we describe some practical results from field experiments (from the actual experiment and hybrid simulations).

8.1 Platforms

Two platforms were available to demonstrate the air-sea securing mission: the ReSSAC AAV from ONERA, and the Daurade AUV from DGA.

8.1.1 The ReSSAC AAV

Technical data for the ONERA ReSSAC AAV (Fig. 20) are given on Table 1.

The ground control station, implemented in a van (Fig. 21), allows its transportation to experimental areas and provides a mission management system and a video control screen.

Fig. 20 ONERA's ReSSAC AAV platform

Table 1 Technical data for the ReSSAC AAV platform

Brand	Vario
Type	Benzin
Total length w/ and w/o blades	1.78 and 1.63 m
Total width w/ and w/o blades	1.78 and 0.4 m
Total height	0.7 m
Empty weight	11 kg
Maximum takeoff weight	15 kg
Engine	Two stroke 29 cm^3
Flight time	20 mn
Mean speed	20 km/h

Fig. 21 ReSSAC ground control station

The on-board software architecture, based on the Orocos middleware (Soetens and Bruyninckx 2005), gives the following capabilities to the AAV (Watanabe et al. 2010; Chanel et al. 2013):

- to reach a waypoint defined by its GPS coordinates;
- to follow complex routes while avoiding obstacles;
- to detect and track objects of interest.

8.1.2 The Daurade AUV

Daurade (Fig. 22) has been designed for Discrete Rapid Environmental Assessment (REA) missions, performing hydrographic and oceanographic surveys.

Technical data for this DGA TN AUV are given on Table 2.

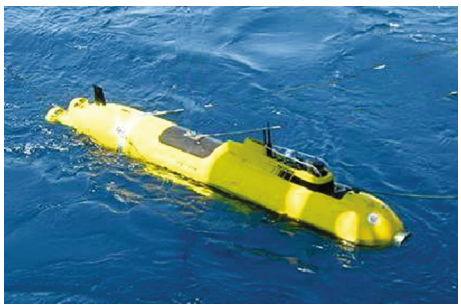

Fig. 22 The Daurade platform of DGA TN

Table 2 Technical data for the Daurade platform

Brand	ECA
Length	5 m
Diameter	0.7 m
Weight	1 t
Maximal immersion	300 m
Energy autonomy	10 h at 4 kts, 2 h at 8 kts
Mean speed	4 kts
Turning radius	25 m

The on-board architecture is based on ProCoSA©, that allows to model the desired nominal and non nominal behavior of the vehicle with Petri nets (Barbier et al. 2006). This architecture gives the following capabilities to the AUV (Barbier et al. 2011):

- to reach an area defined by a polygon and its WGS84 coordinates;
- to compute scans in order to survey a given area respecting maneuver constraints;
- to achieve the inspection of a detected object.

8.2 Hybrid simulations

In order to prepare the experiments with all vehicles, we first decided to make many hybrid simulations. In this section, we focus on the AUV architecture that is the more demanding in terms of logistics; the AAV architecture is then simulated.

The Daurade AUV from DGA TN is located in Brest, France. The interest to perform hybrid simulations on the Brest roadstead (Atlantic Ocean) is the minimization of the logistic for Daurade: the roadstead is its main experimental field, and the launch and recovery of the platform is quite often made from boat *L'Aventurière* which is dedicated to experiments.

In this hybrid setup, the sensors/actuators of the AAV are simulated in MORSE. The *control architecture* of the AAV is then connected to MORSE and runs on a dedicated computer, located in the boat. The software configuration of communi-

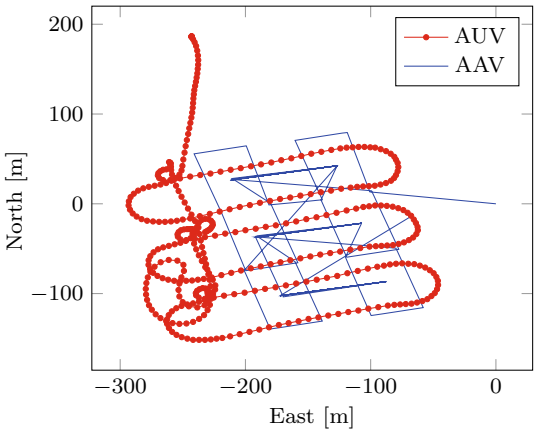

Fig. 23 Vehicles' trajectories on a hybrid simulation with no mine

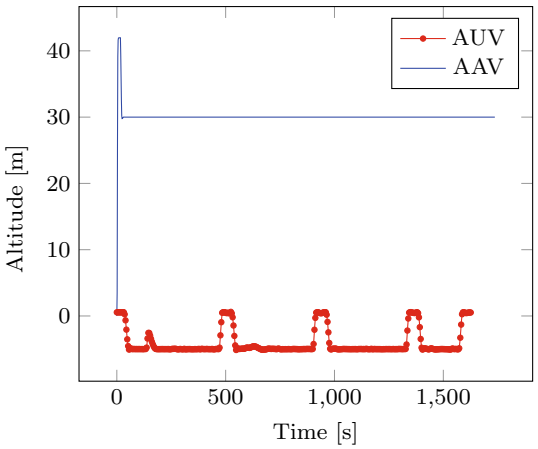

Fig. 24 Vehicles' altitudes on a hybrid simulation with no mine

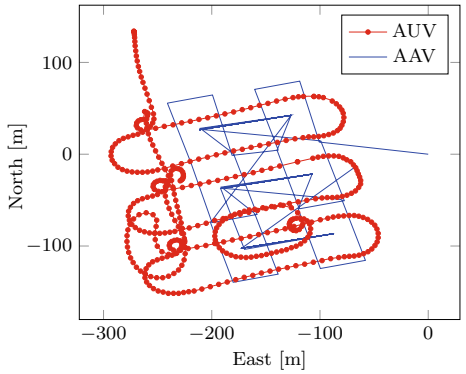

Fig. 25 Vehicles' trajectories on a hybrid simulation with one mine

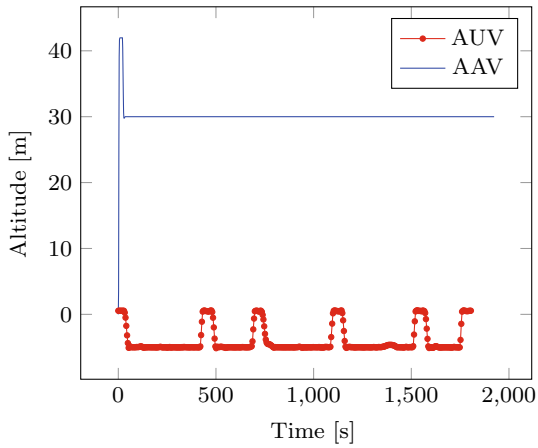

Fig. 26 Vehicles' altitudes on a hybrid simulation with one mine

cations between the AAV, the AUV and the mission operator is then similar to the final experiment with the real vehicles. A lot of tests and setups were performed, and the securing scenario was completely achieved six times, three times without mine, and three times with one mine (the detection time of the mine was triggered randomly on a scan of the survey and emulated on-board).

Figure 23 shows a trace of the vehicles' trajectories (simulated AAV and real AUV) and Fig. 24 shows the evolution of their altitude over time for one of the three hybrid simulations with no mine.

Figure 25 shows a trace of the robots' trajectories (simulated AAV and real AUV) and Fig. 26 shows the evolution of their altitude over time for one of the three hybrid simulations with one mine: the AUV surfaces once more around $t = 700s$ to warn the AAV that it found a mine.

Table 3 presents the results of these hybrid simulations. Several metrics were measured: the mission duration (difference between vehicles are due to the time to go to their final retrieval point), the number of elementary tasks for each vehicle, the number of repairs triggered by each vehicle. The AUV always triggers as many repairs as the number of mines that have been detected. The AAV triggers repairs when it waits for a communication with the AUV more than a given timeout (30 seconds in these experiments). The repair makes the AAV look for the AUV elsewhere in the area in order to manage communication range (by doing a line scan).

The most interesting measures concern the *idle ratio*, which is the time when a vehicle is waiting for a message over the mission duration, and the *repair ratio*, which is the time spent in executing the repair iHTNs over the mission duration. The mean value over the three runs as well as the standard deviation were computed for all these metrics.

Standard deviation is quite big regarding the "raw" values (number of tasks and repairs, especially for the AAV) because these values are highly influenced by the time taken by the vehicles' motion (the AUV in this case, as the AAV is simulated). Indeed, depending on the time taken to join the next position, it will not only influence the mission duration, but also the number of repairs of the AAV due to the communication timeout. Nevertheless, it is interesting to notice that the two other metrics, which are relative to the mission duration, are not varying so much. Moreover, the values are quite similar regardless of the presence of mine in the scenario.

Table 3 Some results on hybrid simulations compared to the actual experiment

	Hybrid - no mine				Hybrid - 1 mine				Expe. - 1 mine	
	AAV		AUV		AAV		AUV		AAV	AUV
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.		
mission dur. (s)	1612.7	146.4	1713	68.5	1667	347.6	1835.7	274.6	2120	2172
nb of elem. tasks	152	59	29.3	4.7	160.7	66.9	38	8.7	92	28
nb of repairs	37.7	7.7	0	0	39.3	10.1	1	0	50	1
idle ratio (%)	9	1.1	10.3	3.3	7.8	1.5	7.8	1.1	5.2	26.3
repair ratio (%)	**11.3**	3.2	0	0	**11.9**	3.3	**2.1**	0.3	**12.2**	**1.7**

Fig. 27 First tests of the AUV in non-salty water

Fig. 28 Take-off of the ReSSAC AAV

8.3 Experiment with all vehicles

The test site to experiment the cooperation with real vehicles had to answer to two major conflicting constraints (in addition to regulation constraints for the AAV): the AUV has to dive sufficiently far from the shore, while the AAV must fly sufficiently close to the position of the safety pilot. The chosen site is a mountain lake in South-East of France which is 5 square kilometers large, has a depth of at most 100 meters and where sloping banks allow the vehicles to share the same mission area. While this forced the Daurade operators to run many tests in order to ensure that the AUV could navigate in non-salty water (the AUV has to have almost the same density as the water it navigates in), it has been the preferred location (Fig. 27).

As this site is almost unique in France, and quite far from the vehicles usual yards (1300 km for the AUV and 550 km for the AAV), it was possible to do only a very few runs with the whole team of robots. The human team in charge of the AUV consisted in 4 persons, plus the site operators involved in the launching itself (2 persons). The AAV team consisted in 3 persons, and 3 more persons were involved in the mission/cooperation layer. Only two periods of 3 days were available at the site as it is used for other military purposes.

Besides logistic difficulties, the large size of the area has been a challenge in itself, as one can imagine looking at Fig. 28. Communication and synchronization between safety operators during tests, as long as military safety procedures made every single test surprisingly time-consuming. Moreover, weather hazards (note that the altitude decreases the AAV lift) and GPS instability in such a mountainous area were an unexpected challenge.

A global view of the experimental area is shown on Fig. 29. The Daurade AUV has been moored and operated from a barge located on the lake (to the North of the mission area on Fig. 29, also seen on background of Figs. 27, 28). The ReSSAC AAV has been operated from its van located on a platform on the lake shore (to the East of the mission area on Fig. 29). The mission operator was set up on a building with a view on the lake. The three operation centers (the barge, the van and the building) where connected together through a WiFi connection. Each vehicle was then directly connected to its operation center using a WiFi connection (excepted when the AUV was underwater).

We have completed the securing scenario with one mine simulated on the seafloor. Results are given in Table 3. The mission lasted 2172 seconds (maximum of the two vehicles),

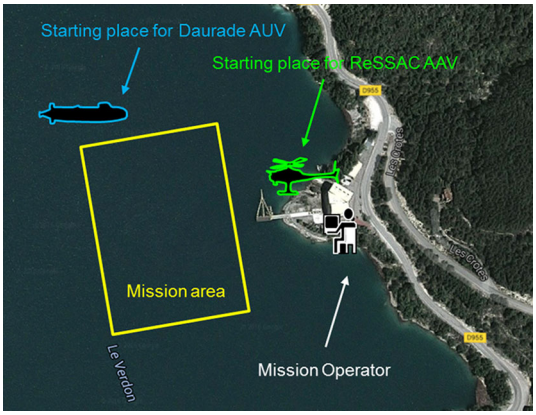

Fig. 29 Experimental area

1 mine has been found by the AUV, while the AAV made 50 repairs due to communication timeouts.

The idle ratio has been quite large compared to what was measured during the hybrid simulations. The main reason is that, during hybrid simulations, the actions of the operator where almost immediate, due to the fact that the AUV was almost ready at anytime (as already in the water), and the AAV was simulated. In the real experiment, the mission operator had to check that the vehicles were ready to proceed: he made around 3 min (against a few seconds in hybrid simulations) to validate the beginning of the mission. During this time, the vehicles were waiting for the communication with the operator, thus leading to an increased value of the idle ratio.

It is nevertheless interesting to notice that the repair ratio values (in bold) are consistent with the values from the hybrid simulations. This shows that, regarding the evaluation of the repairing strategies, hybrid simulations are representative of the operational scenario. First through hybrid simulations, and finally through a real field experiment, we have shown that our architecture is relevant to control a team of heterogeneous autonomous vehicles and to trigger repair strategies in case of disturbances.

9 Discussion

9.1 Related work

A lot of works in multi-robot distributed decision have used the paradigm of Contract Net Protocols (CNP, Smith 1980), e.g. (Rooker and Birk 2007; Atay and Bayazit 2007). Decision is taken locally by the agents, depending on the other agents they are able to communicate with, and on a set of tasks that have to be done. From a general point of view, these CNP-based architectures behaves similarly to multi-agent architectures such as IDEA (Muscettola et al. 2002) or BIIMAPS (Sotzing et al. 2008), where agents can share goals, but each agent is anyway responsible of taking decision for its own mission. On the similar way, Belbachir et al. (2012) has extended the T-Rex architecture (McGann et al. 2008), by defining a reactor that explicitly manages cooperation tasks, and exchange both goals and data between robots. The ALLIANCE project (Parker 1998) proposes a local reactive architecture: each robot can react to events and failures on its own, by applying prioritized behaviors that are conditioned by the current states and observations. MOOS-IvP (Benjamin et al. 2010) proposes a similar paradigm, where behaviors are selected based on conditions, and on criteria evaluated through Interval Programming. MOOS-IvP has been largely applied to (sub)marine robots. While a kind of hierarchical management has been introduced in (Schneider and Schmidt 2010) to manage multi-robot behaviors, these reactive, or behavior-based, architectures do not provide a global decision function. As seen in the requirements, this global decision layer is needed, first prior to the mission (with the production of an initial plan), and then during the mission execution, where repairs must follow the hierarchical structure of the plan.

Another related approach can be cited: the NVL language (Marques et al. 2015) used on top of DUNE (Pinto et al. 2013a). This language allows to define procedures so that the NVL interpreter selects the vehicles to use for each task based on required abilities. It focuses on clear parallelism between tasks, rendez-vous and synchronizations with deadlines. Our approach is more focused on automated plan generation based on higher-level hierarchical procedures, along with autonomous plan repair.

Gancet et al. (2005) has extended the LAAS architecture (Alami et al. 1998), for a multi-robot context. In his architecture, goals are hierarchized, using an HTN representation. While low-level goals are locally managed by each robot, high-level goals are however managed by a central decision node. Failure management, in particular when no communication is possible, is then only partially covered.

The BEAR architecture (Luzeaux and Dalgalarrondo 2001; Vidal et al. 2002) proposes a hierarchical paradigm, where the tasks to be accomplished by the robots are managed at several levels of abstraction. Failures are consequently managed at different levels, depending on their type, leading to a kind of hierarchical and local repair strategy, like HiDeN does. However, the overall structure of the plan is not explicited, nor the decomposition relation between tasks of several levels.

In the ASyMTRe approach (Zhang and Parker 2010), such decomposition is based on a data-flow model of the interaction between robots, then making explicit the moment at which a communication between agents will be mandatory. The notion of leaders and followers of (Chaimowicz et al. 2001; Parker et al. 2004) results in the same need for

local communications (in space and time) to accomplish joint tasks.

Two architectures have however a lot of similarities with HiDDeN, regarding its hierarchical execution and repair scheme based on global HTN plans. The Retsina architecture (Paolucci et al. 2000) uses HTN to decompose a global plan into elementary tasks, but these tasks are then scheduled, distributed, and possibly repaired independently of the hierarchical structure. It is then hard to follow the execution status of the global plan on-line, as this execution may not be consistent with a global hierarchical scheme. These drawbacks are crippling for representing the human expertise, which is expressed with hierarchical task decomposition, and for giving the complete plan as a feedback to the operators.

HOTRiDE (Fazil Ayan et al. 2007) proposes an execution and repair scheme for HTN that uses both the hierarchical structure of the plan, but also causal links between tasks. While this repair process goes further than HiDDeN, it has not been applied to a multi-robot context; in such context, reasoning on causal links may need some complementary knowledge that require communication capabilities. We have then decided to focus on a pure hierarchical repair process, as the hierarchical structure follows a team structure (i.e., a child task involves less robots than the father task), which is appropriate when communication are uncertain.

9.2 Conclusion

In this paper, we have presented HiDDeN, a multi-layer deliberative architecture for managing mission execution for multi-robot systems. The HiDDeN architecture has fulfilled the requirements coming from the air-sea scenario constraints. First, a planner produces offline a **hierarchical initial plan** that ensures the operational constraints raised by the military operators. The execution of this plan is then fully **distributed**: on each robot, a HiDDeN supervisor manages the execution of a local iHTN, that contains the robot actions and synchronization tasks with other robots. This supervisor can also detect several kind of **failures**, that are **managed locally** using a hierarchical repair process. When needed, this process is completed by communication recovery schemes to make a joint repair with one or several teammates. Finally, we have shown that the HiDDeN architecture is **scalable**. On one hand, simulations have shown that HiDDeN, and its hierarchical repair process, scales well with the complexity of the mission, including the number of failures. On the other hand, we have implemented several repairing HTN procedures to adapt to several failures (mine discovery, communication failure, boat discovery – the last one has not been discussed in this paper by lack of place). We have also conducted experiments for a second scenario, involving an AUV, an AAV and an ASV, looking for the wreck of a chemical tanker; in this scenario we easily adapted the HiDDeN architecture to the new type of vehicle, as well as the new actions of the other vehicles. HiDDeN has indeed been designed with the sake of not interfering too much with the control architecture of a robot, making the integration easier.

The experiments conducted on the air-sea scenario have highlighted all the difficulties that may arise when evaluating multi-robot missions on the field. The vehicles have evolved in environments that are particularly constraining. They are even more demanding when performing large-scale or long-term autonomy missions. In such experiments, the challenge is as much on the technical and scientific points as on the logistics. In such context, we took a huge benefit in making hybrid simulations prior to the experiment. Hybrid simulations indeed allowed us to test separately each vehicle architecture, while making experiments on the field. Only part of the logistics difficulties were present, while all the technical developments have been tested. Hybrid simulations have became the way-to-go for preparing multi-robot experiments in our lab.

Acknowledgments The authors thanks the French Defense procurement agency DGA for the funding of the ACTION project and for their strong involvement in the air-sea scenarios with developments and implementation on Daurade AUV and Spartan USV and for provision of the test site.

References

Alami, R., Chatila, R., Fleury, S., Ghallab, M., & Ingrand, F. (1998). An architecture for autonomy. *International Journal of Robotics Research, Special Issue on Integrated Architectures for Robot Control and Programming (IJRR)*, *17*(4), 315–337.

Antonelli, A.M.G., Aguiar, A.P., & Pascoal, A. (2012). A new approach to multi-robot harbour patrolling: theory and experiments. In *International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal

Atay, N., & Bayazit, O. (2007). *Emergent task allocation for mobile robots*. In *International sympositum on robotics: Science and systems (RSS)*, Atlanta, GA, USA

Barbier, M., Barrouil, C., Gabard, J.F., & Zanon, G. (2006). ProCoSA: a Petri net based software package for autonomous system supervision. In *International conference on application and theory of petri nets and other models of concurrency (ATPN)*, Turku, Finland

Barbier, M., Cao, H., Lacroix, S., Lesire, C., Teichteil-Königsbuch, F., & Tessier, C. (2009). Decision issues for multiple heterogeneous vehicles in uncertain environments. In *National conference on control architectures of robots (CAR)*, Toulouse, France

Barbier, M., Gabard, J. F., Bertholom, A., & Dupas, Y. (2011). An onboard software decisional architecture for Rapid Environmental Assessment missions. In *IFAC world congress*, Milan, Italy

Belbachir, A., Ingrand, F., & Lacroix, S. (2012). A cooperative architecture for target localization using multiple AUVs. *Intelligent service robotics* (Vol. 5(2)).

Benjamin, M. R., Schmidt, H., Newman, P. M., & Leonard, J. J. (2010). Nested autnomy for unmanned marine vehicles with MOOS-IvP. *Journal of Field Robotics (JFR)*, *27*(6), 834–875.

Brinon-Arranz, L., Pascoal, A., & Aguiar, P. (2014) Adaptive leader-follower formation control of autonomous marine vehicles. In *IEEE conference on decision and control*, Los Angeles, CA, USA

Chaimowicz, L., Sugar, T., Kumar, V., & Campos, M. (2001). An architecture for tightly coupled multi-robot cooperation. In *International conference on robotics and automation (ICRA)*, Seoul, Korea

Chanel, C.C., Teichteil-Königsbuch, F., & Lesire, C. (2013). Multi-target detection and recognition by UAVs Using Online POMDPs. In *AAAI conference on artificial intelligence (AAAI)*, Bellevue, WA, USA.

Chatzichristofis, S., Kapoutsis, A., Kosmatopoulos, E. B., Doitsidis, L., Rovas, D., & Borges de Sousa, J. (2012). The NOPTILUS project: Autonomous multi-AUV navigation for exploration of unknown environments. In *IFAC workshop on navigation, guidance and control of underwater vehicles (NGCUV)*, Porto, Portugal

Das, J., Harvey, J., Py, F., Vathsangam, H., Graham, R., Rajan, K., & Sukhatme, G.S. (2013). Hierarchical probabilistic regression for AUV-based adaptative sampling of marine phenomena. In *International conference on robotics and automation (ICRA)*, Karlsruhe, Germany

De Cubber, G., Doroftei, D., Baudoin, Y., Serrano, D., Chintamani, K., Sabino, R., & Ourevitch, S. (2012). ICARUS: An EU-FP7 project Providing Unmanned Search and Rescue Tools. In *IROS workshop on robots and sensors integration in future rescue INformation system (ROSIN)*, Vilamoura-Algarve, Portugal

Echeverria, G., Lemaignan, S., Degroote, A., Lacroix, S., Karg, M., Koch, P., Lesire, C., & Stinckwich, S. (2012) Simulating complex robotic scenarios with morse. In *International conference on simulation, modeling, and programming for autonomous robots (SIMPAR)*, Tsukuba, Japan

Erol, K., Hendler, J., & Nau, D. (1994). HTN planning: complexity and expressivity. In *AAAI conference on artificial intelligence (AAAI)*, Seattle, WA, USA

Fazil Ayan, N., Kuter, U., Yaman, F., & Goldman, R. (2007). HOTRiDE: hierarchical ordered task replanning in dynamic environments. In *International conference on automated planning and scheduling (ICAPS)*, Providence, RI, USA

Ferri, G., & Djapic, V. (2013). Adaptive mission planning for cooperative autonomous maritime vehicles. In *International conference on robotics and automation (ICRA)*, Karlsruhe, Germany

Gancet, J., Hattenberger, G., Alami, R., & Lacroix, S. (2005). Task Planning and control for a multi-AUV system: architecture and algorithms. In *International conference on intelligent robots and systems (IROS)*, Edmonton, AB, Canada

Gateau, T., Lesire, C., & Barbier, M. (2012). Knowledge base for planning, execution and plan repair. In *International conference on automated planning and scheduling (ICAPS) workshop on planning and execution (PlanEx)*, Altibaia, Brasil

Gateau, T., Lesire, C., & Barbier, M. (2013). HiDDeN: Cooperative plan execution and repair for heterogeneous robots in dynamic environments. In *International conference on intelligent robots and systems (IROS)*, Tokyo, Japan

Hollinger, G.A., Englot, B., Hover, F., Mitra, U., & Sukhatme, G.S. (2012). Uncertainty-driven view planning for underwater inspection. In *International conference on robotics and automation (ICRA)*, Saint-Paul, MN, USA

Kurowski, M., Korte, H., & Lampe, B.P. (2012). Search-and-rescue-operation with an autonomously acting rescue boat. In *International conference on autonomous and intelligent systems*, Aveiro, Portugal

Luzeaux, D., & Dalgalarrondo, A. (2001). HARPIC, an Hybrid Architecture Based on Representations, Perception, and Intelligent Control: A Way to Provide Autonomy to Robots. In *The international conference on computational science (ICCS)*, London, UK

Marques, E., Ribeiro, M., Pinto, J., Sousa, J., & Martins, F. (2015). *Towards programmable coordination of unmanned vehicle networks*. In *IFAC workshop on navigation, guidance and control of underwater vehicles (NGCUV)*, Girona, Spain

McGann, C., Py, F., Rajan, K., Thomas, H., Henthorn, R., & McEwen, R. (2008). A deliberative architecture for AUV control. In *International conference on robotics and automation (ICRA)*, Pasadena, CA, USA

Mukhopadhyay, S., Wang, C., Bradshaw, S., Bazie, V., Maxon, S., Hicks, L., Patterson, M., & Zhang, F. (2012). Controller performance of marine robots in reminiscent oil surveys. In *International conference on intelligent robots and systems (IROS)*, Vilamoura, Portugal

Muscettola, N., Dorais, G., Fry, C., Levinson, R., & Plaunt, C. (2002). IDEA: Planning at the core of autonomous reactive agents. In *International NASA workshop on planning and scheduling for space*, Houston, TX, USA

Nau, D., Au, T. C., Ilhami, O., Kuter, U., Murdock, W., Wu, D., et al. (2003). SHOP2: an HTN planning system. *Journal of Artificial Intelligence Research (JAIR)*, *20*, 379–404.

Paolucci, M., Shehory, O., & Sycara, K. (2000). Interleaving planning and execution in a multiagent team planning environment. *Linköping Electronic Articles in Computer and Information Sciences*, *5*(18),

Parker, L. (1998). ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation. *IEEE Transactions on Robotics and Automation*, *14*, 220–240.

Parker, L., Kannan, B., Tang, F., & Bailey, M. (2004). Tightly-coupled navigation assistance in heterogeneous multi-robot teams. In *International conference on intelligent robots and systems (IROS)*, Sendai, Japan

Pinto, J., Dias, P., Martins, R., Fortuna, J., Marques, E., & Sousa, J. (2013a). The LSTS toolchain for networked vehicle systems. In *OCEANS*, San Diego, CA, USA

Pinto, J., Faria, M., Fortuna, J., Martins, R., Sousa, J., Queiroz, N., et al. (2013b). Chasing fish: Tracking and control in a autonomous multi-vehicle real-world experiment. In *OCEANS*, San Diego, CA, USA

Rooker, M., & Birk, A. (2007). Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, *15*(4), 435–445.

Schneider, T., & Schmidt, H. (2010). Unified command and control for heterogeneous marine sensor networks. *Journal of Field Robotics (JFR)*, *27*(6), 876–889.

Shkuri, F., Xu, A., Meghjani, M., Higuera, J.C.G., Girdhar, Y., Gigure, P., Dey, B.B., Li, J., Kalmbach, A., Prahacs, C., Turgeon, K., Rekleitis, I., & Dudek, G. (2012). Multi-domain monitoring of marine environments using a heterogeneous robot team. In *International conference on intelligent robots and systems (IROS)*, Vilamoura, Portugal

Smith, R. (1980). The Contract Net Protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, *29*, 12.

Soetens, P., & Bruyninckx, H. (2005). Realtime Hybrid Task-Based Control for Robots and Machine Tools. In *International conference on robotics and automation (ICRA)*, Barcelona, Spain

Sotzing, C., Johnson, N., & Lane, D. (2008). Improving multi-AUV coordination with hierarchical blackboard-based plan representation. In *Workshop of the UK planning and scheduling special interest group (PlanSIG)*, Edinburgh, UK

Vidal, R., Shakernia, O., Kim, H., Shim, D., & Sastry, S. (2002). Probabilistic pursuit-evasion games: Theory, implementation and experimental evaluation. *IEEE Transactions on Robotics and Automation*, *18*, 662–669.

Watanabe, Y., Lesire, C., Piquereau, A., Fabiani, P., Sanfourche, M., & Le Besnerais, G. (2010). The Onera ReSSAC unmanned autonomous helicopter: Visual air-to-ground target tracking in an urban environment. In *AHS Forum*, Phoenix, AZ, USA

Xiang, X., Jouvencel, B., & Parodi, O. (2010). Coordinated formation control of multiple autnomous underwater vehicles for pipeline inspection. *Journal of Artificial Intelligence Research (JAIR)*, *7*(1), 75–84.

Zhang, Y., & Parker, L.E (2010). IQ-ASyMTRe: Synthesizing coalition formation and execution for tightly-coupled multirobot tasks with sensor constraints. In *International conference on intelligent robots and systems (IROS)*, Taipei, Taiwan

Charles Lesire passed his Ph.D. thesis on Computer Sciences at the University of Toulouse, France, in 2006. In 2007, he was a post-doctoral fellow at LAAS-CNRS, in the robotics team. Since 2008, he has been a research fellow at ONERA, in the System Control department. His main topics of interest include software engineering for robotics, embedded decision-making, and cooperation of heterogeneous robots. He has participated to many experiments on the field, involving marine, ground, and aerial robots.

Guillaume Infantes graduated from ENSEEIHT (a french national engineering school, delivering master's degree in computer science and applied mathematics) in 2002, obtained a master's degree in AI in 2002 and another one in Computing Systems in 2003. He received a Ph.D. from University of Toulouse in 2006, conducting research on automated learning and decision making for autonomous robotics. After a year as a research assistant at UMIACS (University of Maryland Institute for Advanced Computer Studies) he joined ONERA (the French Aerospace Lab) in 2008. His research activities are related to decision making under uncertainty, from modelling to problem solving algorithms, also using machine learning approaches.

Thibault Gateau earned his Ph.D. in 2012 in artificial intelligence at ONERA, "The French Aerospace Lab". He is currently a post-doctoral researcher at the Human Factors and Neuroergonomics team at ISAE-Supaéro (Higher Institute of Aeronautics and Space) in Toulouse, France. His research interests include autonomous vehicles, planning, embedded architectures, multi-agent systems, autonomous decision-making, neuroergonomics, adaptive interactions, human-machine interfaces, and brain-computer interfaces.

Magali Barbier received her Ph.D. in Automatic in 1990. She works on unmanned vehicles since 1998. Research on the autonomy of Autonomous Unmanned Vehicles led in 2006 to the sea validation of an embedded software decisional architecture for hydrographic and oceanographic surveys on a vehicle operated by the French Government Defense procurement agency; advanced research continues on supervision and planning functions for this architecture. From 2007, she was responsible for the ACTION Prospective Research Programme which led in 2015 to the scientific demonstration of the cooperation of 12 autonomous air and ground vehicles with 8 real vehicles and 4 simulated vehicles in the loop. A software decisional architecture plugged on-board existing individual decisional architectures allowed the vehicles team to perform its patrolling mission in an autonomous way while adapting to perturbing events.