

# Thetis, an innovative tool for hybrid simulation of heterogeneous marine vehicles

O. Parodi

LIRMM - University of Montpellier 2 - CNRS

161 rue Ada, 34392 Montpellier, France

{parodi}@lirmm.fr

**Abstract**—Simulation is a necessary step to allow the experiment of new multi-vehicle scenarios. In this context, new tools are needed to test and validate the feasibility of the mission, the local and temporal behavior of the vehicles' controllers and the efficiency of the algorithms used. That's why we propose *Thetis*, a real-time multi-vehicles hybrid simulator for heterogeneous vehicles. This simulator allows *Hardware In Loop* (HIL) simulations including virtual sensors which allow to provide a representation of a virtual world, and including the support of communication devices. The architecture of this simulator is conceived so that it ensures a temporal decoupling between the virtual environment, the vehicles, sensors and communication simulators and, of course, the actual embedded controller which warrants us the temporal consistency of the results.

Our contribution concerns the proposed simulator architecture. This architecture gathers all the important features required to simulate a flotilla including communications skills. It is designed to facilitate collaborative work between the different actors involved in this type of scenario.

This paper presents the architecture and functionalities of *Thetis* and present some simulation results with the support of communications.

## I. INTRODUCTION

The use of an autonomous vehicle for marine environment brings an undeniable flexibility compared to the tele-operated devices. The immediate consequence of this flexibility is the level of sophistication required by the controller of the vehicle. Moreover, this level of autonomy is particularly high for marine environment which presents several differences from the air one leading to major difficulties. Indeed, the impermeability of the environment to GPS signals, makes currently difficult the accurate positioning of a vehicle. In addition, communications are only possible via an acoustic link. This results in low data rates which are variable in time and in space. In this context, the real-time control architecture COTAMA (Contextual Task Management Architecture) [4] has been developed to manage such missions, to apply decisions relative to the context of the robot, while allowing to dynamically manage the conflicts related to the use of acoustic communications and sensors.

If a vehicle presents a proved interest, the use of multiple robots, potentially different, allows to improve the quality of acquisitions (multi-scale, multi-sensor perception, spatio-temporal perception, redundancy...) and to improve the velocity with which they are made. The deployment of multi-sensor platforms understood that it is necessary to implement controls and strategies of control for coordinated flotilla.

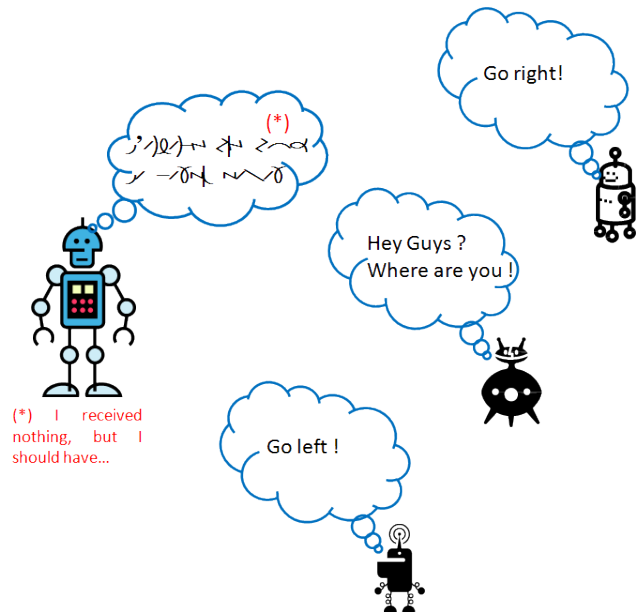


Fig. 1. Difficulties encountered in multi-vehicle scenarios

The localization and communication difficulties previously mentioned make readily the task difficult (Fig. 1).

Therefore, it is no longer conceivable to run experiments involving a fleet of autonomous vehicles, without having first tested the feasibility of the mission, the logical and temporal behavior of the robots' controllers, the effectiveness of algorithms used and the proper functioning of all subsystems. This is all more true when the vehicles are heterogeneous, come from different organizations or institutions and are involved all together for a common mission.

The complexity of the control architecture on the one hand, and the difficulties raised by the choice of multi-vehicle control strategies on the other hand, require the creation of new simulation tools to test and validate control laws, while detecting preliminary inconsistencies of envisaged scenarios. In this context, we propose a new collaborative tool called *Thetis*.

## II. THETIS: HYBRID SIMULATOR

*Thetis* is a new real-time hybrid simulator allowing the consideration of heterogeneous multi-vehicles scenarios, including communication restrictions. A quick overview on the

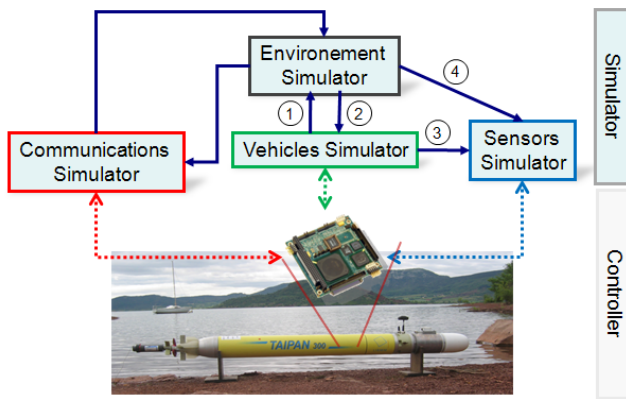


Fig. 2. Simplified architecture of Thetis: there is a logical sequence between 3 simulators even if there are independent processes on different computers. The cycle duration of this sequence must be largely lower than the period of the controller. The temporal decoupling is only effective (and that's enough) between the simulator and the controllers. The link between the communications simulator and the environment simulator is event-driven (when a communication between vehicles occurs).

overall architecture of the simulator is presented on figure 2.

### A. Critical Concepts

This simulator is composed of a set of mechanisms which respects several properties. One of the most important is the capability of the simulator to ensure a *temporal decoupling* between the control loop and the simulation loop. This warrants the coherence between simulation results and expected real behavior. For example, if for any reason, the onboard computer is unable to provide the actuator with commands at an appropriate time (unexpected delay), then the simulator has to exhibit the natural consequence of this delay on the vehicle behavior (open loop). Indeed, in Thetis, there is *no logical synchronization* between the simulation loop and the controller computation onboard the robots. Another important concept is the *upgradeability* of the simulator. We are able to add some new components (sensors, vehicles...) in a very easy way. Indeed, the modularity of this simulator favors the interventions of different specialists. Thus, a sonar specialist will be able to modify the sonar model, without considering the global simulator. Finally, a very interesting aspect of this architecture is its *portability*. Indeed it is able to work with different robots and thus be connected with different control software architectures. We only need to adapt the interface between the 2 systems (simulator and controller), in order to make them work together. Moreover, the distributed aspect of Thetis affords the faculty of the simulator to be divided and executed on different computers, linked on a dedicated local area network via UDP/IP protocol. This avoids overloading computers and affecting the real time capability of the systems.

The main features of *Thetis* are summarized figure 3.

### B. Implementation and Technical Considerations

All the concepts exposed in the previous section are supported by using 3 mechanisms and the architecture is

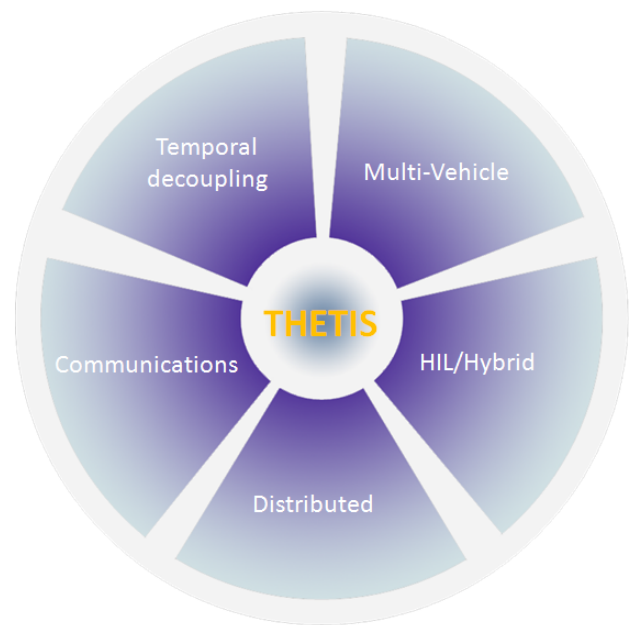


Fig. 3. Critical concepts of *Thetis*

based on four simulators. First an XML-based specifications exchange (XML for eXtensible Markup Language) allows to structure the parameters of the different models (modem, radio, fins, motor...), and configuration files, while promoting the modularity and the portability. Indeed modularity is obtained using the XML format, which allows to specify the components parameters (response-time, accuracy, rate ...), and to modify them in a very easy way. Now, the replacement of a sensor by another is done by calling a XML file in place of the other. All the system components description follows the same idea (actuators, sensors, body dynamics...). Moreover, many components could use the same formalism to describe their intrinsic parameters without using all of them depending of the model used. The validation and extensible properties of the XML language make it an ideal base to enrich the model parameters files. Thus the robots components used in this simulator are described in XML formalism (fig. 4).

Then portability and temporal decoupling are favored by using sockets and local shared memories widely. Thus it is possible to run several processes on different computers, each of them interacting with others using these mechanisms.

In order to ensure real-time performances as well as effective temporal decoupling, the overall simulation system is in fact an application divided into four simulators. The first one is a vehicle simulator which allows the simulation of the robots dynamics. The second one is a sensors simulator allowing the simulation of the various sensors of the robots. The third one is a communications simulator in charge of mimicking the behavior of communications device (signal processing for an acoustic modem, for example). Lastly the fourth one is a stationary environment simulator. It allows the simulation of exteroceptive sensors, it is in charge of

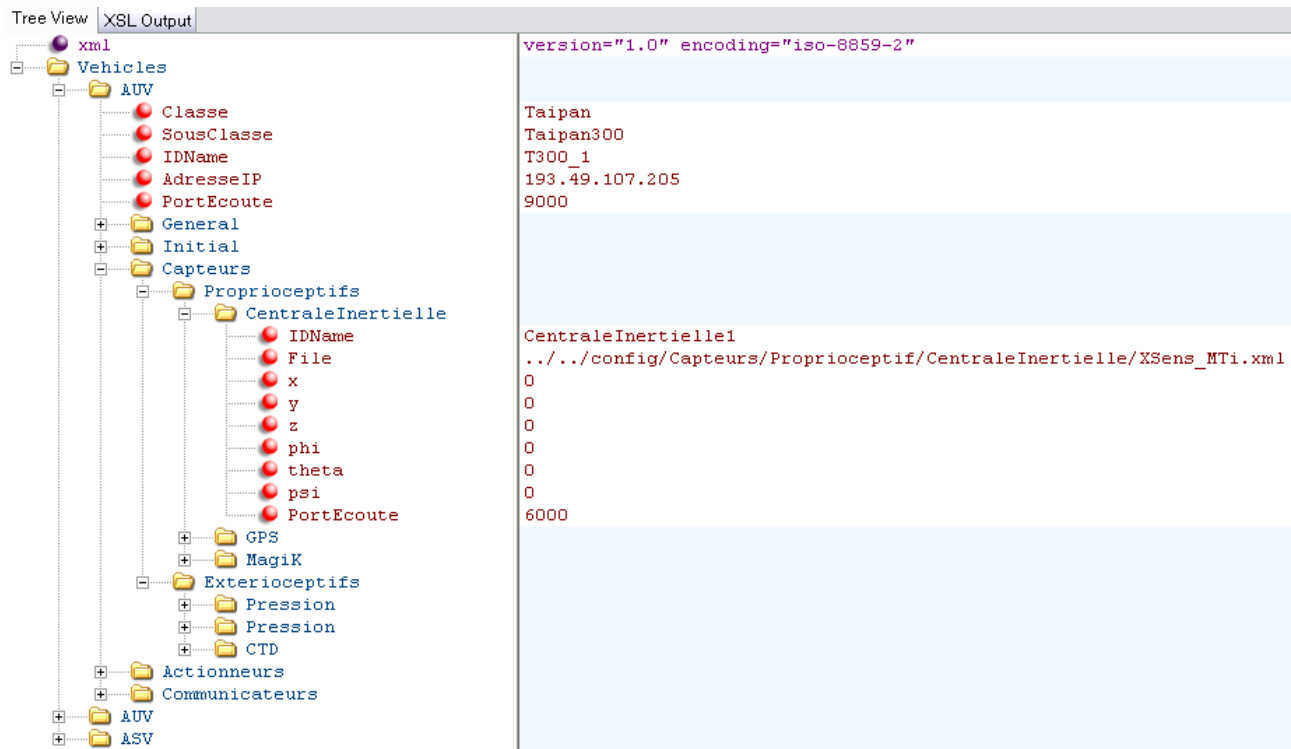


Fig. 4. XML formalism used to describe the robots

computing the possible collisions and of computing the different acoustic signals propagation, the latency and distortions of the communication signals. All these simulators are interconnected on a dedicated UDP/IP network. This avoids overloading the network in order to guarantee real-time performances and to avoid packets losses (potentially possible with UDP/IP network). The connections between these blocks are detailed and explained on figure 2. Only the sensors, the communications and vehicles simulators are connected to the real robot, the environment simulator being connected only to the 3 others. All these simulators work under Linux RTAI. Information about network configuration is described in a shared XML file. All the XML files describing the components of the system are loaded at the initialization and thus allow to instantiate the different objects of the simulator.

Finally we have created a set of libraries containing a set of classes enabling us to build the various objects of the simulation system. All these classes are documented with Doxygen tool [3].

### C. Components of the simulator

Each of the 4 simulators composing the Thetis system, is itself composed of several independent processes (12 in total). The structures of these 4 simulators are quite similar. Indeed, each of them is composed of one or more independent process(es) (called *Dispatcher*) in charge of IPC (Inter Process Communication), decoding and writing the data exchanged by the simulators, to a local shared memory initially created by the Dispatcher processes. For

each simulator, there is one main process which is in charge of the models evolution. This structure, preferentially executed on a dual (or quad) core processor, prevents the main processor from being disturbed during the inter-simulators communications. We will not detail the operation for each of these components since it has been described in [2].

1) *Vehicles Simulator*: The vehicles simulator is in charge of computing the robots dynamic evolution. We present this simulator structure on figure 5. The robots send actuators commands calculated by the onboard computer to the simulator through UDP socket. The *Dynamic Models* represented on Figure 5 compute all the forces and torques applied to the robots in order to determine the vehicles accelerations and, after integration steps, vehicles attitudes and velocities are computed. The computed data are sent to the environment simulator in order to verify the absence of collisions and if necessary to correct positions. Afterwards, the computed data are sent to the sensors simulator.

2) *Sensors Simulator*: The sensors simulator is in charge of providing the real robot (onboard computer) with virtual data from the simulation. The *Reception & decoding process* is in charge of listening to messages from the environment simulator (parts of maps determined according to the position and the range of the robot's sensors). The outputs of the *Sensors Models* are computed according to each sensor model (including noise), to the data from the vehicles simulator (systems state), and at last to the data from the environment simulation. Once these outputs are computed, they are specifically sent to the concerned robots with the

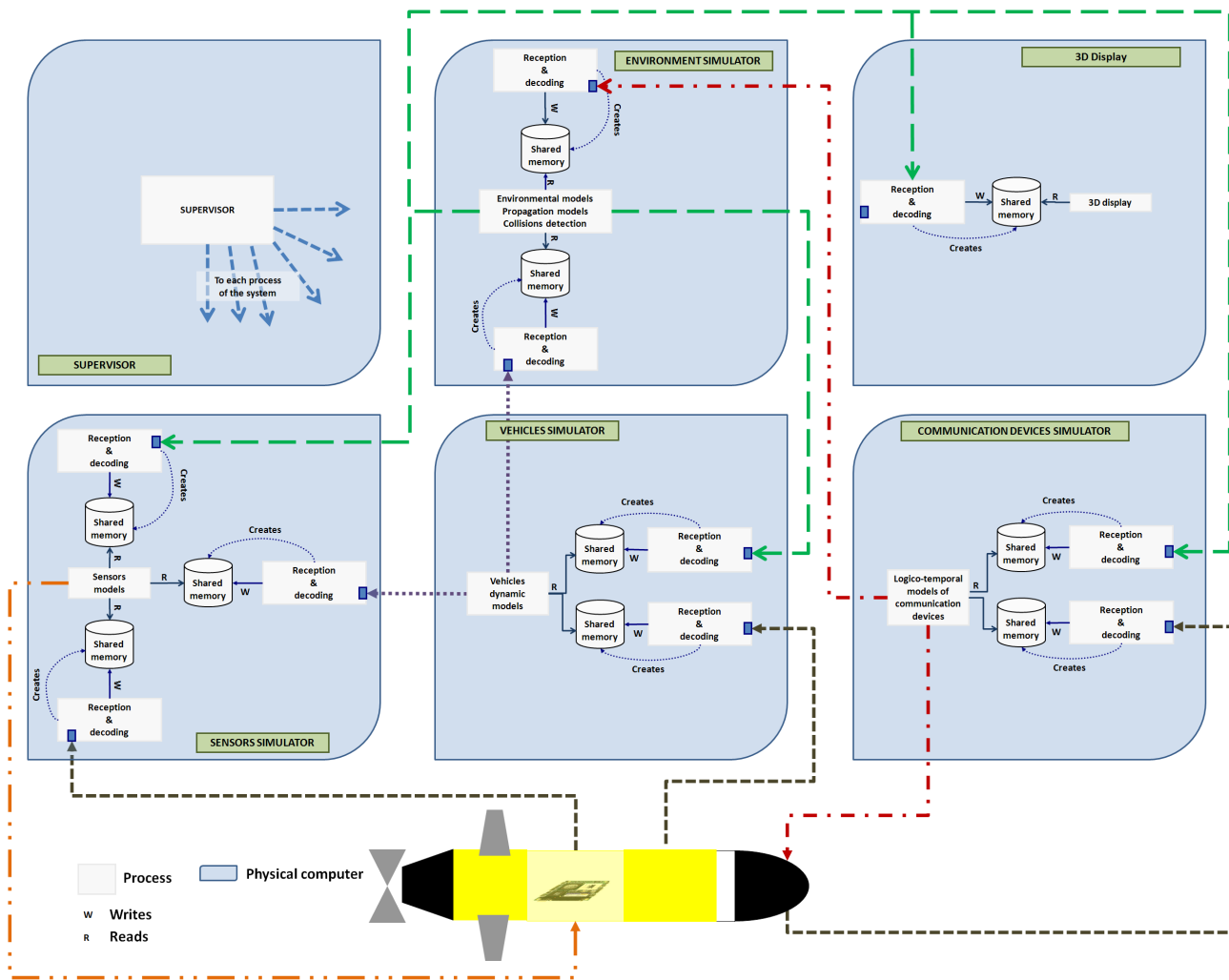


Fig. 5. Detailed view of Thetis. Dashed lines stand for the UDP/IP inter-simulators communications.

same refresh rate as the real sensor.

The proprioceptive sensors are used to acquire an estimation of the state variables of the robot, and its derivative. These sensors are modeled here by using directly the variables produced by the vehicles simulator. Afterwards the sensor simulator provides the samples at the same frequency as the real sensor, taking care to limit its range and adjust its resolution. It is also possible to add noise so as to obtain a more realistic simulation.

3) *Environment Simulator*: The environment simulator (figure 5) is in charge of providing the stored geophysical maps around a given 3D geographical point (Temperature, Salinity, Local current, Plankton density ...), computing signal propagation (when communications between vehicles occur), signal distortion and latency, and detecting collisions between vehicles and ground. At each cycle of the vehicles simulator, the environment simulator is called and sends back the value of the local current and the occurrence of a collision. It has to be noted that the computed communication signals are only sent to the communications simulator when (considering latency and bandwidth) and if necessary (if the

vehicles are not within range of communications no messages will be delivered). Moreover the rate of communication may vary according to the quality of the acoustical channel. When a communication occurs in this area, other communications using the same frequency potentially damage the communication channel, and the messages distribution is disturbed.

The environment is modeled by using different elements: the topography, the temperature and salinity distribution and the environmental disturbances (currents). Presently, only the topography, the temperature, salinity and currents distribution are implemented.

4) *Communications Simulator*: The mechanism of the communication simulator (figure 5) takes into account the delay, the rate and the losses caused by the type of communication device in use, and the propagation medium. It has to be noted that coordinated control of AUVs flotilla is intrinsically dependent on the communications performances, which cannot be guaranteed. Thus a simulator able to perform multi-vehicle simulation has to consider these aspects explicitly. The communications simulator is in charge of simulating the



Fig. 6. Taipan 300 by the Salagou Lake in France

signal processing done by the communication devices. When a vehicle has to emit a message (radio or acoustic waves), it sends a request to its concerned communication chosen device (radio, wifi, acoustic modem...). In the simulation case, these data are not sent to the physical communication device, but routed to the communications simulator (see [1] for more information about the connectivity of the simulator with our AUVs).

We have started with a model developed for the propagation of sound waves in the aquatic environment, since the main subject of our team is underwater robotics. We have created a propagation model which takes into account several physical phenomena which represent the constraints that we actually meet in our experimentations. We mainly model the transmission losses (absorption and dispersion) and the sea relative noise level (see [2] for all the details of this model).

### III. CONNECTIVITY

We have 2 AUVs which are currently being upgraded. The first one is Taipan 300; this is a small AUV which is designed for very shallow water operations. It is 193cm long for a diameter of 15cm and a weight of 32 Kg (figure 6). As displayed on Figure 6 the vehicle is fitted with a CTD (Conductivity, Temperature, Depth sensor located at the front of the vehicle just behind the white nose), but we are currently replacing this device by an acoustic modem (as we can see on figure 8). This vehicle is moving at a speed of about 2m/s and its autonomy is about 3h. This AUV can reach a depth of 100m. We have developed a very useful software architecture for this AUV and the connectivity of this architecture with our simulator is presented in [1]. This kind of simulator is fully useful only if the connection with the robot and its control architecture does not require any modification on the robot. Indeed, the transition from simulation to real experiments has to be as transparent as possible, otherwise the expected behavior will not be guaranteed. Thus the best way is, of course, to physically connect the AUV sensor devices to the output of the sensors simulator.

In order to do this, the sensors simulator must be able to reproduce the electrical signals of each sensor. But this solution is very difficult to implement. Indeed, often, constructors do not provide complete information about the sensors internal specifications. Hence, another solution has to be implemented, for which the code modification has to be as

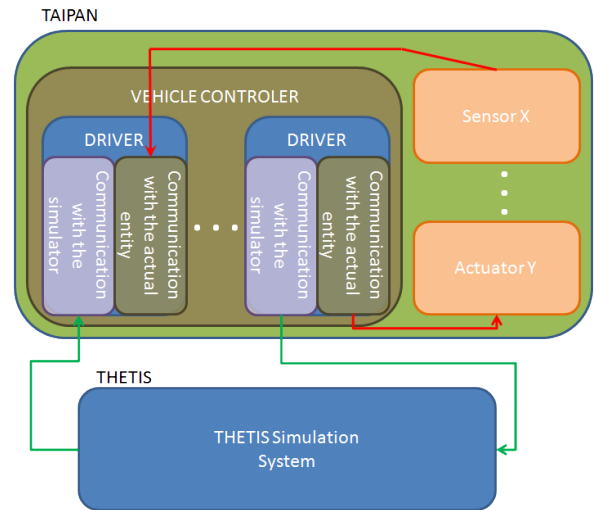


Fig. 7. Connection with the simulator

light as possible, in order to avoid inducing specific behavior of the control architecture. Thus, from the robot's controller side, it is necessary to implement a modular architecture, in order to modify only the data-supply-mechanism of the control architecture, shunting the real sensors. For its part, the sensors simulator has to provide the sensors data, with the same updating rate, range, errors, noise etc... as the real ones. In this context, we stay close enough to reality, in order to validate our control architecture. In order to obtain a realistic behavior during simulation, we have to design the interactions between the modules of the software architecture. The only difference lies at the driver level: the driver can be provided with real data or data from the simulator. In the same way, the commands sent to the actuators are sent to the simulator instead (figure 7).

On Figure 8 we can see the Thetis system linked to our AUV T300. On the photo we can see 4 computers (P4HT@2.7GHz + 1024Mo DDR) (linked to a dedicated LAN via a switch), each of them running a simulator. These 4 computers boot on a USB key enabling us to deploy our simulators anywhere. Moreover on the photo we can see a fifth computer (laptop) which enables us to upload the configuration files and the executables update on the network, to monitor the simulators and finally to manage the launch sequence of the system. All the log files (also written in XML) are uploaded from these simulators to the supervision laptop and then analyzed using a matlab XMLToolbox.

### IV. EXEMPLE OF MISSION SIMULATED WITH THETIS

The simulation presented in this section involves five Taipan-class AUVs. Each of them are configured to carry out an acoustic modem (ORCA TAPAC), enabling them to communicate to about 20 bit/s. At the initial time, all vehicle stay along an east-west axis and are separated by different distances (figure 9). The goal of the mission is to synchronize heading changes: at a given moment, an AUV decides to change its heading and sends to the flotilla the new heading

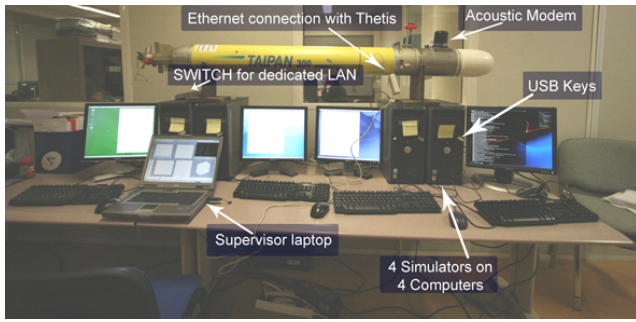


Fig. 8. The Thetis system and the AUV Taipan300

to be followed. *T300.1* and *T300.3* are chosen to change their direction at 73s and 75s respectively.

The instruction sent to change the heading is "Nouveau\_CAP\_XXX" where XXX is the new heading in degrees. The depth of the AUVs is 40m and they have to maintain this depth during the mission. The same mission is sent to all the vehicles: follow a constant heading at 40m with a constant speed (1.6 m/s). In order to be able to properly observe all the phenomena caused by the propagation of the messages, the speed of sound was divided by a factor ten.

The results of the simulation are presented figure 10. We can observe that the results are far from what we could have expected. First all the vehicles did not change their heading. *T300.2* and *T300.5* kept the same heading (0) during the simulation. Among the AUVs which changed their heading, it seems that there was no coordination: some of them follow the heading 120, others the heading 230.

Among the AUVs who follow the desired heading, the direction change is not done simultaneously with the other AUVs. Figure 9 presents a timeline on which is reported the content of the simulation results files. We can see the communication activities of the five AUVs: it appears the messages sent and received and the dates associated with these events. Furthermore, we have highlighted the communication errors:

- error A: protocol error. Indeed, *T300.3* sent a bad formatted message which did not be able to be parsed correctly by *T300.1* (Fig. 11).
- error B: interferences error. *T300.2* was receiving a message from *T300.1*, but a transmission from *T300.3* occurs during this reception. It results that the two messages are not intelligible by *T300.2* (Fig. 12).
- error C: media occupation error. *T300.3* was sending a message when a communication from *T300.1* occurred. Its acoustic antenna was requisitioned to emit the message and thus, the incoming message was not delivered (Fig. 13).
- error D,E,F: attenuation error. The level of the messages are too low to be heard by the AUVs (Fig. 14).

Finally, only *T300.4* receives a message well formatted and with a sufficient level, indicating to take heading 120. These errors perfectly explain the results presented figure 10: *T300.1* and *T300.3* change their heading according to the

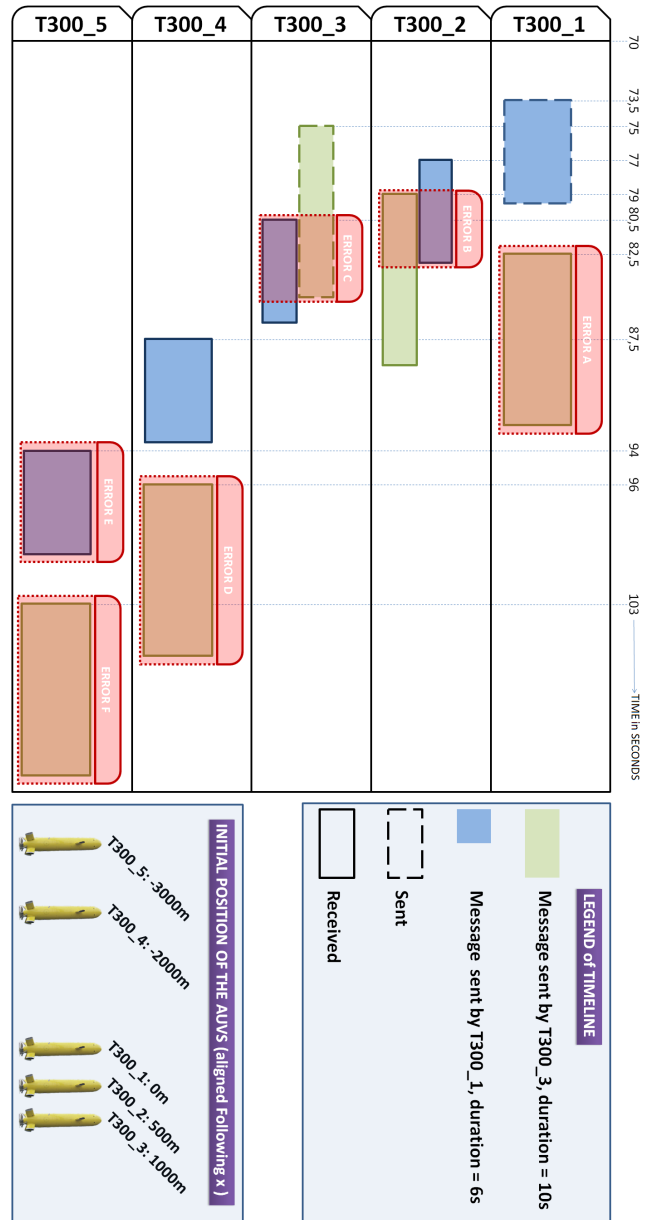


Fig. 9. Initial disposition of the vehicles, and communications diagram

mission file (120 and 230 respectively), *T300.2* and *T300.5* do not change their heading because of the above errors, and *T300.4* change its heading to 120.

This simulation can also highlight the delay due to constraints linked to the propagation medium: figure 10 shows clearly that *T300.4* does not change its heading immediately when *T300.1* takes this decision. Indeed it takes about 14s between the date of the emission of *T300.1* and the date of the effective change of heading of *T300.4*. This corresponds to the delay for the message to be sent (about 6s) and the propagation delay (about 2000m at 174m/s). For these reasons, *T300.4* kept the 0 heading for a longer time that we could have expected.

Through this simulation, the importance of a model of

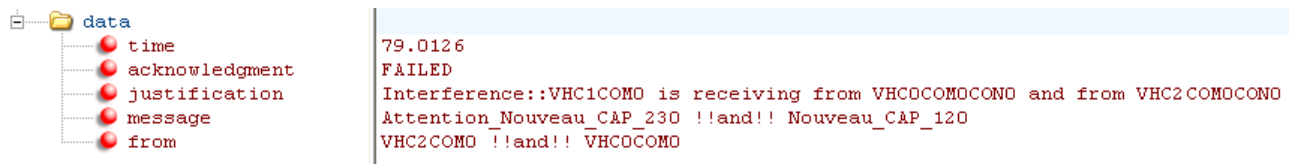


Fig. 12. Interferences error: two messages are received at the same time

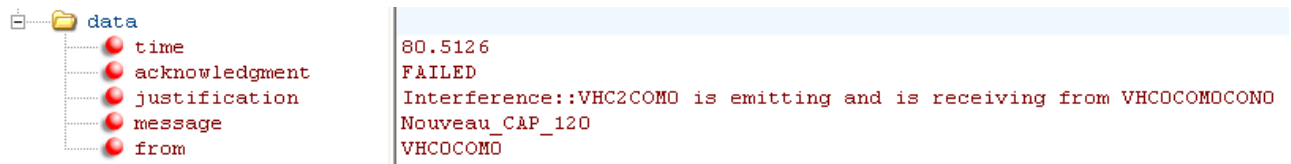


Fig. 13. Media occupation error: a message is sent while another one is being propagated



Fig. 14. Attenuation error: the level of the propagated messages is too low to be understood by the receiving AUVs

communication taking into account the latency, interferences, attenuation, and behavior of communication devices is a necessary requirement for the multi-vehicle cooperation context. Indeed, simulations run in simulators which do not take into account these parameters will be far away from reality and it will not be possible to validate solutions for multi-vehicle coordination.

## V. CONCLUSION

This HIL simulator plays an important role in the development of the controllers of robots, in the validation of cooperation strategies and in the protocol design between several teams. This architecture gathers all the important features that a simulator must include in order to allow simulation of

flotilla. The models used are maybe less accurate than the ones within traditional simulators, but the structure of our simulator allows an easy evolution. So we have designed and implemented a simulator architecture which guarantees the relevance of the results, the upgradability, the modularity and the portability. We have only evoked the use of this simulator in an underwater frame but it is generic enough to be used with other robots and thus it can allow us to imagine more complex scenarii like the coordination of AUVs and air drones in order to make coastal monitoring. *Thetis* is currently one of the only simulator able to take into account all of the necessary requirements for heterogeneous multi-vehicle scenarios (please refer to [7] for an exhaustive bibliography on the available simulators - until 2009 - able

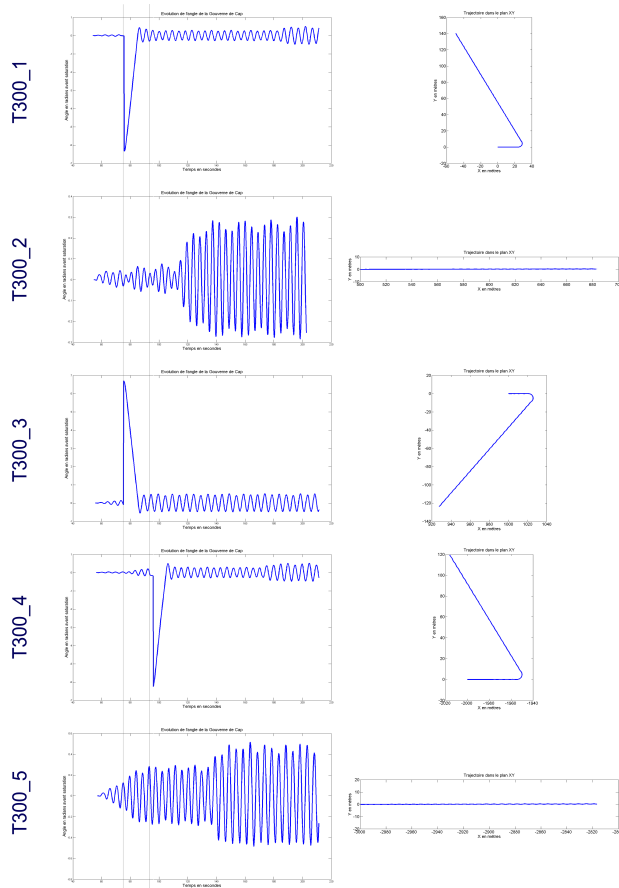


Fig. 10. Evolution of the heading fins angle and trajectory of the AUVs in the XY plan



Fig. 11. Protocol error: the expected message was "Nouveau\_CAP\_230"

to minimally take into account simulations involving marine crafts). Preliminary tests involving two Taipan-class AUVs (on which the COTAMA architecture was running) on the one hand, and one ASV (Autonomous Surface Vehicle) Charlie (developed by ISSIA-CNR in Italy) and on which another control architecture was running) on the other hand, have been done, proving the feasibility and the validity of our approach.

## REFERENCES

- [1] O. Parodi, A. El Jalaoui, D. Andreu Connectivity of Thetis, A Distributed Hybrid Simulator, with a mixed Control Architecture *The Fourth International Conference on Autonomic and Autonomous Systems ICAS*, 2008
- [2] O. Parodi, V. Creuze, B. Jouvencel Communications within Thetis, a Real Time Multi-vehicles Hybrid Simulator *The 18th International Offshore (Ocean) and Polar Engineering Conference*, 2008
- [3] <http://sourceforge.net/projects/doxygen/> accessed on 02/08
- [4] A. El Jalaoui, D. Andreu, B. Jouvencel Contextual Management of Tasks and Instrumentation within an AUV control software architecture *Conf. on Intelligent Robots and Systems (Iros06)*, 2006
- [5] O. Parodi, B. Jouvencel, L. Lapiere. Thetis : A real-time multi-vehicles hybrid simulator for heterogeneous vehicles *IROS08 : International Conference on Intelligent Robots and Systems, Workshop on robot simulators : Available Software, Scientific Applications and Future Trends*, Nice 2008
- [6] O. Parodi, B. Jouvencel, V. Creuze. Communications with thetis, a real time multivehicles hybrid simulator *ISOPE'08 : International Society offshore and Polar Engineers*, July 2008
- [7] O. Parodi. Simulation hybride pour la coordination de vehicules htrogenes au sein d'une flottille *Thse soutenue l'Universit de Montpellier 2*, Dc. 2008