Comparison Between Results Obtained with Thetis, a Real-Time Multi-Vehicles Hardware-in-the-Loop Simulator, and Results Obtained During Sea Trials

Olivier Parodi, Vincent Creuze, Bruno Jouvencel, Xianbo Xiang

Comparison between results obtained with *Thetis*, a real-time multi-vehicles hardware-in-the-loop simulator, and results obtained during sea trials

Olivier Parodi*, Vincent Creuze*, Bruno Jouvencel*, Xianbo Xiang*
*LIRMM - University of Montpellier 2 - CNRS
161 rue Ada
34392 Montpellier, France
{parodi,creuze,jouvencel,xiang}@lirmm.fr

***Abstract*— The purpose of this paper is to present 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 quality of the results. This paper presents a classification and a short state-of-the-art of the different types of existing simulators. Then we will introduce the architecture and functionalities of Thetis. Finally we will compare results obtained during sea-trial with our AUV Taipan300 and with our simulator.**

I. INTRODUCTION

The development of an AUV is not an easy task. Indeed, beyond the challenges of the mechanics and electronics, an intelligent software architecture is also required to play the fundamental role of controlling the machine in order to fulfil its mission. Of course one of the strong constraints is the ability of this architecture to work in real time and to react correctly to the different events occurring during the mission. The computing power and the miniaturization of the computers allow to imagine sophisticated architectures to assume scenarii increasingly more complex. In parallel the number and the complexity of tests necessary to validate this kind of architecture is growing up. Hence, simulation tools play an important role: they help us to test and validate control laws and software architecture, and to detect preliminary inconsistencies within the scenarii. Moreover, these technologies reduce the required human resources, decrease the number of necessary real experiments, and the time spent.

There are different types of simulators, each of them having its own limits and they can be used at the different steps of the development. So first we'll present the different architectures of the existing simulators and their applications. Then we'll expose the structure of our simulator specifying the models we consider. For details on previous development, please refer to [14] where we focus on the connectability aspects of Thetis to our real robots, and to [3] in which we demonstrate the simulation capabilities of communications between 2 AUVs. Finally we will compare sea-trials results with those obtained with *Thetis*.

II. DIFFERENT TYPES OF SIMULATORS

Simulators can be classified into 4 categories: the offline simulators, the online simulators, the hardware in loop simulators and the hybrid one [1].

A. Classification

1) Offline simulators: This type of simulator allows to design the control of robots in a first step. Matlab/Simulink is often used for this kind of simulation because of the availability of toolboxes (such as the one proposed by Fossen described in [4]) and because of the easy implementation of mathematical models. But we have to keep in mind that the temporal aspect of the simulation is not taken into account, and it potentially makes the control algorithm inoperative when it is transferred on a real robot. Thus it is not possible to validate control architecture or a sensor referenced command with such a simulator.

2) Online simulators: This type of simulator belongs to another family which allows to take the temporal consistency of the simulation into account. Indeed, with this type of simulation, 1 second of simulated time actually corresponds to 1 second in real time. This is the case in the SubSim simulator (see [5]). However the algorithms are still not executed on the robot itself, and the temporal behavior of the computer used for the simulation can be different from the one onboard the robot.

3) Hardware in loop simulators: The control algorithm is executed on the robot itself, but the commands sent to the actuators are routed towards the simulator instead of the real robot (see [6]). The simulator then considers the actuation commands in order to compute the dynamic evolution of the system. However, in this sort of simulation the external world is not taken into account, except for the dynamic effects of the environment. Only the proprioceptive sensors are simulated and the overall algorithms suite cannot be fully tested (obstacles avoidance, sensor based control...).

4) Hybrid simulators: These simulators are HIL simulators where real and virtual systems interact together in an virtual environment. It is therefore necessary to simulate an environment (static or dynamic) in which the robot software architecture will be fully functionally operative. Therefore, it is possible to test all the algorithms of the machine, from low level control of sensors or actuators recruitment to the high level algorithmic architecture. This approach has been used by several authors such as in [1], in which, the Neptune simulator is presented., This is a real time graphic multi-vehicles simulator, allowing to perform online, HIL and hybrid simulation.

B. State-of-the-art

In order to elaborate a command and a strategy to make our AUVs cooperate together, we are particularly interested in HIL or hybrid simulators which are able to simulate several heterogeneous vehicles. Several studies have been made on this topic, in the scope of underwater robotics.

The first works were done by the DARPA Naval Technology Office in 1988 ([7]). This simulator allows the cooperation between many underwater platforms which are driven by a real time intelligent sense-decide-act system. The simulation of sensors and environment are taken into account. In [8], a distributed simulator for underwater vehicles called Core Simulation Engine (CSE) has been developed. This system is equipped with operator training capabilities, mission feasibility assessment and mission replay. A subscription method and a run time infrastructure allowed the distributed programming. The Cooperative AUV Development Concept (CADCON) simulator ([9]) uses a client-server model in which it is possible to handle interactions between vehicles controlled by the simulation clients through an environment simulator. This simulator system focuses on the high level communication and does not deal with the dynamics and control of the heterogeneous vehicles. In [10], a simulator called DEVRE, composed of a set of processes running on a local area network, has been developed. The simulation runs onto 3 computers which are the onboard AUV computer, a human machine interface computer and a third computer used to calculate system dynamics and to represent a virtual world. In this simulator multi-vehicles simulation and communications between the AUVs are not allowed.

Many other simulators have been developed like in [11],[12], [13], or in [5] and all these simulators present several very helpful specificities, through several models (environment, vehicles) which are able to cope with reality accurately. Some of them are specialized in HIL simulation, while others are mostly designed for multi-AUVs applications, and others are focusing on accurate environment simulation. But, among all these references no one has addressed to the problem of temporal decoupling between the command computed by the onboard computer and the simulator(s). Yet this critical problem is addressed to in the next sections. Most of these simulators are not based on a distributed architecture, meaning that it is not possible to add another computer when the load of the existing ones becomes too heavy. This leads some teams to simplify the used models at the expense of the quality of the results. Moreover the considered systems are always AUVs (sometimes heterogeneous) but there is no mention about cooperation capabilities between a surface vehicle and an underwater one (for example). Yet, we find more and more scenarii in which an UAV (Unmanned Aerial Vehicle), an ASV (Autonomous Surface Craft) or a land vehicle are required to cooperate.

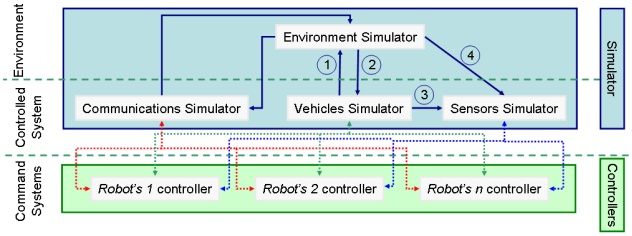

Fig. 1. Simplified architecture of Thetis: there is a logical sequence between 3 simulators even if there are independent processes on different computers. The cycle duration of this sequence must be largely lower than the period of the controller. The temporal decoupling is only effective (and that's 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).

For all these reasons we have developed Thetis, a hybrid simulator which provides the necessary environment to experiment HIL simulation with support of communications between the vehicles, and allows the use of exteroceptive sensors. Such a simulator which concurrently provides all these functionalities is a necessary tool to envisage such complex scenarii. This is a challenging multi-disciplinary task because of the number of specialists who must collaborate (acoustician, computer specialist, control engineer etc). For this reason the aim of this simulator architecture is not to provide a set of "perfect" models: in this paper we describe a simulator architecture allowing us to deal with the problem of heterogeneous coordinated vehicles, under communication constraints. Moreover, the open source Thetis project is built to allow many different specialists to develop their own models. On the other hand this simulator architecture is distributed, in order to allow the implementation of complex models, without being restricted by computational burden, thus guaranteeing the real-time aspect. Finally this simulator allows to interchange the used models, in function of the desired accuracy. As this paper focuses on the architectural aspects of the simulator, the different models, already implemented, are deliberately basic, except for the vehicles, where full hydrodynamic models are considered, as exposed in [4].

III. THETIS: HYBRID SIMULATOR

Thetis is a new real-time hybrid simulator allowing the consideration of heterogeneous multi-vehicles scenarii, including communication restrictions. A quick overview on the overall architecture of the simulator is presented on figure 1.

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.

B. Implementation and Technical Considerations

All the concepts exposed in the previous section are supported by using 3 mechanisms and the architecture is based on 4 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.

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 4 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 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 1. 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 [15].

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 [3].

1) Vehicles Simulator: The vehicles simulator is in charge of computing the robots dynamic evolution. We present this simulator structure on figure 2. The robots send actuators commands calculated by the onboard computer to the simulator through UDP socket. The *Dynamic Models* represented on Figure 2 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.

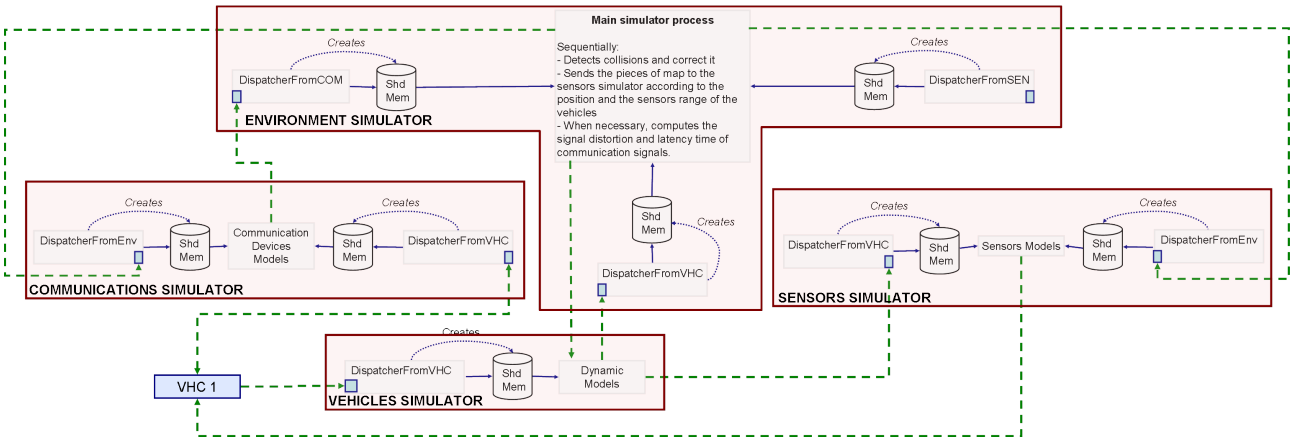

Fig. 2. Detailed view of Thetis. Dashed lines stand for the UDP/IP inter-simulators communications.

2) Sensors Simulator: The sensors simulator is in charge of providing the the real robot (onboard computer) with virtual data from the simulation. Presently the models of proprioceptive and simple exteroceptive (Temperature, Conductivity) sensors are implemented. Complex exteroceptive sensors (sonar and camera) will be the next steps of our work. The structure of this simulator is presented on Figure 2. The *DispatcherFromENV* 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 same refresh rate as the real sensor.

3) Environment Simulator: The environment simulator (figure 2) 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. 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.

4) Communications Simulator: The mechanism of the communication simulator (figure 2) 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 vehicles simulation has to consider these aspects explicitly. The communications simulator is in charge of simulating the 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, acsoutic modem). In the simulation case, these data are not sent to the physical communication device, but routed to the communications simulator (see [14] for more information about the connectivity of the simulator with our AUVs).

IV. Models and assumptions

In this chapter we briefly present the modeling methods (more details can be found in [14]) chosen to develop the simulator. Although it is not exclusively meant to simulate AUVs, it is the first model that we have implemented because our team works on this type of robot [16]. Other models will be implemented later if needed. Hence, the simulated sensors suite is dedicated to submarine applications and the modeled environment is exclusively underwater. Obviously, all this can be easily modified in order to deal with heterogeneous robots and environment like coordination between AUVs and UAVs.

A. AUV modeling

The robot model computes the robot accelerations according to the command vector. The modeling of the AUV, is made up of the hydrodynamic, hydrostatic and dynamic phenomena of the robot on the one hand, and of the actuators model on the other hand. Here is a brief description of these models:

- *Hydrodynamic forces*: the simulator uses the 6 DOF non linear equations of AUV expressed in the body fixed frame [4]. External disturbances(waves, oceanic currents, wind...) are not implemented yet. Moreover the potential damping, the skin friction and the wave drift damping are not considered. Only the damping due to vortex shedding is computed.

- *Thruster Model*: A quasi-steady modeling of thrust and torque is used. This approach has been used in [17]. The thrust T and the torque Q can be written as:

$$T = \rho D^4 K_T(J_0) n|n|$$
$$Q = \rho D^5 K_Q(J_0) n|n|$$

where ρ is the water density, D the propeller diameter, K_T and K_Q are 2 coefficients, J_0 the advance ratio and finally n the shaft speed.

- *Fins Model*: the robots Taipan have a cylindrical shape and are equipped with rudder at the stern and with 2 pairs of diving planes located at the bow and at the stern. The diving planes allow to control the pitch and heave dynamics while the rudder allows to control the heading [16]. We consider that the axis of rotation of the planes is located at a distance d_a from the origin of the local frame of the robot. The forces of the lift and drag and the induced moment are therefore given by:

$$\begin{bmatrix} F_x \\ F_z \\ M_q \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}\rho S_s V_0^2 (C_{zs}\sin\delta + C_{xs}\cos\delta) \\ -\frac{1}{2}\rho S_s V_0^2 (C_{zs}\cos\delta - C_{xs}\sin\delta) \\ F_z(lc_s\cos\delta - d_{as}) + F_x(lc_s\sin\delta) \end{bmatrix}$$

where l is the distance between the leading edge of the planes and the system metacenter, b_s is the wingspan, c_s is the chord, S_s is the surface wing defined by $S_s = b_s c_s$, l=0.2 for the taipan class of vehicles, C_{xs} and C_{zs} are respectively the lift coefficient and the drag coefficient corresponding to the axes of the surface.

B. Sensors modeling

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. Presently a GPS (Trimble Lassen SKII), a loch doppler (RDI Workhorse Navigator Doppler Velocity Log), as well as an attitude and heading reference system (XSens MTi) are modeled. As for the exteroceptive sensors, they are used to perceive the world surrounding the robot. For instance, a sonar, a camera or a CTD sensor can be used. As the physical phenomena driving the exteroceptive measurements are complex, the modeling process of these sensors can be heavy. We are presently working on a simple sonar model using classic ray tracing method.

C. Environment modeling

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. This model considers these phenomena as stationary.

Fig. 3. Taipan 300 by the Salagou Lake in France

D. Communication modeling

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 [3] for all the details of this model). Then using the following inequality, it is possible to determine the radius of the sphere that is bounded by the reception threshold of the receiver AUV.

$$SL - TL - NB > Threshold$$

where SL stands for Source Level of transmitter AUV, TL are the Transmission Losses, NB the Noise level reported on the Bandwidth and $Threshold$ is the reception Threshold of the receiver AUV. Thus it is possible to determine for each AUV (other than the emitter AUV) if it is in the signal propagation sphere. If not, the message will not be delivered. Else, we determine the reception level more precisely by taking into account the attitudes of the 2 vehicles and the directivity diagram of the antenna (using a lookup table). If the computed signal reception level is higher than the reception threshold of the modem, the message is delivered after a delay t which is computed by taking the distance between the 2 antennas into account, depending on the size of the message and finally the modems communication rate (20bit/s in our case).

V. Results

In this section we present some results obtained with Thetis on the one hand, and during sea-trials on the other hand. 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 3). As displayed on Figure 3 the vehicle is fitted with a CTD (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 5). 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

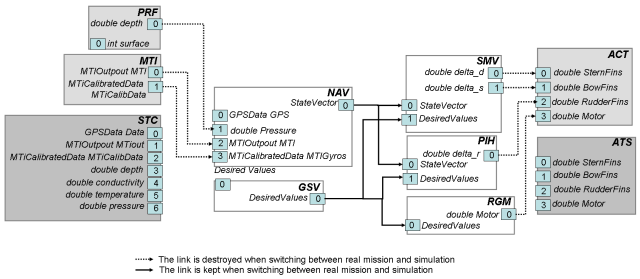

Fig. 4. Connection between the architecture modules

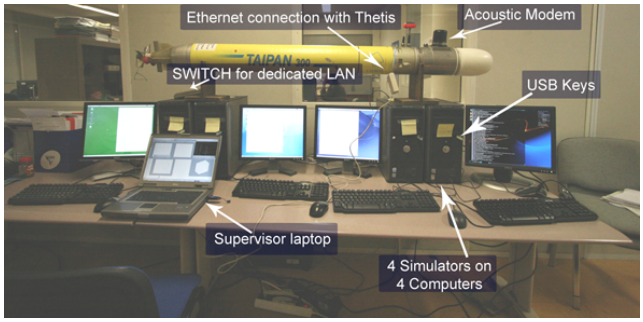

Fig. 5. The Thetis system and the AUV Taipan300

simulator is presented in [14]. 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 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. So in order to program a simple setpoint mission (keeping a desired heading for a given duration), we have to interconnect the modules as shown on fig. 4. The linkage of modules consists in establishing dynamic or static links which support the data and control flow upon the architecture. A module carries 6 categories of ports (data input port, data output port, events input port, events output port, parametrization port, request input port). The activity of these modules are controlled by a particular module called *scheduler*. On Figure 4, light gray blocks represent the modules used during the real mission. The dark gray ones represent those which are used during the simulation. Finally the white modules are shared modules used during real missions and simulations. As we can see, we only have to replace the light gray module by the dark grey ones to switch between real mission and simulation.

On Figure 5 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),

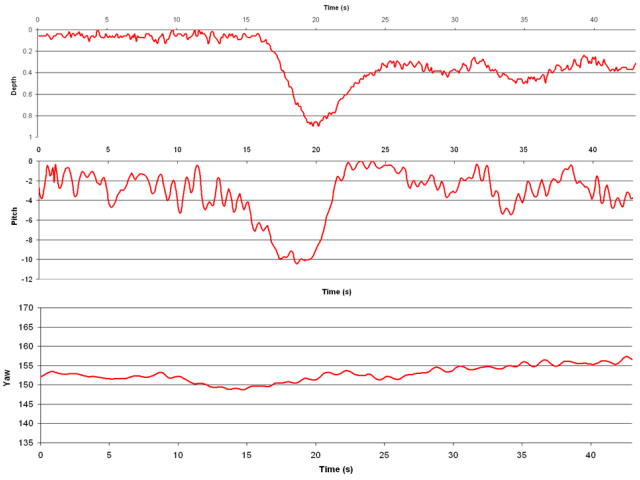

Fig. 6. Sea-Trials in the Salagou Lake in France

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.

Now we present some results illustrating one of our sea-trials. These results have been obtained at the Salagou Lake in the South of France. The objective of these missions was to tune the control gains and to validate our software architecture. The programmed mission was a simple 2-pitch-setpoint with a heading of 150 deg. and a desired depth of 0.4m. As we can see on Figure 6, the vehicle converges to the desired depth in approximatively 25s with an overshoot of about 0.4m.

We use the same control (and of course the same AUV) to obtain the results presented on Figure 7. These results have to be qualitatively interpreted, and confirm the quality of the AUV model we have used in the vehicle simulator. Note that the considered noise in the simulator is not realistic since the external disturbances have not been modeled yet. Even if this mission is very simple, this allows to validate our

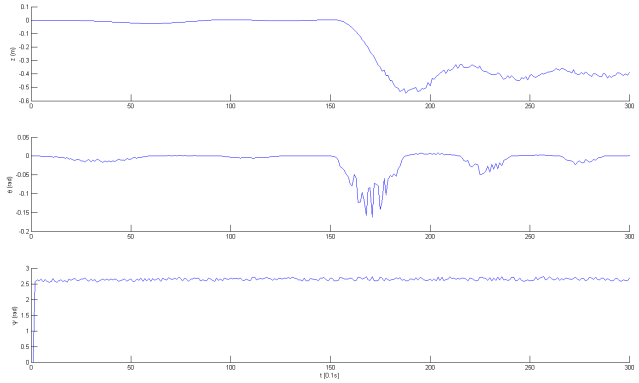

Fig. 7. Simulation of the Salagou Lake mission

work partially. Of course some more complex missions have been tested in simulation (see [3]) but as our AUVs are being upgraded, it is difficult to perform so complex real missions.

VI. CONCLUSION

This HIL simulator plays an important role in the development of the controllers of our robots. Thetis is a real-time multi-vehicles simulator which needs to be completed (including complex exteroceptive sensors). We have compared results from the simulation with basic real experimentation and we have seen that the behavior of our AUV was quite the same. The next step of our work will be to refine our hydrodynamic gains. Our contribution concerns the proposed simulator architecture. This architecture gathers all the important features that a simulator must include in order to allow simulation of flotilla. The models used are certainly less accurate than the ones within traditional simulators, but the structure of our simulator allows an easy evolution. The communications model we have presented is not very complex and accurate, but it could be useful for control design purpose. 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.

REFERENCES

[1] P. Ridao, E. Batlle, D. Ribas, M. Carreras Neptune: a hil simulator for multiple UUVs *OCEANS '04. MTS/IEEE TECHNO-OCEAN '04*, pages 524- 531, 2004

[2] 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

[3] 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

[4] TI. Fossen Marine Control Systems: Guidance Navigation and Control of Ships, Rigs and Underwater Vehicles *Marine Cybernetics AS*, 2002

[5] T. Bielohlawek SubSim - An Autonomous Underwater Vehicle Simulation System *Ag Robotersysteme Fachbereich Iinformatik An Der Unuversitt Kaiserslautern*, 2006

[6] Suriano, Moriconi A Distributed Simulator for the Development of the Unmanned Underwater Vehicles Control Software *Robotics and Applications and Telematics*, 2007

[7] Albus System Description and Design Architecture for Multiple Autonomous Underwater Vehicles *National Institute of standards and Technology, Gaithersburg, MD, Technical Note 1251*n 1988

[8] D.M Lane et al. Mixing simulation and real subsystems for subsea robot development *Proc; IEEE OCEAN'98, Nice, France, pp. 1382-1386*, 1998

[9] Chappell, Steven G.; Komerska, Rick J. An Environment for High-Level Multiple AUV Simulation and Communication *Proceedings of UI*,2001

[10] Ridao P., Battle, J., Amat, J., Carreras, M. A distributed environment for virtual and/or real experiments for underwater robots *Proceedings 2001 ICRA. vol.4, no., pp. 3250-3255 vol.4*, 2001

[11] W. Hornfeld DeepC, the German AUV Develoment Project *status report of the STN ATLAS Elektronik GmbH*, 2001

[12] Kobayashi et al. Development of an autonomous underwater vehicle maneuvering simulator *OCEANS vol.1, no., pp.361-368 vol.1*, 2001

[13] Carlson, E.A., Beaujean, P.-P., An, E. Simulating communication during multiple AUV operations *Autonomous Underwater Vehicles, vol., no., pp. 76-82*, 2004

[14] O. Parodi, A. El Jalaoui, D. Andreu (2008) Connectivity of Thetis, A Distributed Hybrid Simulator, with a mixed Control Architecture *The Fourth International Conference on Autonomic and Autonomous Systems ICAS08*, 2008

[15] http://sourceforge.net/projects/doxygen/ accessed on 02/08

[16] J.M. Spiewak, B. Jouvencel, P. Fraisse A New Design of AUV for Shallow Water Applications: H160 *ISOPE'06: International Offshore and Polar Engineering*, 2006

[17] LL. Whitcomb, DR. Yoerger Development, Comparison and preliminary experimental validation of nonlinear dynamic thruster models *IEEE journal of oceanic engineering*, pasges 481-494, 1999