# Thetis: A Real-Time Multi-Vehicles Hybrid Simulator for Heterogeneous Vehicles

Olivier Parodi, Lionel Lapierre, Bruno Jouvencel

# *Thetis*: A real-time Multi-Vehicles Hybrid Simulator for Heterogeneous Vehicles

Olivier Parodi, Lionel Lapierre, Bruno Jouvencel
Department of Underwater Robotics, LIRMM
University of Montpellier 2
34000 Montpellier, France
{parodi, lapierre, jouvencel}@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 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).**
**This paper presents the architecture and functionalities of *Thetis* and present some simulation results with the support of communications.**

## 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 [1] where we focus on the connectability aspects of Thetis to our real robots, and to [2] in which we demonstrate the simulation capabilities of communications between 2 AUVs. Finally we will compare sea-trials results with those obtained with *Thetis*.

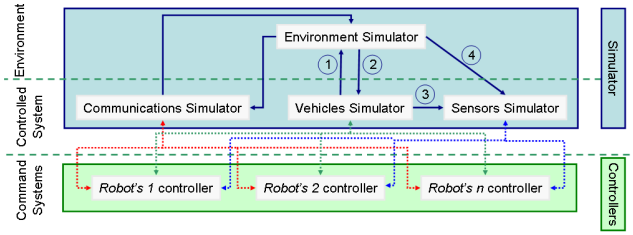


Fig. 1. Simplified architecture of Thetis: there is a logical sequence between 3 simulators even if there are independent processes on different computers. The cycle duration of this sequence must be largely lower than the period of the controller. The temporal decoupling is only effective (and that's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

## II. 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 [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 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.
Since we only own AUVs in our team, we have currently implemented a full hydrodynamic model for torpedo-shaped robots. 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.

*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.

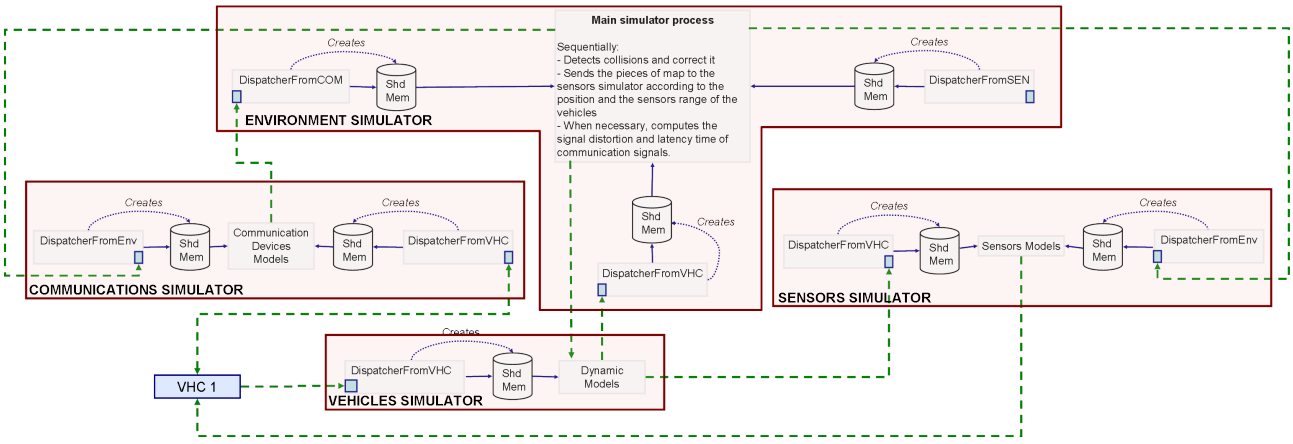


Fig. 2. Detailed view of Thetis. Dashed lines stand for the UDP/IP inter-simulators communications.

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 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.
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.

*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


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

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 [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 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

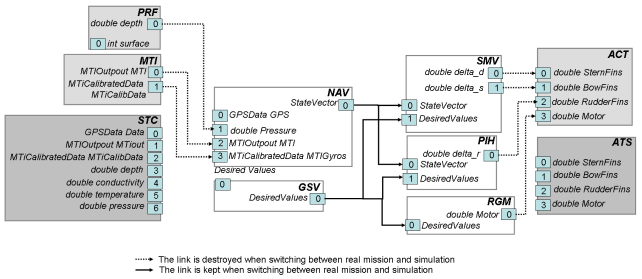


Fig. 4. Connection between the architecture modules

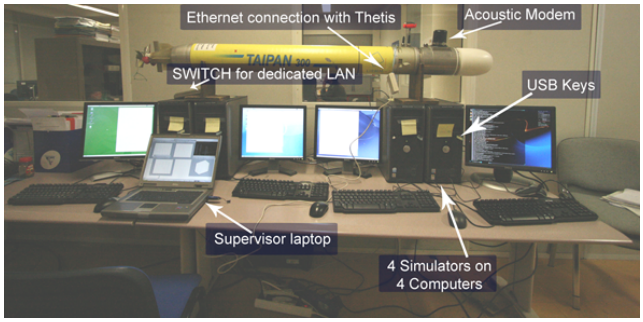


Fig. 5. The Thetis system and the AUV Taipan300

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, presented in [4], 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 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), 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. Preliminary results

These 2 AUVs carry a Tapac modem. These modems have a transmission power of 177 dB and a bandwidth of 6Khz. Their reception trigger level is equal to 15dB beyond environmental noise. Their radiation chart is omnidirectional. Its transmission frequency is 33 Khz (chirp modulation). In this simulation we assume that the wind velocity is equal to 5 knots. Using the Wenz model we determine that the modems maximal communication range is equal to 1900 m, which corresponds to constructor values. In order to simplify the interpretation of the results, only one of the 2 vehicles is moving during the simulation. The first vehicle (Taipan 300) is moving along a straight line, restricted to the horizontal plane. Moreover, the pitch and roll angle are set to zero, which guarantees the vehicle to follow a straight line, forced to a constant depth. The second AUV is fixed (actuators are set to 0) with a constant depth (the depth value is forced in the vehicle simulator). In this simulation, Taipan 2 is in a fixed point and emits messages (constituted by the string "HELLO WOLRD TAIPAN 2 IS BROADCASTING A MESSAGE"). The trajectory followed by Taipan 300 is shown (fig 6). Each 350s a dot is drawn on the trajectory. The blue circle on the figure represents the transmission area of Taipan 2. Taipan 2 emits its message at t=60s, t=1500s et t=2300s. These instants are represented on the Taipan 300 trajectory by 3 squares. We can see that at t=0, Taipan 300 is out of the communication range of Taipan 2. At t=650s communication becomes possible.

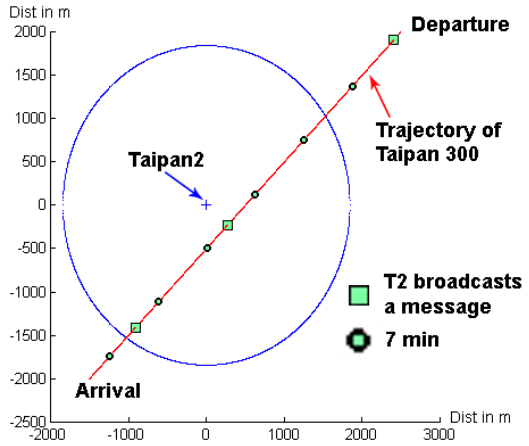


Fig. 6. Trajectory of Taipan300

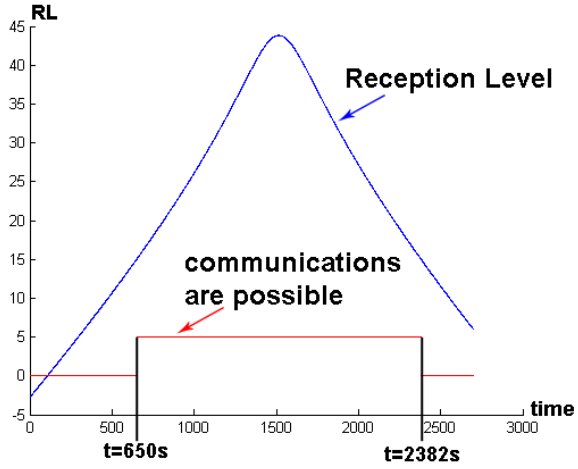


Fig. 7. Reception level during the trajectory of Taipan 300

Finally at t=2382s T300 is out of the T2 communication range (fig 7). Figure **??** represents a timeline with the date of emission/reception of messages from the vehicles. We assume that each transmission has a fixed delay and a null duration. Figure **??** presents data logged by the 2 vehicles and the environment simulator. We distinguish these data on the graph by the height of their peak 1: Taipan 2, 2: Environment simulator, 3: Taipan300. On the "zoom 2" we can see that the first message sent by Taipan 2 is received by the environment simulator (a few milliseconds later) but never by Taipan 300 (no Taipan 2 peak after). The second and the third message are sent by Taipan 2 and received by Taipan 300 after 20s ("zoom 1"). This duration corresponds to the latency and the necessary duration to send the message. This simple simulation allows us to illustrate all mechanisms mentioned in the previous sections. AUV trajectories are enough to validate the environment and communications simulators.

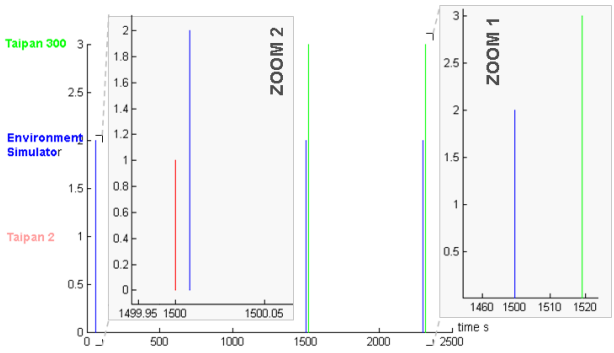


Fig. 8. Logs of the simulated mission

## V. 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 need to compare the results from the simulation with real experimentations. This is especially true for the underwater communications. Taipan 2 should be available in a few months and we will be able to experiment the effective rate of communication between the AUV and a second acoustic modem located on a surface vessel. 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. 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] 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