HLA Interoperability for ROS-Based Autonomous Systems

Arnau Carrera[1(✉)], Alberto Tremori[1], Pilar Caamaño[1], Robert Been[1], Diego Crespo Pereira[2], and Agostino G. Bruzzone[3]

[1] NATO STO CMRE, La Spezia, Italy
{arnau.carrera,alberto.tremori,pilar.caamano, robert.been}@cmre.nato.int
[2] Universidade de Coruña, Coruña, Spain
diego.crespo@udc.es
[3] University of Genoa, Genoa, Italy
agostino@itim.unige.it

Abstract. The requirements for autonomous systems (of systems) have started to include the cooperation between heterogeneous assets in order to accomplish complex missions. Therefore, interoperability – both at the conceptual and technical level – between different types of systems and domains is essential. In the M&S community HLA is the reference standard to design, develop and test interoperable systems of systems. In this article, a HLA-based link between simulation and an autonomous system using the ROS middleware is presented. The integration of an Autonomous Underwater Vehicle (AUV), more specifically the SPARUS II, in a HLA federation using the proposed link has been tested for a harbour protection mission. For this scenario, the hardware and software of the AUV has been included in a federation together with a virtual simulator. The link allows easy inclusion of ROS-based assets in HLA federations, thereby enriching both the M&S and robotic communities which will benefit from this approach which allows the development of more complex and realistic simulated scenarios with hardware- and software-in-the-loop.

Keywords: High level architecture (HLA) · Robot operating system (ROS) · Interoperability · Autonomous underwater vehicle (AUV) · Autonomous systems · Hardware in the loop (HIL) · Software in the loop (SIL)

1 Introduction

The development of autonomous systems requires the construction of complex hardware and software systems. To simplify the software development, several robotics frameworks have been developed in the last years. The aim of the majority of these frameworks is to provide hardware abstraction, low-level device control, implementation of commonly-used functionality, and message passing. The best known frameworks are ROS [1], MOOS [2], Player [3], YARP [4], and OROCOS [5]. One of the most widely used is the Robotic Operating System (ROS).

Developments in the area of robotics are commonly expensive and lengthy. After the first stages of design, the hardware (HW) and software (SW) are tested through several (re-)defined phases until the final system is deployed. In these phases, the use of simulation is desired and is widely diffused in the robotic community with simulated events injected directly in the operating system of the robot. Nevertheless the use of an approach closer to the M&S community, with the "immersion" of the system in a simulated scenario by HLA is still not very diffused. By *cross-communities* this approach of validation and verification (V&V) of both the SW and HW can be carried out in more realistic environments prior to their implementation and deployment, thereby increasing the reliability and robustness of the final systems. Also, new applications for an existing robotic platform would ideally need to go through a similar V&V process. Moreover, the use of simulators with synthetic environments is a complementary approach to the standard simulation used in robotics and will improve the quality of testing systems, reducing both time and resources for V&V [6].

The use of Modelling and Simulation (M&S) techniques could really benefit robotics developers. In order to optimize this process, and to promote interoperability, the developers are advised to follow standards, which for M&S is the High Level Architecture (HLA) [7] (IEEE standard). The use of HLA allows the creation of heterogeneous and complex experimental frameworks, even mixed simulated and real systems simultaneously increasing the richness of the simulations [8]. Two examples of complex simulations where M&S and HLA are extensively used are NASA [9] and ESA [10] agencies, but HLA is also the reference standard for the defence sector and in particular by NATO (STANAG 4603).

The authors of this paper have proposed a new approach to bridge the Robotic and M&S communities by designing and developing a wrapper to grant HLA-based interoperability to autonomous systems using the ROS Framework. The main purpose of developing the connection between HLA and ROS is to minimize the effort of including existing autonomous 'ROS systems' in complex, interoperable and standards-based simulated scenarios.

To test the proposed wrapper between the M&S standard and the robotic middleware, a HLA federation has been developed including hardware and software in the loop. The included hardware consisted of an Autonomous Underwater Vehicle (AUV) called SPARUS II, together with its control architecture COLA2. A maritime simulator was used, for the representation of a complex maritime scenario in a 3D realistic environment.

The paper is organized as follows. Section 2 introduced previous cases of HLA in the robotics field. Section 3 compares the communication system used in ROS and HLA and it presents the designed middleware. Section 4 explains the use case and the different elements composing it. Section 5 summarizes the contributions of this paper and presents the future work.

2 Background

To the best of the author's knowledge, there is no previous work published with the purpose of connecting the ROS framework with HLA-based federations. There have been other developments related to the robotics field and interoperability using HLA; below, we have highlighted three different examples of HLA and robotics:

- In 2001, Lane et al. [11] proposed an architecture for Unmanned Underwater Vehicle (UUV) based on HLA. The team divided the robot in subsystems according to functionalities and made them interoperable using HLA. This architecture allows inclusion or replacement of subsystems. Furthermore, they use HLA to perform fast simulations with simulated sensors. However, they conclude that the effort to maintain and develop UUV subsystems in this structure requires a lot of extra effort. Moreover, they doubt that the proposed fast simulation of some sensors was valid for long term simulations.
- In 2005, Joyeux et al. [12] proposed inclusion of simulation capabilities in an existent robotic architecture called LAAS. The aim of this work was to enable the use of the same code in simulation and in real environments, and the combination of simulated and real elements. In this case, the control architecture was modified replacing the message passing between elements of the HLA standard.
- In 2011, Nebot et al. [13] presented a new architecture for controlling a group of heterogeneous robots cooperating to achieve a global goal. In this case the HLA allows the data distribution and implicit communication among the parts of the systems, this also allows operation of simulated and real systems simultaneously. In this case, the sensors and complete vehicles are HLA compliant. A layered control architecture was proposed where the Player robotics framework controls the low level modules, HLA distributes the communication and the JADE [14] software coordinates the cooperation of the robots.

In the literature sub-set mentioned above, the authors have detected as a common goal the possibility to connect simulated and real systems using a standardized architecture. The first article considered an overhead to maintain HLA compatibility in the low levels, whereas others have integrated HLA in the abstracted layers of the hardware. The mentioned cases have developed a new architecture model based on HLA standards.

The aim of our proposal is to preserve the existent control architecture and its framework in the system, and to only add a layer to enable interoperability of the desired parts of the (system of) systems, at software and/or hardware level. This approach will avoid re-designing the entire system and maintains the manufacturers' software 'as is', i.e. without modifications.

3 ROS-HLA Wrapper

A ROS-HLA wrapper has been developed to guarantee technical and conceptual interoperability between the robotic system (HW and SW) and the simulated environment. This wrapper is installed in the ROS system and is able to centralize the different communications in the autonomous system and to exchange the desired information with the HLA federation.

In order to design the wrapper, it was necessary to perform a study of the different types of communications of both standards to find the most suitable adaptation. Section 3.1 summarizes the principal similarities in ROS and HLA. This study points out the required elements and a unique intermediary module linking the HLA and ROS in order to respect the requirements of both communications protocols. The details of the design and implementation are presented in the Sect. 3.2.

3.1 Comparison of the Communication Protocols in ROS and HLA

To understand the communications it is necessary to remember the aim of the two frameworks. HLA is designed in order to standardize the exchange of information between different simulators distributed over a heterogeneous network. ROS, however, offers support for passing messages between processes mainly in the same system or over a local network. The aim of this functionality is to share information of sensors, actuators and algorithms to control the behaviour of a system.

To facilitate the comprehension it is worth to mention the organization of HLA and ROS. HLA is organized in a *federation* which is formed by *federates*. A *federate* is a simulator composed by an *ambassador* and multiple processes; the *ambassador* is responsible to share the information. ROS is organized in a more modular form where each process is called *node* and is responsible to share the information. A *master* is required only to stablish the connection between nodes.

These different characteristics imply some design differences which are explained in the following:

Network Topology/Structure. The first important difference is the network topology, HLA uses a centralized network (bus) in order to increase efficiency and to provide certain functionalities (e.g., time management). This bus of communication is called runtime infrastructure (RTI). On the other side, ROS uses a decentralized structure (peer-to-peer), to be more robust against failures and to increase scalability. To resolve this difference the wrapper will have to centralize all the different ROS communications in one node and transmit them to the HLA Federation.

Definition of the Shared Information. Both systems share the requirement to define a document with the structured data shared between the elements. In HLA this document is called Federation Object Module (FOM), it has to be shared by all the systems in the federation. It specifies all the information related with the data sharing. ROS uses separate documents for each type of shared information, which only need to be known by the processes that use this data. These documents are called *messages* and are created using a simple interface definition language (IDL) which only defines the type and name of the data contained. The proposed wrapper defines a FOM which includes the information of the *message* and extends it to fulfil the requirements of HLA.

Communication Style. HLA and ROS systems use two types of communications: synchronous and asynchronous.

Asynchronous Style. In HLA each kind of data (*Object*) is composed of two parts: the *Class* and the *Instance*. For example: *federate A* declares in the *federation* that it will include *Object_1* when the RTI finds some other, say, *federate B* which has an *Instance* of the *Object_1*; it will then create a new *Instance* to receive the information in *federate A*. The instance in *B* is *posting* the information and the instance in *A* is *reflecting* it. On the other hand, in ROS, each *node* declares the information to be sent (*publish)* and to be received (*subscribed*) assigning them a name (*topic*) and type (*message*). In this case, when the *node* is connected to the *master* at initialization time, it gives this information to it. The *master* creates the connection (*pipes*) between the different *nodes* according to *publishers* and *subscribers*.

Synchronous Style. In HLA this communication is called *Interactions*, and the same class can perform the function to receive and send interactions through the *ambassador*. The ROS approaches this communication style, calling it *Services*, where each node can declare service *servers* (receivers) or *clients* (senders) which are defined by name (*topic*) and type (*message*).

The wrapper is situated between the two communication protocols, so when it receives information from one framework it has to send it on the other. The wrapper can create different *Objects* and *Instances* according to a configuration file. The *Instances* have been modified to include ROS functionalities, when it *posts* it is due to the reception of a ROS *message* (*subscription* callback) and when it *reflects* it is *publishing* a ROS *message*. The synchronous communications is limited by the ROS part being only possible to receive or send interactions, not in a bidirectional manner like in HLA. So when it receives an HLA *Interaction* it *calls* a ROS server, and when it receives a ROS *call* it sends an HLA *Interaction*.

3.2 ROS-HLA Wrapper

The authors adopted the strategy of developing a ROS *node* which converts the information from ROS to HLA, and vice versa. Only one *node* will be required in each ROS system and it centralizes all the exchange information between the autonomous system and the *federation*. This approach isolated the HLA code in one process avoiding modification of the control architecture existing in the autonomous system.

To adapt the node to different autonomous systems and federations it is necessary to identify the required information to be exchanged adapting the FOM and developing the HLA *Objects* and *Interactions* together with the ROS *messages* and *services* (see Fig. 1). So for each kind of data exchanged, the user implements the required *Class*, *Instance*, and *Interaction* which besides the HLA structure includes the ROS functionalities. Figure 2 shows a standard HLA implementation used in all *federates*. The wrapper is a ROS node which creates the necessary HLA Classes, Instances, and Interactions according to the configuration file. This file defines the exchange of information and the direction of the communication in order to define the correct ROS initializations. Figure 1 illustrates the description of the wrapper with an object diagram.

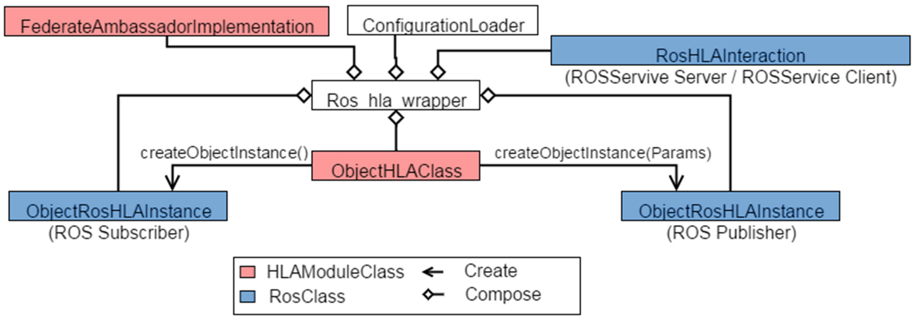

Fig. 1. Object diagram of the ROS HLA middleware or wrapper implemented.

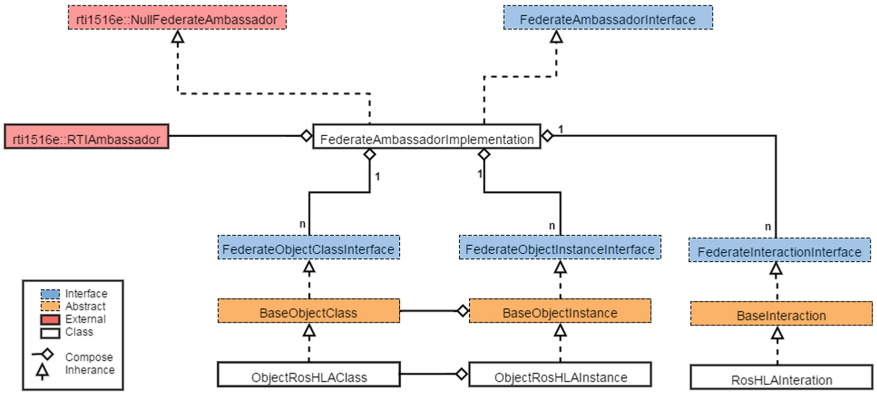
rti1516e::NullFederateAmbassador
FederateAmbassadorInterface
rti1516e::RTIAmbassador
FederateAmbassadorImplementation
1
1
1
n
n
n
FederateObjectClassInterface
FederateObjectInstanceInterface
FederateInteractionInterface
BaseObjectClass
BaseObjectInstance
BaseInteraction
ObjectRosHLAClass
ObjectRosHLAInstance
RosHLAInteration
Interface
Abstract
External
Class
Compose
Inherance

Fig. 2. Class diagram of the HLA architecture implemented by the HLA system developed in this work.

4 Experimental Use Case

The ROS-HLA wrapper presented in this paper has been developed at the NATO STO’s Centre for Maritime Research and Experimentation (CMRE) in the context of the Persistent Autonomous Reconfigurable Capability (PARC) project. The aim of the PARC project is to address the technology and engineering requirements of future unmanned systems of systems in the maritime domain. It is focused on increasing the persistence, interoperability and scalability of such systems whilst addressing standardization, information assurance and cost aspects.

Taking into account the environment and the expertise of the CMRE, the chosen use case is a harbour inspection scenario, where an AUV has to patrol a harbour area, visiting several strategic (inspection) points. The harbour has been represented using a virtual simulator of a maritime scenario and the autonomous system used is the SPARUS II AUV [15] and COLA2 [16] control architecture, which is ROS based. Moreover, the control architecture has been divided in two parts: the *back-seat* controlling the high-level decision and the *front-seat* at the low-level (i.e. sensors and actuators). Therefore, the federation designed is composed by three federates (see Fig. 3): the simulator, the front seat, and the back seat of the SPARUS II AUV. The Run Time Infrastructure used is the product PitchRTI distributed by Pitch; experimentations based on MAK RTI are also planned.

The following sections introduce a brief description of the SPARUS II AUV and the COLA2. We also present the new division in COLA2 to split the front seat and back seat in different systems and the addition of the dynamic simulation.

4.1 Sparus II AUV

The SPARUS II AUV is a lightweight AUV developed by the University of Girona (UdG) which is controlled using the Component Oriented Layer-based Architecture for

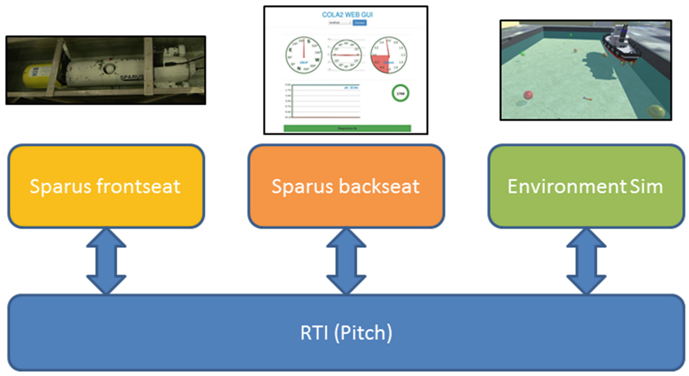

Fig. 3. Schema of the harbour protection federation composed by three federates.

Autonomy (COLA2). The COLA2 is a ROS-based control architecture, hence all the different elements are organized as independent *nodes* which interact using a *publish/subscribe* method (or *service* calls). The COLA2 has been organized in four different groups/layers: backseat, front seat, SPARUS simulator and the SPARUS real hardware. This new semantic separation allows us to organize the COLA2 in a more usual structure for AUVs. Moreover, it makes it easier to execute the modules independently on different machines or federates.

Figure 4 shows the four modules which can be executed on different machines using the ROS-HLA wrapper to send the information through the RTI. In addition, the shared information by the HLA and ROS is highlighted in green. Below, we explain each module and the kind of information which has been selected as input and output.

COLA2 *Back-Seat*. This group contains all the behaviours of high-level decision making. In the use case presented in this paper, the SPARUS II AUV is performing either way-point navigation or navigates according to the commands given by a human operator. The inputs of this layer are desired positions; trajectory (formed by several way-points) and navigation information (position and velocity of the AUV). This information or the Human interaction commands are used to generate the output which consists of position and velocity commands for the AUV.

COLA2 *Front-Seat*. This group contains the low-level controls; it is able to mix all the information received by the sensor and estimate the current position and velocity of the AUV. On the other hand, it generates the proper commands for the AUV to reach the desired position or speed. Moreover, there are safety procedures which are enabled if some error appears in the sensor data. Therefore, the inputs are the position or velocity commands from the COLA2 backseat and the IMU (acceleration and orientation), GPS (global position in surface), DVL (velocity of the AUV), and Pressure (to estimate the depth); the outputs are the estimated position and velocity of the AUV, desired position generated from safety node and thruster set-point.

Simulator and Real Hardware. Depending on the specific application, the vehicle can be completely simulated, partially simulated or not simulated by using just the real

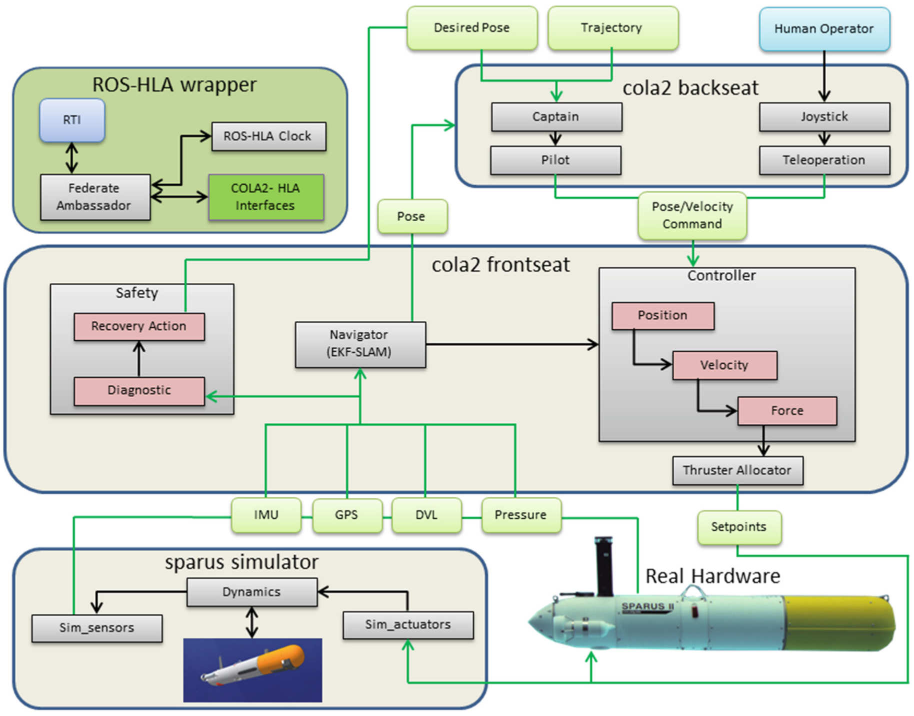

Fig. 4. Schema of COLA2 architecture organized in four different groups.

hardware. When the real hardware is used, the COLA2 front seat is running on the system the HLA communication is skipped, using the real hardware connections. When the vehicle is simulated, the dynamics and sensors simulation can be executed on a different system than the COLA2 front seat. To simulate the AUV, the node receives the Set-point of the thrusters, which are then converted to forces applied to the vehicle; then, using a model of the vehicle, the new position and velocity are estimated. Finally, the new position and velocity are used to generate the simulated navigation sensors.

4.2 Validation Scenario

In the scenario used for the validation of the ROS-HLA wrapper, the SPARUS II AUV was put in a water tank and executing the COLA2 front seat. Outside the vehicle, there are two other systems: the back seat and the environment simulator. The navigation sensors of the AUV are simulated in the external simulator and sent through HLA to the COLA2 front seat. The front seat then estimates the position of the vehicle and according to the commands of the backseat generates the commands for the actuator. In this scenario the commands are shared between the real hardware and the simulator. The harbour mission consists of following a trajectory and visiting a set of inspection points, where the AUV carries out simple operations (search for threats and communication through gateways), see Fig. 5.

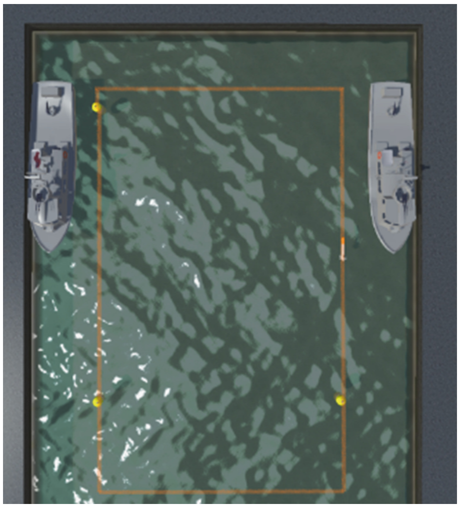

Fig. 5. A top view of the simulated harbour scenario with the SPARUS AUV.

In this scenario configuration, both systems (AUV HW and SW in the loop) have been able to follow the desired trajectory in the virtual environment. Moreover, this scenario has allowed the validation of the correct exchange of all the different communication protocols. The asynchronous communications of the low-level sensors and actuators, shared between the COLA2 front seat and the SPARUS II simulator, and the high-level communications represented by navigation information and position and velocity commands, shared between the COLA2 backseat and the COLA2 front seat. The synchronous communications have been tested sending command to start and stop trajectories and define points to be visited by the AUV using the simulator.

5 Conclusions

This article presented the first steps towards the creation of a HLA-based environment to connect ROS-based robotic systems. This environment may be used for testing (V&V), but also for a broader area of applications such as the simulation of a heterogeneous vehicle/sensor network over the internet. The authors explained the approach to develop a wrapper connecting the ROS framework with the HLA standard, which has been designed and implemented with the objective of easily connecting autonomous systems (hardware- and/or software) to realistic modelling and simulation HLA- based environments. The proposed wrapper maintains the existing control architecture, and can be adapted to the user's requirements. It has been successfully tested using a SPARUS II AUV in a simulated harbour inspection scenario. The hardware and software of the AUV have been included in the loop of the simulation and the different types of communications have been tested.

Future developments of the federation presented will include a more complex environmental simulator and a simulator for the underwater/radio frequency communications between the different assets in the scenario. New missions will for example include a two AUV scenario, i.e. both a real one at sea and a virtual AUV in a simulated representation of the real environment. On the other hand, will be investigated the integration of Command and Control (C2) systems in the Federation, in order to enrich the simulation with the possibility to evaluate the scenario and command the virtual and real assets.

In this context we envision a continuous effort to improve and expand the M&S capability to support autonomous systems S&T by contributing in a more consistent way to all three main areas related to the life cycle of new systems and processes (Engineering-CD & E-Training), for example by exploring new areas in the future such as the application of synthetic environments for supporting Human Machine Interface innovation in scenarios involving autonomous and manned assets in the maritime framework.

References

1. Quigley, M., Conley, K., Gerkey, B.P., Faus, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: and open-source Robot Operating System. In: International Conference on Robotics and Automation (ICRA) Workshop on Open Source Software, Kobe, Japan (2009)
2. Newman, P.: MOOS: Mission Oriented Operating Suite. In: Oxford Mobile Robitcs Group (2001)
3. Gerkey, B., Vaughan, R.T., Howard, A.: The player/stage project: tools for multi-robot and distributed sensor systems. In: Proceedings of the 11th International Conference on Advanced Robotics (ICAR 2003), Coimbra, Portugal (2003)
4. Metta, G., Fitzpatrick, P., Natale, L.: YARP: yet another robot platform. Int. J. Adv. Robot. Syst. **3**(1), 43–48 (2006)
5. OROCOS Project, Open Robot Control Software (2003). http://www.orocos.org/
6. Hodicky, J.: Modelling and simulation in the autonomous systems' domain - current status and way ahead. In: Hodicky, J. (ed.) MESAS 2015. LNCS, vol. 9055, pp. 17–23. Springer, Heidelberg (2015)
7. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) Framework and Rules. In: IEEE Std. 1516-2010 (Revision of IEEE Std. 1516-2000), pp. 1–38 (2010)
8. Hodicky, J.: HLA as an experimental backbone for Autonomous System integration into operational field. In: Hodicky, J. (ed.) MESAS 2014. LNCS, vol. 8906, pp. 121–126. Springer, Heidelberg (2014)
9. Reid, M.R., Powers, E.I.: An evaluation of the high level architecture (HLA) as a framework for NASA modeling and simulation. In: Proceedings of the 25th NASA Software Engineering Workshop, Greenbelt, MD, USA (2000)
10. Arguello, L., Miró, J.: Distributed interactive simulation for space projects. In: ESA Bulletin, pp. 125–130 (2000)
11. Lane, D.M., Falconer, G.J., Randall, G., Edwards, I.: Interoperability and synchronisation of distributed hardware-in-the-loop simulation for underwater robot development: issues and experiments. In: Proceedings of International Conference on Robitics & Automation (ICRA), Seul, Korea (2001)

12. Joyeux, S., Alami, R., Lacroix, S., Lampe, A.: Simulation in the LAAS architecture. In: Proceedings of the International Conference on Robotics and Automation Workshop on Software Development in Robotics, Barcelona, Spain (2005)
13. Nebot, P., Torres-Sospedra, J., Martínez, R.J.: A new HLA-based distributed control architecture for agriculture teams of robots in hybrid applications with real and simulated devices or environments. Sensors **11**(4), 4385–4400 (2011)
14. Bellifemine, F., Poggi, A., Rimassa, G.: JADE-A FIPA-compliant agent framework. In: Proceedings of PAAM, London, UK (1999)
15. Carreras, M., Candela, C., Ribas, D., Mallios, A., Magí, L., Vidal, E., Palomeras, N., Ridao, P.: Sparus II, design of a lightweight hovering AUV. In: Proceedings fo the 5th International Workshop on Marine Technology, Martech 2013, Girona, Spain (2013)
16. Palomeras, N., El-Fakdi, A., Carreras, M.: COLA2: a control architecture for AUVs. IEEE J. Ocean. Eng. **4**, 37 (2012)