# IMC: A Communication Protocol for Networked Vehicles and Sensors

Ricardo Martins
Paulo Sousa Dias

Eduardo R. B. Marques
José Pinto

João B. Sousa
Fernando L. Pereira

LSTS – Underwater Systems and Technology Laboratory
Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias s/n
4200-465 Porto, Portugal
{rasm,pdias,edrdo,zepinto,jtasso,flp}@fe.up.pt

***Abstract*—This paper presents the Inter-Module Communication (IMC) protocol, a message-oriented protocol designed and implemented in the Underwater Systems and Technology Laboratory (LSTS) to build interconnected systems of vehicles, sensors and human operators that are able to pursue common goals cooperatively by exchanging real-time information about the environment and updated objectives. IMC abstracts hardware and communication heterogeniety by providing a shared set of messages that can be serialized and transferred over different means.**

**The described protocol contrasts with other existing application level protocols by not imposing or assuming a specific software architecture for client applications. Native support can be automatically generated for different programming languages and/or computer architectures resulting in optimized code which can be used both for networked nodes and also for inter-process and inter-thread communication.**

**The protocol has already been tested throughout various experiments led by LSTS where it has taken care of communications between vehicles, sensors and operator consoles. We are now developing the protocol in the direction of having multi-vehicle cooperation using live data from environmental sensors and mixed-initiative user interaction.**

## I. Introduction

This paper presents the Inter-Module Communication (IMC) protocol, a message-oriented protocol designed and implemented in the Underwater Systems and Technology Laboratory (LSTS) for communication between heterogeneous vehicles, sensors and human operators. Recent technological advances have led to the proliferation of several types of unmanned robotic vehicles that are able to perform dangerous, long and dull tasks even while unattended. In LSTS, we have built several such vehicles, namely Autonomous Underwater Vehicles (AUVs) [1], Autonomous Surface Vehicles (ASVs) [2], Unmanned Air Vehicles (UAVs) [3], and Remotely Operated Vehicles (ROVs). Some of those are shown in Figure 1. Along with vehicles, we are also using different types of wireless sensors that can be embedded in static or drifting buoys, shown in Figure 2.

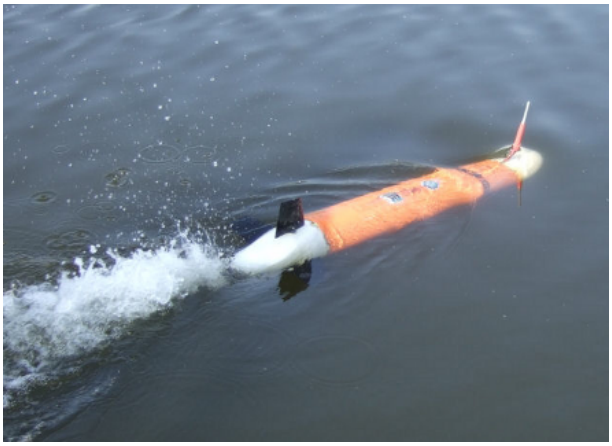
Our main objective is to build interconnected systems of vehicles, sensors and human operators that are able to pursue common goals cooperatively by exchanging real-time information about the environment and updated objectives. In these systems, operators can be kept in the loop in order to change the behaviour of the system in real-time. These operators can also answer to unforeseen situations where their input is crucial, following a mixed-initiative interaction pattern. There is a large number of applications for such systems like adaptive sampling [4], border patrolling or cooperative warfare [5], but all of these come at the cost of high complexity, in part because of sensor, vehicle and communication links heterogeneity. To answer this problem, we are using reusable, interoperable and independent blocks, all communicating with IMC, enabling rapid integration of new hardware and easing the development of new cooperation algorithms.

This paper is structured as follows. Section II gives an overview of the protocol and how we are applying it to control different existing systems. In Section III we provide technical details about the current implementation. Section IV discusses related work. Section V concludes and highlights directions for future work.

## II. IMC Overview

The IMC protocol comprises different logical message groups for networked vehicle and sensor operations. It defines an infrastructure that is modular and provides different layers for control and sensing. The use of IMC in the context of the Seascout Light AUV (LAUV) [1], [6] - depicted in Figure 1a - is a typical example. Figure 3 illustrates the use of IMC within the LAUV. The message flow corresponds to the several control and sensing layers within IMC:

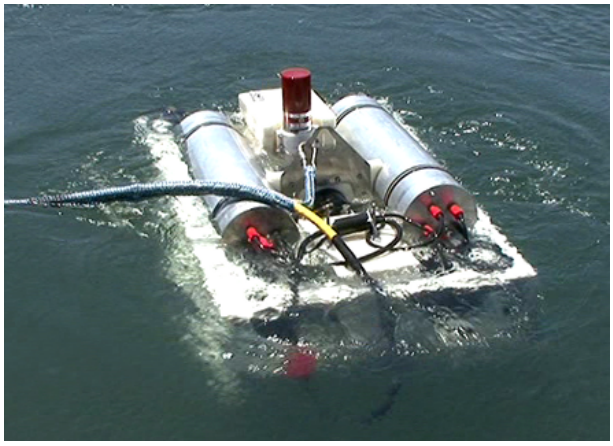
1) *Mission control messages* define the specification of a mission and it's life-cycle, for the interface between a CCU (Command and Control Unit) such as a Neptus [7] console and a mission supervisor module. A mission is a sequence or graph of maneuvers - see 3 below.
2) *Vehicle control messages* are used to interface the vehicle from an external source, typically a CCU or a mission supervisor module, for example to issue maneuver com-



(a) Seascout Light AUV

(b) Lusitânia UAV

(c) Swordfish ASV



(d) KOS ROV

Fig. 1: Some autonomous vehicles that use IMC.

mands or other external requests, and to monitor the vehicle's state.

3) *Maneuver messages* are used to define maneuvers, associated commands and execution state. In the LAUV, the simplest maneuver types are related to waypoint tracking - encoded through a *Goto* message - or loitering patterns - *Loiter*.
4) *Guidance messages* are related to the guidance used for autonomous maneuvering. In the LAUV, a guidance step generates new reference measures for the vehicle heading, depth, and velocity, in the form of a *Desired Guidance* message.
5) *Navigation messages* define the interface for reporting a vehicle's navigation state. The *Estimated State* message defines a vehicle's navigational state by the SNAME convention [8].
6) *Sensor messages* are used to report sensor readings by the respective hardware controllers. Sensor messages in the LAUV configuration shown are related to sensor readings from an IMU, a GPS, and LBL (Long Base Line) acoustic positioning system: *Euler Angles*, *GPS Fix*, and *LBL Range*, respectively, among others in diverse configurations.
7) *Actuator messages* specify the interface with hardware actuator controllers. The actuators that impact in the LAUV guidance are the fins and the thruster, interfaced through the *Set Fin Position* and *Set Thruster Actuation* messages respectively.

Fig. 2: Static and drifting buoys.

This layered control and sensing infrastructure is in line with typical control infrastructure for autonomous vehicles, and enables modular development of applications. Software components can run in logical isolation, interfacing with other

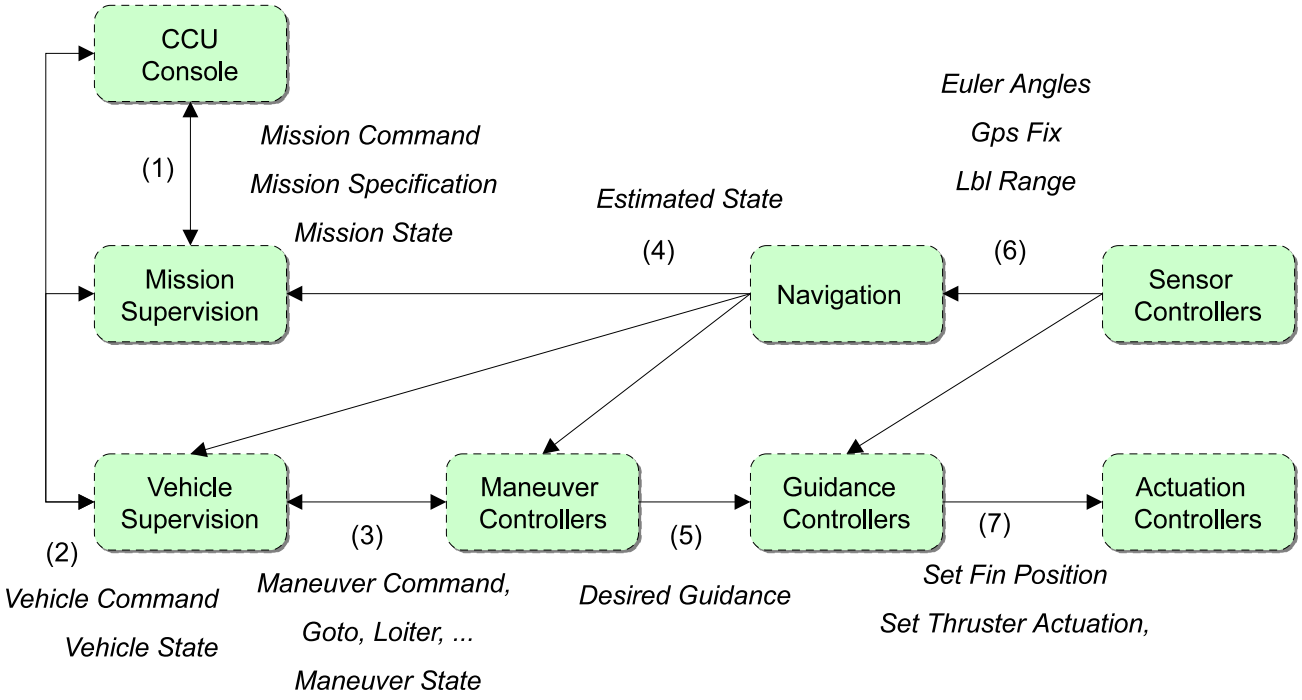



Fig. 3: IMC message flow in Seascout Light AUV.

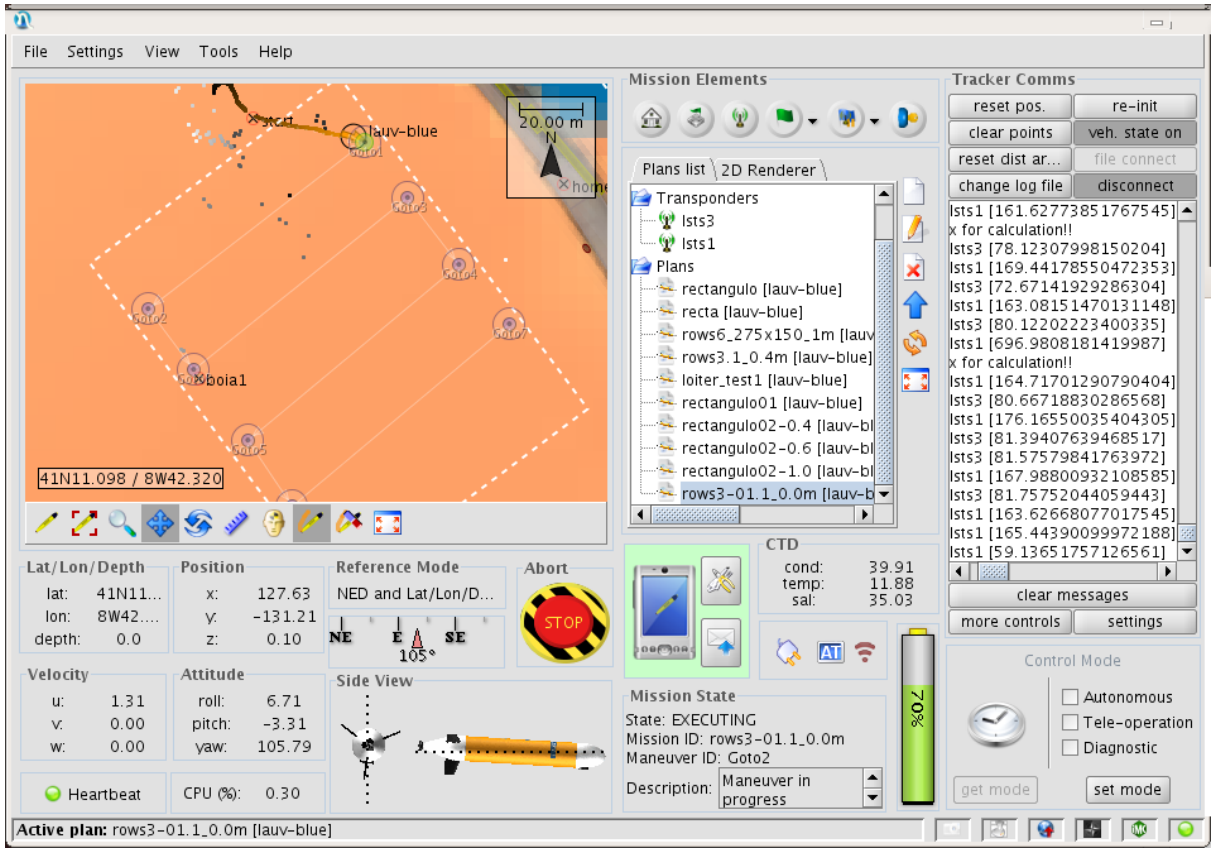



Fig. 4: A Neptus console.

modules merely through the exchange of IMC messages. The control infrastructure for autonomous vehicles implemented on-board, using the DUNE framework [9], that enables message exchange using a *message bus* abstraction, and provides transport mechanisms for external communications. Vehicles can be monitored and controlled externally using Neptus consoles [7]. A sample screenshot (for LAUV) is shown in Figure 4.

Networking of vehicles and consoles, is enabled through traditional IP-based communication-mechanisms, like raw UDP or TCP sockets, or by other means, such as the Real-Time Publish-Subscribe protocol [10], or underwater acoustic modems [11]. The drifting and static buoys being used are able to communicate data over long periods of time either by short distance multi-hop networking [12], by using ubiquitous GSM/GPRS communications, or also by communicating large bursts of stored data when a communication link can be established.

## III. Implementation

IMC defines the message entity as having an associated uniquely identifying number and consisting of a (possibly empty) sequence of data fields capable of representing fixed-width integers, floating point numbers, variable length byte sequences and inline messages (messages within messages). Integers can be signed or unsigned with sizes ranging from 8 to 64 bits. Floating point numbers have two sizes: 32 and 64 bits.

Messages are prefixed with a header and suffixed with a footer to form a packet. Header and footer entities are defined as non-empty sequences of data fields and have the same structure for all packets. Organization of data fields within one IMC packet is described in detail in Table I.

TABLE I: IMC Packet Format.

| Data Field | Size | Type |
|---|---|---|
| Synchronization Number | 2 | 16-bit unsigned int |
| Message Identification | 2 | 16-bit unsigned int |
| Message Size | 2 | 16-bit unsigned int |
| Time Stamp | 2 | 64-bit floating point |
| Message Data | Message Size | - |
| CRC | 2 | 16-bit unsigned int |

In order to transmit a message or save it to persistent storage the message has to be encapsulated in a packet and serialized. Serialization is performed by translating the data fields of the packet entities (header, message and footer) to a binary stream in the same order as they were defined. The first field of the packet header is the synchronization number, used to mark the beginning of a packet and to denote it's protocol version. By inspecting the synchronization number the recipient is able to deduce the byte order of the remaining data fields and perform the necessary conversions for correct interpretation. Using this approach, communication between nodes with the same byte order incurs in no byte order conversion overhead and communication between nodes with different byte orders only introduces the conversion overhead when deserializing packets.

The complete IMC protocol is defined in a single eXtensible Markup Language (XML) [13] document [14] that, when changed, can be verified against a XML Schema (XSD) [15]. The XML document is organized into the following sections:

1) Description of the IMC protocol, used mainly for documentation purposes;
2) List of supported types with associated size (minimum size in the case of variable length field types);
3) Serialization/deserialization rules for variable length field types;
4) List of supported units for data fields;
5) Definition of the packet header;
6) Definition of the packet footer;
7) List of message groups;
8) Definition of messages and respective fields.

In the XML definition, each message field must have at least a name, one abbreviation (used for code generation) and a type. Optionally units and range of permissible values can also be defined. Thus, a message representing Euler Angles is defined using the following XML fragment (documentation was ommited for brevity's sake):

Listing 1: XML Code Fragment

```
<message id="109" name="Euler Angles"
         abbrev="EulerAngles">
  <field name="Device Id" abbrev="id" type="uint8_t"/>
  <field name="Device Time" abbrev="time"
         type="fp64_t"/>
  <field name="Roll" abbrev="roll" type="fp64_t"
         unit="rad" min="0" max="6.283..."/>
  <field name="Pitch" abbrev="pitch" type="fp64_t"
         unit="rad" min="0" max="6.283..."/>
  <field name="Yaw" abbrev="yaw" type="fp64_t"
         unit="rad" min="0" max="6.283..."/>
</message>
```

Having a XML document describing the protocol has proven to be very practical for continuous development and testing. This happens because just after agreeing upon a specific version of IMC, two nodes can use the XML document to understand each other. This has been used thoroughly in our tests where new messages could be created as needed.

The XSL Transformations (XSLT) language [16] is used to automatically generate the IMC protocol reference documentation and optimized implementations in C++, C# and Java.

Generating native code from the XML document using XSLT has provided not only flexibility (additional programming languages are added by providing respective transformations) but also enough performance for real-time operation.

The following simplified C++ code fragment shows the result of applying a XSLT transformation to Listing 1:

Listing 2: C++ Code Fragment

```
class EulerAngles: ...
{
  // Device Id
  uint8_t id;
  // Device Time
```

```
  fp64_t time;
  // Roll
  fp64_t roll;
  // Pitch
  fp64_t pitch;
  // Yaw
  fp64_t yaw;

  bool   validate();
  Packet serialize();
  void   deserialize(Packet);
};
```

In addition to the main serialization format described above, there are two complementary serialization formats with specific intents. One is the LLF (LSTS Log Format) format, a text format used for logging IMC, amenable to direct human understanding and easier to parse directly by many standard applications e.g. Matlab, MS Excel, and custom mission review and analysis software [17]. In order to be possible to review data from past missions, the LLF format had to be independent of the originating IMC protocol description (since the message format can change over time). Our approach was to define this tab-separated log format where, for each column (message field), there is an header describing its data type, name and units to be used when representing the data.

Another format is the IMC-XML, which can be used as a simplified serialization format itself for inter-module interoperability. The main reason for this additional format is to enable the integration of web-based components and web-enabled third-party sensors into large-scale data dissemination [18] applications.

## IV. Related Work

IMC falls in the scope of similar C2 (Command and Control) protocols. In one hand it allows low level control commands like JAUS (Joint Architecture for Unmanned Systems)/SAE AS-4 [19] and CCL (Compact Control Language) [20] that provide a set of commands (messages) to control the vehicle. Additionally, IMC has a set of higher-level messages providing a level of functionality in the line of command languages like NATO's STANAG 4586 [21] and Common Control Language [22]. This type of command languages allow sending generic messages/commands and monitoring vehicle's real-time progress in detail.

Both CCL [20] and NATO's STANAG 4586 [21] target one specific type of unmanned vehicles. CCL and Common CL [22] target AUVs, being carefully designed with low-bandwidth underwater acoustic communications in mind. In this protocol, all messages are designed to fit into highly compact packets, in order that link throughput is maximized. IMC provides different serialization formats, being the most commonly used an efficient binary format that enables communication also for low-bandwidth links.

NATO created STANAG 4586 [21] to promote UAV interoperability through the specification of common interfaces and architectures for UAV control systems. The commercial Piccolo UAV autopilot system [23], that we use on our UAVs [3], also defines a standard message protocol. The intent of these is to define standards that can be used to control UAVs with distinct features being possible for them to operate jointly. In this manner, the protocols can be used to command any UAV, despite its owning force or manufacturer (considering security restrictions, of course). We consider the existence of such intersection protocols to be a major concern in the area of unmanned vehicle control and IMC is developed also having interoperability and hardware abstraction in mind. Currently, IMC provides a standardized protocol not only for controlling UAVs or AUVs but several other types of vehicles and sensors.

Like IMC, JAUS [19] targets a wide range of types of vehicles: surface, underwater, land, or aerial vehicles. It has C and Java bindings provided by at least one implementation [24]. Again, JAUS is similar to IMC in the sense that it takes an hierarchical view of vehicles as systems with subsystems, nodes, components and component instances.

## V. Evaluation and Future Work

Using IMC as the common interface between every component in our system, made it more flexible to hardware and software changes, highly simplifying the process of building support for newly built hardware. Currently, we are using IMC for communication between all our vehicles, sensors and consoles (including portable handheld consoles). The DUNE onboard software which is used by our vehicles also uses IMC as the sole inter-process communication mean. IMC has been therefore thoroughly tested in all the various real-world experiments led by LSTS, resulting in many successful AUV underwater missions, UAV autonomous flights and ROV inspections. Moreover, we have also made several hardware-in-the-loop simulations where multiple (simulated) AUVs are controlled or followed simultaneously by different operators. In these simulations each operator is able to control the actions of one vehicle at a time, being possible to hand-over control of vehicles to other operators.

On top of the hardware built at LSTS, we are also interested in using IMC for supporting vehicles and sensors from outside the laboratory. In collaboration with the Portuguese Air Force Academy (AFA) we are working in a conjoint project whose objective is to achieve cooperative multi-UAV missions using UAVs from AFA. In the scope of our collaboration with the Portuguese Navy, we will also use IMC to integrate new AUVs and sensors from this institution with the objective of creating heterogenous vehicles and sensor networks for applications like adaptive sensing and harbor security.

IMC is currently in the (stable) version 2.2.2, being still under active development to support new hardware and concepts of operation. One of our short-term goals is to build support for multi-hop message routing over different communication means. This will enable the use of IMC both for extending the connectivity of vehicle networks using homogeneous communication links and to have also network nodes that act as mobile gateways between the various supported communication means (underwater, zig-bee, wi-fi, GSM, etc), actively extending network range.

Moreover, we are considering the cases when some nodes are only connected for a fraction of the time (as happens with AUVs and other environmental sensors that operate in remote areas). Addressing this problem, we intend to incorporate DTN concepts and specifications from the Delay Tolerant Networking Research Group [25] into IMC in order to have nodes acting as data mules transparently to the rest of the network.

The way we are addressing nodes in the network is also being changed into announce strategies where capabilities and communication means are also informed to peers in the network. This can be seen as a way of establishing cooperation links between nodes which will, in the end, enable the pursuit of common goals using link-aware controllers.

## REFERENCES


[1] "Seascout Project Web Site," 2009. [Online]. Available: http://whale.fe.up.pt/seascout/

[2] H. Ferreira, R. Martins, E. Marques, J. Pinto, A. Martins, J. Almeida, J. Sousa, and E. Silva, "SWORDFISH: an Autonomous Surface Vehicle for Network Centric Operations," in *OCEANS 2007 - Europe*, June 2007.

[3] P. Almeida and R. Bencatel and G. Gonçalves and J. B. Sousa and C. Ruetz, "Experimental results on Command and Control of Unmanned Air Vehicle Systems," in *6th IFAC Symposium on Intelligent Autonomous Vehicles (IAV'07)*, 2007.

[4] N. E. Leonard, D. A. Paley, F. Lekien, R. Sepulchre, D. M. Fratantoni, and R. E. Davis, "Collective Motion, Sensor Networks, and Ocean Sampling," in *Proceedings of the IEEE, Vol 95*, January 2007, pp. 48–74.

[5] J. Sousa, T. Simsek, and P. Varaiya, "Task Planning and Execution for UAV Teams," *43rd IEEE Conference on Decision and Control*, 2004.

[6] A. Tinka, S. Diemer, L. Madureira, E. Marques, J. Sousa, R. Martins, J. Pinto, J. E. Silva, P. Saint-Pierre, and A. M. Bayen, "Viability-based computation of spatially constrained minimum time trajectories for an autonomous underwater vehicle: implementation and experiments," in *American Control Conference (ACC'09) - to appear*, 2009.

[7] P. S. Dias, J. B. Sousa, and F. L. Pereira, "Networked Operations (with Neptus)," in *MAST 2008, 3rd annual Maritime Systems and Technologies conference*, Cadiz, Spain, 11 2008.

[8] *Principles of Naval Architecture - 2nd revision*. Society of Naval Architects and Marine Engineers, 1989.

[9] "Seascout On-Board Software Distribution 0.1," 2007. [Online]. Available: http://whale.fe.up.pt/seascout/

[10] E. R. B. Marques, G. M. Gonçalves, and J. B. Sousa, "Seaware: A Publish/Subscribe Communications Middleware for Networked Vehicle Systems," in *Proc. IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC)*, 2006.

[11] E. R. B. Marques, J. Pinto, S. Kragelund, P. S. Dias, L. Madureira, A. Sousa, M. Correia, H. Ferreira, R. Gonçalves, R. Martins, D. P. Horner, A. J. Healey, G. M. Gonçalves, and J. B. Sousa, "AUV control and communication using underwater acoustic networks," in *Proc. IEEE Oceans Europe*, 2007.

[12] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[13] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, and J. Cowan, "Extensible Markup Language (XML) 1.1 (Second Edition)," World Wide Web Consortium, Recommendation REC-xml11-20060816, August 2006. [Online]. Available: http://www.w3.org/TR/2006/REC-xml11-20060816

[14] J. Pinto, P. S. Dias, G. M. Gonçalves, R. Gonçalves, E. Marques, J. B. Sousa, and F. L. Pereira, "Neptus – a framework to support a mission life cycle," in *7th IFAC Conference on Manoeuvring and Control of Marine Craft*, Lisboa, 09/2006 2006. [Online]. Available: http://whale.fe.up.pt/Papers/2006/PAPER_MCMC2006-Neptus.pdf

[15] D. C. Fallside and P. Walmsley, "XML Schema Part 0: Primer Second Edition," World Wide Web Consortium, Recommendation REC-xmlschema-0-20041028, October 2004. [Online]. Available: http://www.w3.org/TR/2004/REC-xmlschema-0-20041028

[16] "Extensible Stylesheet Language (XSL) Version 1.1," World Wide Web Consortium, Recommendation REC-xsl11-20061205, December 2006. [Online]. Available: http://www.w3.org/TR/2006/REC-xsl11-20061205

[17] P. S. Dias, J. Pinto, G. Gonçalves, R. Gonçalves, J. Sousa, and F. Pereira, "Mission Review and Analysis," in *9th International Conference on Information Fusion (Fusion 2006)*, 2006.

[18] J. Pinto, P. Dias, J. B. Sousa, and F. L. Pereira, "Large Scale Data Collection Using Networks of Heterogeneous Vehicles and Sensors," in *Proc. Oceans'09 Europe*, 2009.

[19] "Joint Architecture for Unmanned Systems," September 2008. [Online]. Available: http://www.jauswg.org

[20] R. P. Stokey, L. E. Freitag, and M. D. Grund, "A compact control language for AUV acoustic communication," in *Oceans 2005 - Europe, Vols 1 and 2*, 2005, pp. 1133–1137, Oceans 2005 Europe International Conference, Brest, FRANCE, JUN 20-23, 2005.

[21] "STANAG 4586 Second Edition, Standard Interfaces of UAV Control System (UCS) for NATO UAV Interoperability, NATO Standardization," November 2007.

[22] C. N. Duarte and B. B. Werger, "Defining a common control language for multiple autonomous vehicle operation," in *Oceans 2000 MTS/IEEE*, 2000, pp. 1861–1867, MTS/IEEE Oceans Conference and Exhibition on Where Marine Science and Technology Meet, PROVIDENCE, RI, SEP 11-14, 2000.

[23] B. Vaglienti, "Communications for the Piccolo Avionics, version 2.1.0k," Cloud Cap Technology, 2009.

[24] "Open JAUS." [Online]. Available: http://openjaus.com

[25] "Delay Tolerant Networking Research Group," 2009. [Online]. Available: http://www.dtnrg.org/wiki