Networked Operations (with Neptus)

Paulo Sousa Dias, João Borges Sousa, Gil M. Gonçalves

Abstract—**Networked operations are becoming a major trend in what concerns operational scenarios involving humans, autonomous vehicles and systems. In these operations, systems from different vendors have to interoperate. This paper will present our C4I (Command, Control, Communication and Intelligence) environment, Neptus, discuss the implementation of an inter-operability standard and illustrate the presentation with experimental results. Neptus implements the NATO's STANAG 4586 standard to support networked operations of inter-operated unmanned vehicles in a mixed initiative environment (operators in the planning and control loops). Neptus supports concurrent operations. Vehicles, operators, and operator consoles come and go. Operators are able to plan and supervise missions concurrently. Additional consoles can be built and installed on the fly to display mission related data over a network. Neptus has a Console Builder application. This facilitates the addition of new vehicles with new sensor suites to Neptus. Neptus is design to support the control of several UAVs, AUVs and ASVs concurrently.**

I. Introduction

NOWADAYS cooperative missions have great demand. Missions using heterogeneous vehicles to perform complex operations are demanding more complex infrastructures to support them. There are many issues to address when working with different kind of vehicles from different vendors, different capabilities and more important different interfaces to the end user. Interoperability is one of the most problematic issues. This is because there is no common acceptable interface. There are several interoperability efforts in several stages of development, but still not one widely accepted and used.

Let's for example think of a mission in what we use UAV (Unmanned Aerial Vehicles) with Piccolo Autopilot from Cloudcap Technology, one Woods Hole Oceanographic Institution REMUS model AUV (Autonomous Underwater Vehicle) and one LAUV AUV, [1], from our Underwater Systems and Technology Laboratory. How to have one integrated interface to deal with these three different vehicles when they have three different interfaces?

Considering this type of operations we start developing Neptus. Neptus is a C4I (Command, Control, Communications, Computer and Intelligence) framework, [2][3][4], and aims to:

- Support multi-vehicle operations with a common interface;
- Support a mission lifecycle – planning, execution, review & analysis and dissemination;
- Easy integration of new vehicles and sensors;
- Integration of custom embedded message protocols;
- Allow customization of command and control user interface; and
- Concurrently operate multiple vehicles.

Throughout this paper we will guide you in a presentation of Neptus framework and how the above mentioned aims were achieved. This will be done with a general overview of the framework and then how it supported the scenario presented.

This paper is organized as follows. Section II introduces an operational scenario on which Neptus would help implement and execute. Section III presents the Neptus framework with focus on its several modules and how they are used in the mission lifecycle. In section IV details on the integration of heterogeneous vehicles with different interfaces in a common user interface are presented. Section V presents an example scenario of a deployment of several heterogeneous vehicles using Neptus. Section VI presents some conclusions on the benefits of such common user C2 (Command and Control) interface and future directions and challenges.

II. Operational Scenario

Let's consider a heterogeneous multi-vehicle mission using three vehicles from three different vendors (Fig. 1). One UAV with Piccolo Autopilot from Cloudcap Technology. Piccolo Autopilot comes with a user interface that allows interaction with the autopilot [5]. You can configure it and send waypoints constructing in this way the airplane mission. Piccolo uses its own protocol to interface with it but it also provides libraries that we can use to interact with custom software. The second element is a well known AUV, a Woods Hole Oceanographic Institution REMUS class AUV. This also has custom interface and also provides a user interface designed for simple pre-launch

Manuscript received October 10, 2008.

This work was supported in part by AdI (Agência de Inovação), POSI (Plano Operacional Sociedade da Informação), FEDER and the Portuguese government. Paulo Sousa Dias had the financial support of FCT – Fundação para a Ciência e a Tecnologia.

Paulo Sousa Dias is with FEUP, Faculty of Engineering of Porto University, USTL/LSTS, Underwater Systems and Technology Laboratory, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal, URL:http://whale.fe.up.pt (pdias@fe.up.pt).

João Borges de Sousa is with the Electrical and Computer Engineering Department, Faculty of Engineering, Porto University, R. Dr. Roberto Frias, 4200-465 Porto, Portugal (jtasso@fe.up.pt).

Gil M. Gonçalves is with the Informatics Engineering Department, Faculty of Engineering, Porto University, R. Dr. Roberto Frias, 4200-465 Porto, Portugal (gil@fe.up.pt).

checkout, mission planning, and data reporting. It has a custom mission language and logging. In our case it's one of the first models but the interface still applies. The third element is another AUV. This is our Underwater Systems and Technology Laboratory (USTL) development and is called LAUV (Light AUV). Because is being developed in USTL, it interfaces natively with Neptus.

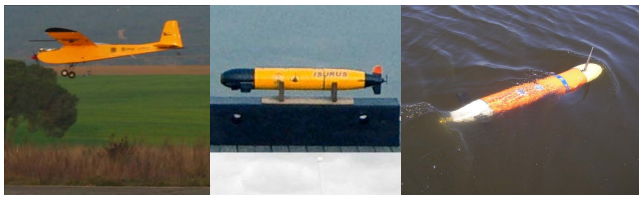

Fig. 1. UAV Lusitânia (Commercial RC model airframe Super Telemaster with a Cloudcap Technology Piccolo Autopilot | left), Isurus AUV REMUS model (middle), LAUV Blue (right)

So if we want to perform a mission with these three autonomous vehicles and don't want to jump around different user interfaces to complete the tasks one common interface has to be used.

III. NEPTUS FRAMEWORK

Neptus is a C4I (Command, Control, Communications, Computer and Intelligence) framework. It is build to support a mission lifecycle: planning, execution, review & analysis and dissemination.

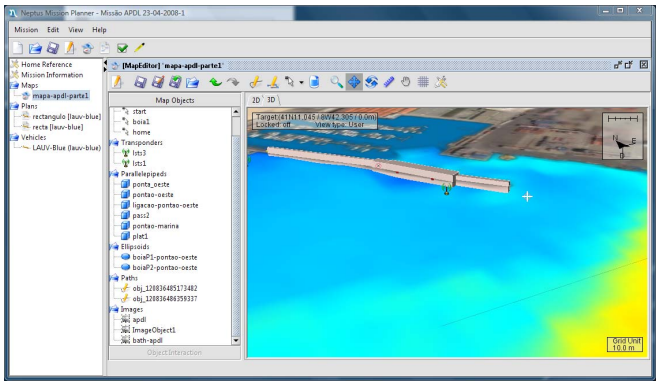

Fig. 2. Map representation

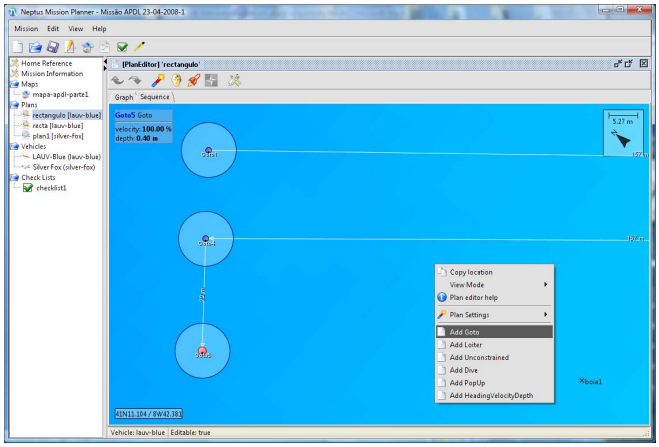

Fig. 3. Vehicle mission planning interface

The planning phase is the one that is for the preparation of a mission [6]. For this, Neptus presents a virtual mission environment construction interface. In it a virtual representation of the mission site can be created. Geo-referenced images (with optional elevation map), geometric figures, 3D models and paths can be used to enrich the virtual representation. The virtual map edition interface can be seen in Fig. 2.

After we have the virtual representation of the mission site, the vehicles' mission plans have to be prepared. The planning of each mission plan is in the form of a graph and, for ease of use, can be drawn directly into the map previously created. As seen in Fig. 3, upon click on the map where we want a maneuver to be placed, a popup menu appears displaying the possible choices of maneuvers available for that particular vehicle.

Each vehicle has a configuration file in Neptus describing several capabilities that it can provide. This encompasses its feasible maneuvers, which communication protocols it supports and more graphical information on how to represent it in the virtual world. This information can be viewed and some modified through the interface like the one in Fig. 4. This way for each vehicle only the set of feasible maneuvers are presented to the user.

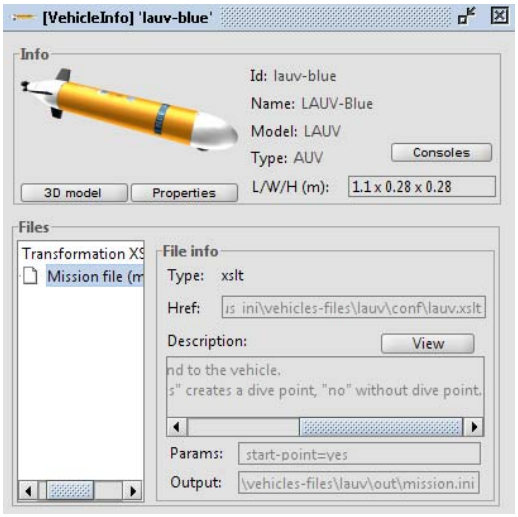

Fig. 4. Vehicle configuration interface

In the mission planning interface (Fig. 3), several plans can be created for several vehicles. At the end we get what we call a mission file which stores the maps, the vehicles' information and the several plans. All are stored in XML (eXtensible Markup Language) files.

After planning the mission we will execute it. For then on the focus is all on executing the mission. It is still possible to adjust or create more plans for the vehicles but for larger changes we can always open the mission planning interface. The main interface to the execution phase is depicted in Fig. 5. In it we can select and send plans to the vehicles and control and monitor the vehicle during the entire mission execution.

The operational console (Fig. 5) is connected to a communications interface that interfaces to several communication protocols [7]. These protocols can encompass, for e.g., Ethernet, GSM or acoustics. Then all data received is redirected to a shared data environment and then the several console components can subscribe to this data and update themselves accordingly.

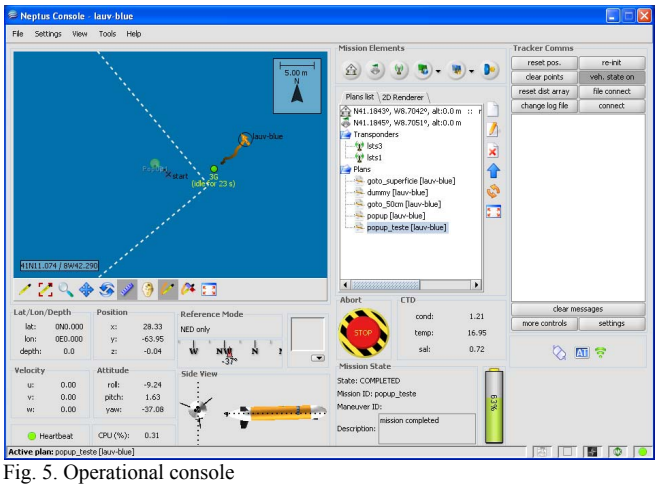

Fig. 5. Operational console

To improve the flexibility, the console layout can be edited allowing adding, removing and placing of the individual console components. In addition several monitor components can monitor specific messages to provide alarms (visual and vocal). Then the layout can be store in an XML file and then associated to vehicles. This makes also very easy to design new consoles and components allowing customization for each vehicle operational console.

To help the tactical communication between several human control units, a tactical channel is provided.

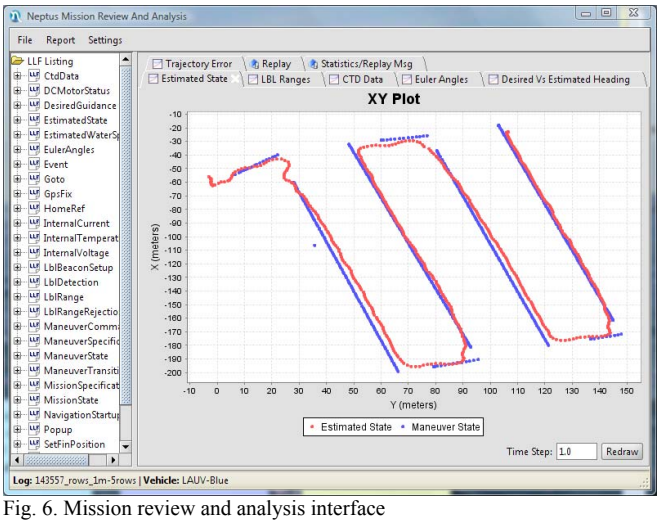

Fig. 6. Mission review and analysis interface

Upon conclusion of a mission, all collected data can be analyzed in the Mission Review & Analysis interface depicted in Fig. 6 (previous version of this module is described in [8]). This interface provides simple display with a history of logged data. There are a number of predefined graphics that are automatically generated upon opening a mission logs. Besides these predefined charts, with a couple of mouse clicks several others can be easily created.

Another useful feature is the replay capability of the logged data. This comes in two flavors. One simpler that displays the evolution of the estimated state of the vehicle in a 2D or 3D renderer similar to the one on the operational console (see top left of Fig. 5). The other flavor of this replay capability is a heavier one that consists on taking the logged messages and sending them to the Neptus communications interface. Then, by opening the mission in an operational console, the evolution of the mission events can be followed. Each replay flavor capability gives access to the mission timeline and then the possibility to pause, advance, rewind, speed up or slow down the replay.

This tool also allows the generation of a report of the mission execution with the graphics and statistics in the form of a PDF document.

Another output of this tool is a partial automated elevation map generation for map integration (like the example depicted in Fig. 7).

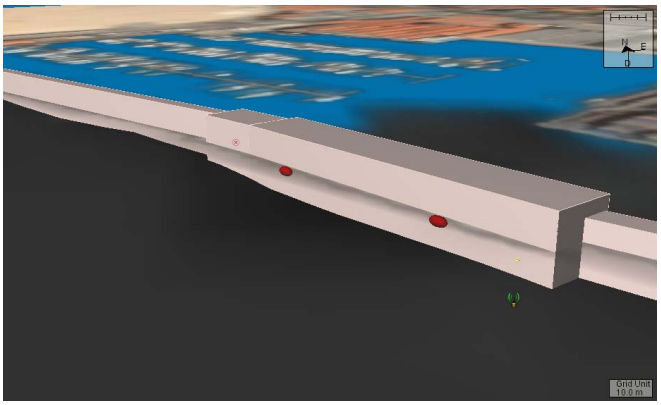
Fig. 7. Bathymetric profile in Leixões Harbor, Porto, Portugal (the two red buoys mark the surface of the water)

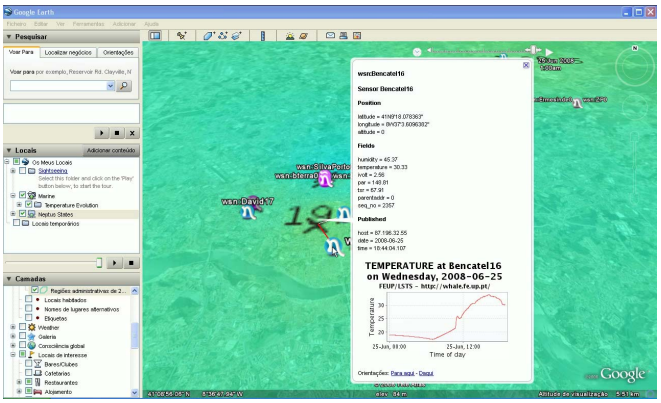

Fig. 8. Neptus Google Earth integration using data from wireless sensor networks deployment in Porto, Portugal, displaying temperature evolution as an animation or with popup dialogs for individual sensors

The dissemination interfaces is still in development. At this stage, real-time following of a mission can be achieved. On the mission site the vehicles positions and additional

sensor data can be periodically be sent to a web server. This server can be consulted by a Web browser or a more dynamically option by Google Earth (Fig. 8). Using this last option the several vehicles positions are updated and visually followed. In addition using the Time Span facility an animation of collected sensor data can be displayed.

IV. IMPLEMENTATION

In section III the several modules of the Neptus framework were introduced so you get an idea of the scope of it. Now we will focus on showing how we are able to work with the three autonomous vehicles introduced in section II with such different interfaces using the same command and control user interface, Neptus.

As seen above, planning interface provides the ability to create mission plans customized to the capabilities of each vehicle, by using a common user interface. This way, the planning phase is common to all vehicles.

For this to work, Neptus has a set of generic maneuvers, trying to map the most commonly used ones. For the vehicles of the presented scenario, four maneuvers are used: GoTo, Loiter, Popup and Tele-operation. When a mission plan is sent to a specific vehicle a transformation to the vehicle mission language has to be made. This is exactly what happens but with different implementations for each vehicle.

In the case of Isurus AUV (a REMUS model) the mission to be executed is uploaded in a RMF (REMUS Mission Format) file. The way we did this is by using XSLT (eXtensible Stylesheet Language Transformations) which is a language to transform a XML document tree into some other format (like HTML, text or another XML). If you remember from the previous section, all data that Neptus stores for the mission specification is in XML format. Because of this, we provide a way to include a series of XSLT documents that can be run and transform the selected mission plan into other format that can then be uploaded to the vehicle (in [9] is another example of this kind of use of XSLT for mission plan transformation). The user interface for this operation is depicted in Fig. 9.

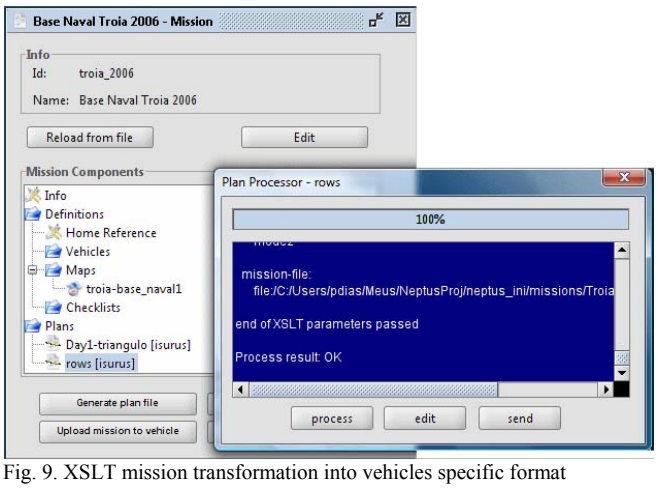

Fig. 9. XSLT mission transformation into vehicles specific format

In the execution of the mission for the Isurus, we are going to use an acoustics tracking console. This is similar in function to the REMUS Ranger but integrated in the Neptus framework allowing external tracking of the vehicle's position and the ability to abort its mission. The operational console used for this is depicted in Fig. 10. You can see on the bottom left panel the ranges to each of the acoustics transponders and from them an estimated position of the vehicle is calculated and disseminated to the other Neptus consoles connected to the network.

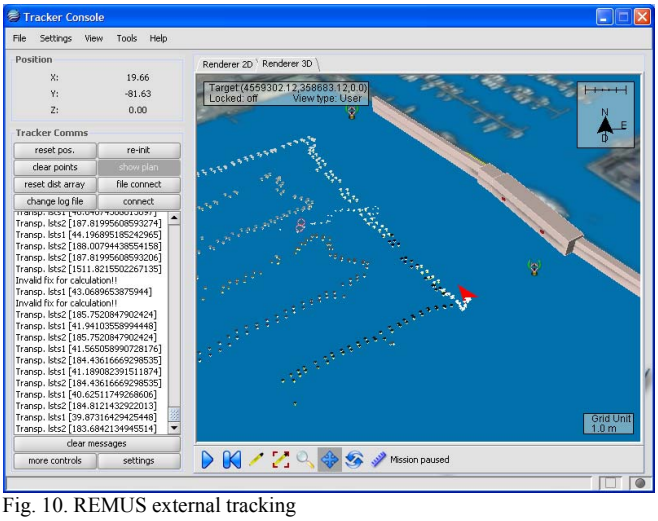

Fig. 10. REMUS external tracking

Now we present how the UAV with Piccolo autopilot is treated in Neptus. For this kind of vehicles we implemented a NATO Standardization Agreement that defines a standard interface of UAV control system (UCS) for NATO UAV interoperability, the STANAG 4586 version 2 [10][11]. This was created to promote UAV interoperability through the specification of common interfaces and architectures for UAV Control Systems (UCSs). The perceived value of UAVs in helping Joint Force Commanders (JFC) to meet a variety of theatre, operational and tactical objectives motivated these developments. By providing a common interface, it is possible to deploy the available UAVs and disseminate results at different command echelons [10]. In spite of the orientation towards UAVs, there is nothing in the standard that prevents it from being used with other types of vehicles but is much focused on UAVs.

A STANAG 4586 enabled operational console is depicted in Fig. 11. 4586 defines one a message interface consisting of one embedded protocol. An embedded protocol means that messages are transmitted with a header in which some fields provide information on how to handle the message, and on how data is encoded or decoded. The messages are separated in several functional groups in which system messages, flight vehicle command and status messages, data link or payload messages are examples.

To support 4586 a custom communications interface was developed to receive and send 4586 messages. Then several console components were created to support its message set

and logic. Despite this, the planning of the missions is still the same as the normal planning interface. But in the case of STANAG 4686 the mission plan transformation is coded in one of the operational console components. In the UAV the ground station translates the Piccolo's commands to and from 4586 messages.

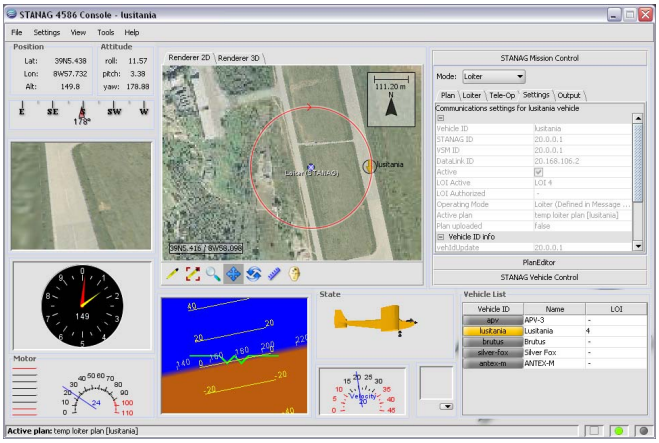

Fig. 11. STANAG 4586 ready console with UAV Lusitânia in loiter mode

In the case of the LAUV, it interfaces natively into Neptus and uses IMC (Inter Module Communication), our base message protocol [1]. The IMC is a custom embedded message protocol based on an XML configuration file, [3]. We also define STANAG 4586 in this format. IMC also has categories for the messages in similar forms of 4586. But since it's Neptus and LAUV native control and mission language every interaction with the vehicle is straight forward. LAUV operational console is depicted above in Fig. 5.

Despite each operational console is apparently showing only one vehicle data, this is not a restriction. In fact, if the estimated state (which is the position and attitude) of other vehicles arrive in Neptus they are displayed in the renderer panel as well. In one console data from more than one vehicle can be displayed in it, but one should be careful because that can confuse the human operator. Nevertheless changing the main vehicle of the console is available and this operation can be easily performed.

Here are presented three different approaches to connect different vehicles to a common command and control infrastructure. In the case of REMUS, we use a more simplistic approach where there isn't much interaction. However missions can still be planned in Neptus and the vehicle position is disseminated to the network. A second approach is by incorporating an interoperability effort for UAV provided by NATO through a standardization agreement. In this case STANAG 4586 is defined as an embedded message protocol that is described in a XML configuration file, Neptus can read it and then, with a small effort, it becomes integrated. This is true for most embedded protocols that can be defined in this way, [3]. The third approach is the native connection to Neptus which is using the IMC embedded message protocol.

V. Maritime Incident Scenario

Consider the case of a maritime incident spanning a wide geographical area. With the current technologies, tools and models, it is simply not possible to bring together, in a systematic manner, vehicles, sensors and communication networks from all over Europe to address this problem. In what follows, we discuss what could be possibly done in a maritime incident with the tools and technologies from the USTL. We consider, for this purpose the case of an oil spill resulting from the collision of two ships in the Gulf of Biscay. The year is 2015.

The response plan from the maritime authority identifies the need for the use of several types of autonomous systems: 1) UAV to track the oil spill from the air and to inspect one of the ships; 2) UAV for local communication relays; 3) drifters to track the currents; 4) AUVs for detailed inspections of a protected marine area; and 5) ASV for a close inspection of one ship.

The maritime authority accesses the USTL database to confirm that the required autonomous vehicles are available with a 12h advance notice at the USTL. Upon confirmation of availability the deployment protocol is invoked. A crew from USTL is dispatched with the vehicles to the site of operations. Meanwhile, the logistical support required for the operation of the vehicles is prepared prior to their arrival. The situation maps are shared with the USTL to develop the operations plan in coordination with the maritime authority. This takes place in transit to the site of operations. The maritime authority and the USTL crew exchange standardized plans and data for this purpose. The vehicles are unloaded and prepared for the deployment at a staging area close to the site of operations. This is done by elements from the maritime authority following the deployment protocol, with the help of the USTL crew. Operations are run by a joint team, with full integration of communications and data exchanges. The operational deployment is supervised remotely by a team from USTL in Porto, Portugal (Fig. 12 shows an example of such interface). The team not only provides technical advice, but also develops and uploads new code to the vehicles.

This is a realistic scenario that brings several challenges in several areas like control and coordinated control of vehicles and command and control interfaces.

In the case of Neptus, it is presently focused on implementing a flexible C2 (command and control) in order to achieve such operational scenario. As seen in the previous sections many of the challenges that this kind of operational scenario brings are already covered in this stage of Neptus development: heterogeneous multi-vehicle operations, inter-team communications, remote supervision of mission execution and support for mission life cycle.

Using one common C2 user interface facilitates the deployment, communication and the execution of such a scenario eliminating the need for a human operator to have to translate every necessary mission steps to a specific

interface and language of the different vehicles used. Also being able to disseminate collected data to the different operator in an automated way can improve dramatically the success of such complex mission.

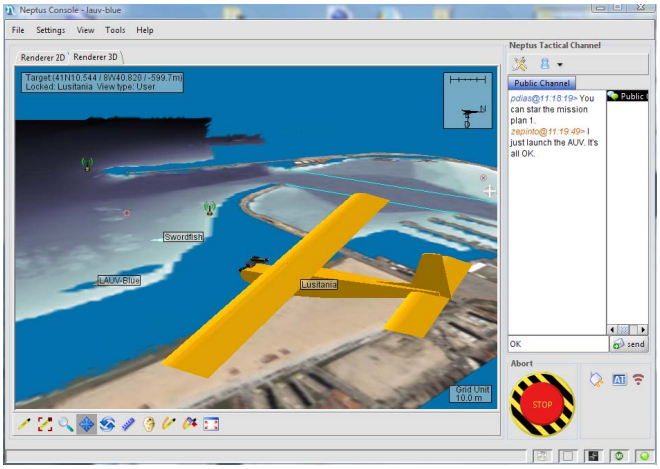

Fig. 12. Mission commander user interface with vehicles positions and a tactical channel (the UAV on foreground and on ground level on the left the AUV and a little up the ASV boat visible by their name labels, it can be seen a bathymetric profile of the harbor in bone color map)

VI. Conclusion

Network operations with heterogeneous multiple vehicles pose significant challenges in several fields. In this paper we focus on the way to interface heterogeneous vehicles, with a diversity of interfaces, under a common command and control framework. In this way more complex missions using multiple vehicles can be achieved in a more integrated way. The C4I framework, Neptus, was introduced and three approaches were presented showing different levels of integration of different interfaces from three autonomous vehicles. In this way, heterogeneous multi-vehicle missions can be achieved using one common user interface allowing, not only multiple vehicles in the network, but also multiple operator consoles with dissemination of vehicles' data and the use of a tactical channel for inter-operator communications. Additionally a more complex scenario was presented and we've shown how Neptus is approaching a state in which is covering many of the needs for such complex missions.

Future work will follow a better integration of more complex operational scenarios like the one presented in the previous section and evaluate how to improve Neptus to better support them. Also the ability to add controllers to coordinate more complex plans in an automated way is also being considered.

Acknowledgment

Neptus is a collective effort of its developers, Paulo Sousa Dias, José Pinto and Rui Gonçalves and the test and integration in the rest of USTL (Underwater Systems and Technology Laboratory) tool chain (especially Eduardo Marques and Ricardo Martins) and for that our thanks to the all USTL team from Faculty of Engineering from Porto University.

References

[1] Seascout, http://whale.fe.up.pt/seascout (October 2008).

[2] Paulo Sousa Dias, Rui M. F. Gomes, José Carlos Pinto, Sérgio Loureiro Fraga, Gil M. Gonçalves, João Borges Sousa, Fernando Lobo Pereira, "Neptus – A Framework to Support Multiple Vehicle Operation", Oceans05Europe, Brest, France, 20-23 of June, 2005, (ISBN: 0-7803-9104-7, IEEE Catalog number: 05EX1042C), 2005.

[3] José Pinto, Paulo Sousa Dias, Gil Gonçalves, Rui Gonçalves, Eduardo Marques, João Sousa, Fernando Pereira, "Neptus – A Framework to Support a Mission Life Cycle", MCMC2006, 7th IFAC Conference on Manoeuvring and Control of Marine Craft, Lisboa, Portugal, 20-22 Set., 2006.

[4] Paulo Sousa Dias, José Pinto, Rui Gonçalves, João Borges Sousa, Gil Gonçalves, Fernando Lobo Pereira, "Neptus, Command and Control Infrastructure for Heterogeneous Teams of Autonomous Vehicles (video)", ICRA2007, IEEE International Conference on Robotics and Automation, Rome, Italy, 10-14 April, 2007.

[5] Pedro Almeida, Ricardo Bencatel, Gil M. Gonçalves, João Borges Sousa, Christoph Ruetz, "Experimental Results on Command and Control of Unmanned Air Vehicle Systems", IAV2007 – 6th IFAC Symposium on Intelligent Autonomous Vehicles Toulouse, France, September 3-5, 2007.

[6] Paulo Sousa Dias, Rui M. F. Gomes, José Carlos Pinto, Gil M. Gonçalves, João Borges Sousa, Fernando Lobo Pereira, "Mission Planning and Specification in the Neptus Framework", ICRA2006 IEEE International Conference on Robotics and Automation, Orlando, Florida, USA, Mai 15-19, 2006.

[7] Eduardo R B Marques, Gil M. Gonçalves and João B. Sousa, "Seaware: a publish/subscribe based middleware for networked vehicle systems", MCMC2006, 7th IFAC Conference on Manoeuvring and Control of Marine Craft, September 20-22, Lisbon, Portugal, 2006.

[8] Paulo Sousa Dias, José Pinto, Gil Gonçalves, Rui Gonçalves, João Sousa, Fernando Pereira, "Mission Review and Analysis", Fusion2006, The 9th International Conference on Information Fusion, Florence, Italy, 10-13 Jul., 2006.

[9] Eduardo R. B. Marques, José Pinto, Sean Kragelund, Paulo S. Dias, Luís Madureira, Alexandre Sousa, Márcio Correia, Hugo Ferreira, Rui Gonçalves, Ricardo Martins, Douglas P. Horner, Anthony J. Healey, Gil M. Gonçalves, João B. Sousa, "AUV Control and Communication using Underwater Acoustic Networks", OCEANS2007, Oceans IEEE 2007, Aberdeen, Scotland, 18-21 Jun., 2007.

[10] STANAG 4586 Ed. 2 Nov 2007, Standard Interfaces of UAV Control System (UCS) for NATO UAV Interoperability, NATO Standardization Agency (NSA), 2007.

[11] AEP - 57 Volume 1, Nov 2006, Allied Engineering Publication 57 STANAG 4586 Implementation Guideline Document, NATO Standardization Agency (NSA), 2006.