Improved Situation Awareness on the Operation of Unmanned Systems through Simulation

Paulo Sousa Dias, José Pinto, João Borges Sousa
LSTS - Underwater Systems and Technology Laboratory
Universidade do Porto – Faculdade de Engenharia
Rua Dr. Roberto Frias s/n, sala I203
4200-465 Porto, Portugal
URL: http://www.lsts.pt
PORTUGAL

pdias@lsts.pt, zepinto@lsts.pt, jtasso@fe.up.pt

ABSTRACT

Unmanned vehicles are usually run in unknown environments where additional information can help improving the vehicle's path and safety. Neptus is a command and control framework targeting the operation of unmanned vehicles and leveraging simulation in order to do so.

Neptus uses a set of simulation tools allowing a better understanding of the site and how to effectively use the assets. During an initial planning stage, Neptus helps the operator assessing predicted results by simulating the execution against available environmental data. As an example, a bathymetric profile can be used for checking the depth feasibility and verify sensor coverage on the area of interest.

During mission execution, Neptus monitors the state of the vehicles and maintains a set of shore-side simulators synchronized with the real vehicles by using any states received from the vehicles (over Wi-Fi, acoustic, satellite or other communication means). When connectivity with vehicles is lost (e.g. while underwater), the operator is instead presented with a simulation of the vehicles allowing him/her to understand the predicted behavior.

This paper will describe the Neptus and toolchain exposing the simulation tools, and their relevance, that aids the operator plan to more safely plan unmanned vehicles' operations.

1.0 INTRODUCTION

Autonomous vehicles operations are more and more ready to not only speedup data collection but to improve it. With conventional means, in most cases, it would be costlier, time consuming, or near impossible to collect some data in a persistent manner. With today's pressing need for a sustained, persistent, and affordable presence in the oceans to help tackle issues such as climate change, ocean acidification, unsustainable fishing, etc. [1], and the inherent difficulties on persistent data collection with conventional means, makes the use of heterogeneous autonomous vehicles, either individually, or in a coordinated ways, very appealing (Figure 1). To take the most advantage of this growing new technology, an incremental, multi-dimensional and multi-disciplinary approach is needed. Only through synergies between scientists and engineers [2] from different fields the boundaries of science can be extended.

The number and diversity of systems used in these operations has been increasing, asking for tools not only

to analyze data, but also to command and control these heterogeneous assets. Successfully networking the existing systems and the new robotic vehicle systems, will guarantee usability in such large scale deployments. This can be done, as example, by using ships of opportunity not only to collect but also do serve as data mules, transporting data from site to site. These systems also allow the removal of humans from hazardous environments or situations.

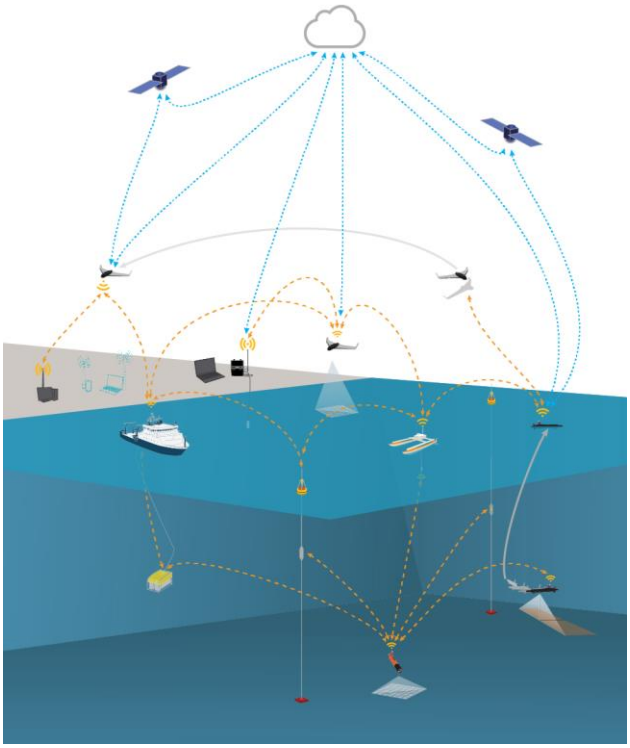

Figure 1: Networked vehicle system

LSTS has developed a toolchain for autonomous systems that encompass the onboard control software, a message protocol definition, and a command and control framework. This toolchain can be use in full or in parts, allowing a level of adaptation to the needs of the systems. This paper will introduce the LSTS toolchain in section 2. Section 3 will focus on the command and control framework features that will aid the autonomous vehicle system operator to run these assets. It will highlight the simulation tools that are available to enhance the safety and awareness, not only for planning but for the execution of mission plans with autonomous systems. Section 4 concludes this paper.

2.0 SOFTWARE TOOLCHAIN

The LSTS' toolchain is composed of three main components: DUNE, IMC (Inter-Module Communications), and Neptus. These components target at the different levels of command and control of an autonomous vehicles system. DUNE is the onboard software, IMC the message protocol, and Neptus the command and control software that interfaces with the operator in order to command the autonomous vehicle(s). This toolchain is available at http://www.lsts.pt and at GitHub [3].

2.1 DUNE

DUNE (DUNE Uniform Navigation Environment) [4] is the on-board software running on the vehicle,

which is responsible not only for every interaction with sensors, payload and actuators, but also for communications, navigation, control, maneuvering, plan execution and vehicle supervision.

It is CPU architecture independent (Intel x86 or compatible, Sun SPARC, ARM, PowerPC and MIPS) as well as operating system independent (Linux, Solaris, Apple Mac OS X, FreeBSD, NetBSD, OpenBSD, eCos, RTEM, Microsoft Windows 2000 or above and QNX Neutrino). Thanks to its modularity and versatility, DUNE does not only run in our ASVs, ROVs, AUVs and UAVs, but also in our Manta communication gateways.

DUNE's architecture, detailed in depth in [3], can be summarized as a collection of tasks, hierarchically structured, where related logical operations are isolated from each other in tasks which usually run in separate threads of execution. Tasks communicate with one another only by using a message bus which is responsible for forwarding IMC messages from the producer to all their registered receivers. Each task follows a common life-cycle and also has method handlers for all messages that it consumes.

Through a configuration text file, drivers, controllers, planners, communication means, etc., are declared with their parameters in order to be run to control a system (e.g. UAS – Unmanned Autonomous System, communication hub).

2.2 IMC

Inter-Module Communications (IMC) [4][5][6] protocol is a message-oriented protocol designed and implemented 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 heterogeneity by providing a shared set of messages that can be serialized and transferred over different means. The 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.

All IMC messages contain a header with information such as its type, version, timestamp, origin and destination. Origin and destinations are encoded as a two-part identifier: the physical component and the computational component. The check sum field is computed using the CRC-16-IBM with polynomial 0x8005 ($x^{16} + x^{15} + x^2 + 1$). The data contributing for the CRC includes all preceding header and message bytes.

The definition of the messages is maintained by LSTS and available at GitHub [3] with documentation available online [6]. New messages can be proposed to be added through GitHub pull requests and evaluated for incorporation in the definition.

2.3 Neptus

Neptus [4] is a distributed command and control framework for the operation of all types of unmanned vehicles/systems. It's based in the Java programming language and supports Microsoft Windows and Linux operating systems (it is been known to run on Apple's MacOS, though not officially supported). It supports the typical phases of an autonomous system mission life cycle: planning, simulation, execution and post-mission analysis. Can be adapted by operators and can be tailored to fit mission-specific requirements and extended by developers through a comprehensive plug-in framework.

There are two main interfaces to Neptus. One is the console and the other the MRA (Mission Review & Analysis). Neptus console (Figure 2) provides an interface to planning, monitoring and command your

autonomous systems. It is fully configurable by adding the available plugins to the console. These plugins can have a visual component (either by occupying space on the console window or through a popup dialog window, either by drawing and interacting on top of a map component), or extending functionality (e.g. extending other components, importers/exporters, or menu calls). The console can be visually setup by configuring the layout of the visual components added to the console. It allows the creation of more than one profile (or components layout) that can be changed by the operator while commanding the systems. This allows adaptation of the view to the task at hand, highlighting the components most needed for a vehicle, specific task, or status of the mission/system.

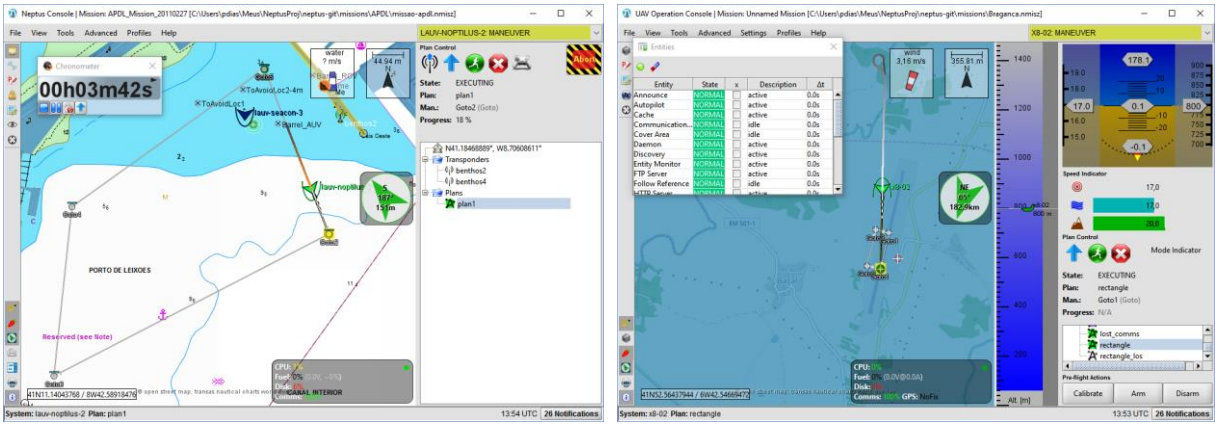

Figure 2: Neptus Consoles with different layouts (left for AUVs, right for UAVs)

The other main interface is the Mission Review & Analysis (MRA), Figure 3, which serves the purpose of post mission plans execution analysis. It is composed by a number of predefined plots (with additional plots easily created by multi-variable selection) allowing inspecting several vehicle/system variables. Additionally the mission plan can be played-back with a map visualization allowing the inspection of the path covered as well as replaying the messages produced by the vehicle to the network, or animating it in the map through available painters. Furthermore, the available visualizations are activated by the presence of the needed data. This is generally sensor data like e.g. sidescan sonar data, multibeam data, camera, or CTD. Through a plug-in interface, additional visualizations can be added, extending the functionality available.

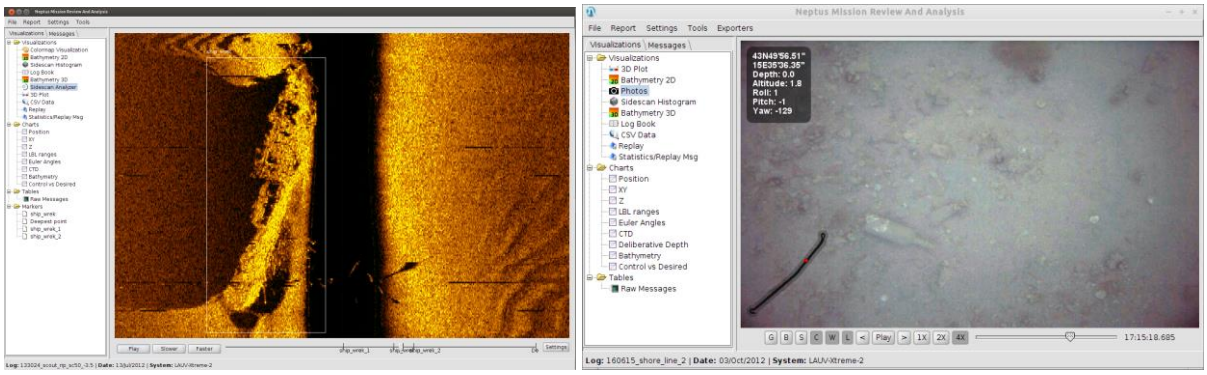

Figure 3: Neptus MRA (sidescan sonar on the left, onboard underwater camera on the right)

Not only visualizations are available, exporters can also be used to transform data into different formats (e.g. XYZ, MATLAB™, or CSV). Also through the plug-in framework this can be extended as needed. A simple PDF report can be created summarizing the mission plan progress exposing marks that were generated or

created by the operator (masks can be timestamps of interest with e.g. associated sidescan sonar image snippets).

Neptus supports planning for different vehicle types: underwater, surface, and aerial autonomous systems. This is done by having a profile of each vehicle's sensors and maneuvering specifications, allowing in this way to show/validate only supported maneuvers. This allows the operation of a heterogeneous fleet using a common interface in an integrated way.

The main Neptus communication interface is IMC, making it interoperable with any other IMC-based peer. Neptus has been used to command all our systems which correspond to very heterogeneous classes of autonomous vehicles and sensors. Despite the heterogeneity of the controlled systems, Neptus provides a coherent visual interface to command all these assets. Despite IMC being the primary communication interface, there are other available means to interface to other systems, e.g.: NMEA strings, web interfaces for situation awareness (like the one used by MBARI or NASA-AMES). Some years ago we test implemented in Neptus, as a case study, a subset of the NATO's STANAG 4586 [7] (Standard Interfaces of UAV Control System (UCS) for NATO UAV Interoperability) being able to interface with compliant UAVs.

Finally, an added feature of Neptus is the ability to create plug-ins independently of the main Neptus source and added as a compile JAR file. This way Neptus can be extended by a third party with new components with the added possibility of not sharing source code among developers (which can sometimes be a requirement for ITAR-constrained code, for instance).

3.0 AIDS TO PLANNING AND AWARENESS

Planning for autonomous vehicle systems requires some knowledge of the area. This is especially important if the vehicles don't have a lot of sensors to avoid obstacles or has low autonomy to react to environment challenges. To aid the operator and vehicles to run their mission with increase safety, Neptus provides some tools that will do just that. Planning aids as well as situational awareness are available in the console and are described in the following sections.

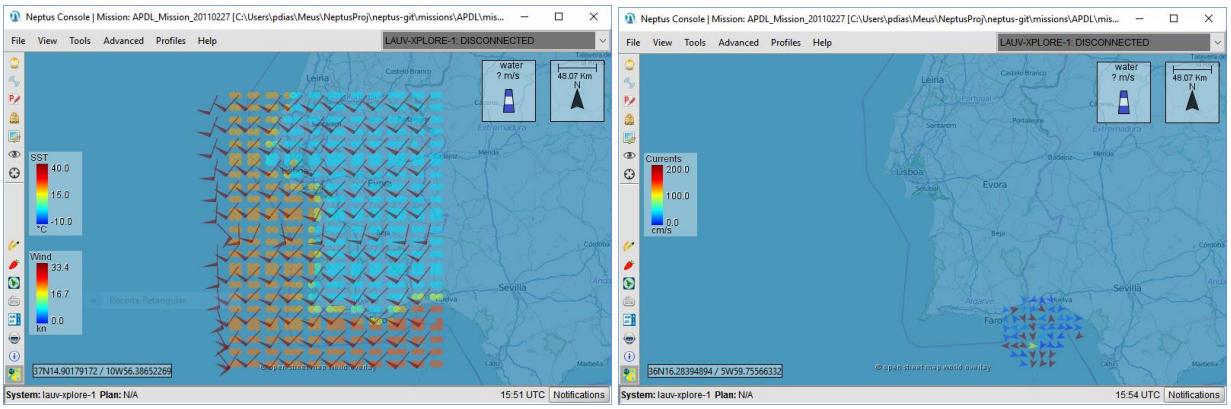

Figure 4: Environmental data (SST and Wind on the right, currents on the left)

3.1 Planning Aids

3.1.1 Environmental Data

Currents and wind information is something that is good to know if you are planning for water or areal operations. There are some components capable of loading this data, namely from NetCDF Climate and

Forecast (CF) Metadata Conventions [8] files. Figure 4 depicts such data where on the left we see sea surface temperature (SST) and wind, and on the right water currents. Other data such waves, chlorophyll are also available, with easy extension to support more variables.

3.1.2 Depth Soundings

While preparing a mission plan to be executed by underwater vehicles one important information to have is the water column depth. For this, Neptus provides tide data either by importing, or by reading TID files (tide file format from CARIS). This data can be viewed in Figure 5 which shows the tide evolution along a time window with the indication of the current time tide (which in the figure is about 1.5m at around 20h).

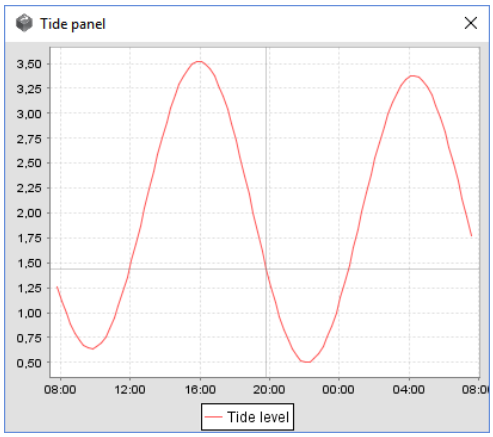

Figure 5: Tide level for Porto's Leixões Harbor

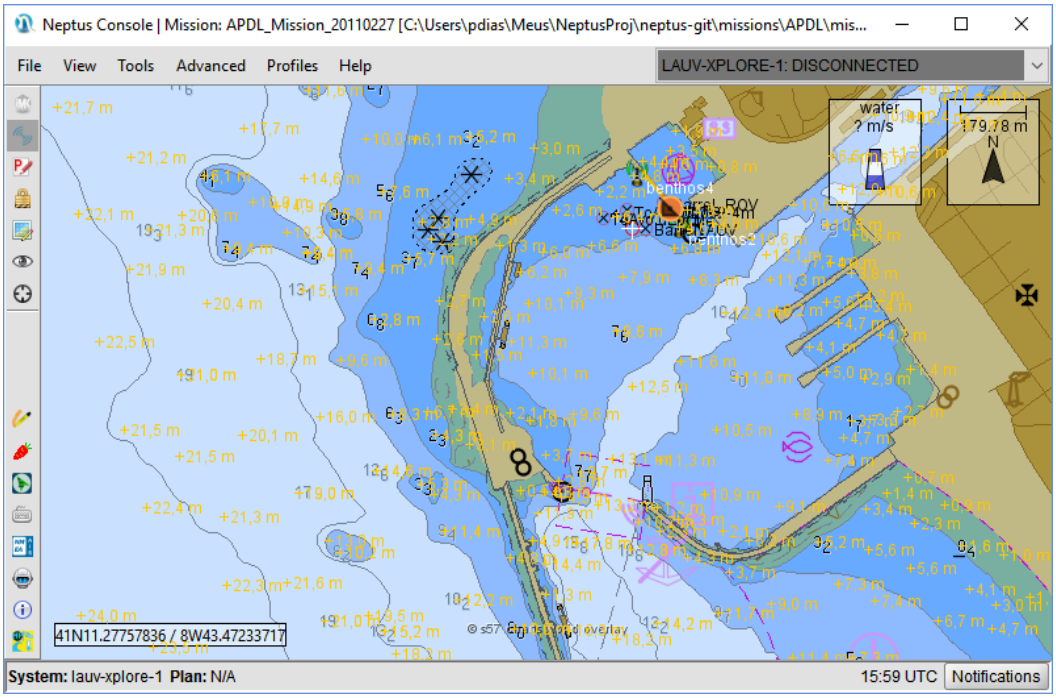

Figure 6: Depth sounding imported (in yellow) and S57 chart sounding (in black)

For a complete altimetry information we also need the depth sounding of the area of operation. This data can be inputted by several means. One is by inputting the sounding that is known or surveyed in previous operations. Neptus supports reading and displaying S57 nautical charts, which have depth soundings embedded in them. This data can be imported for the area of interest. This results in what we see in Figure 6 where in black we see the numbers of the depth soundings from the S57 charts and the resulted imported values (in yellow, which can be also added by the operator), corrected with the tide.

Alongside this, in the map editor the operator can mark features as obstacles to provide adjustments to known maps. Neptus is also in the process of integrating obstacles data from S57 and other sources like e.g. OpenStreetMaps.

3.1.3 Aids

The previous sections showed some examples of environmental data that can be loaded into Neptus. Some of this data can be used to help the planning for the autonomous vehicles mission plans.

First let us define, in the context of Neptus, what a mission plan that can be loaded and executed by the vehicles is. A mission plan defines a graph of maneuvers and plan level configurations. Maneuvers defines patterns of movements with associated maneuver level configurations and payloads settings (e.g. sidescan sonars, multibeam, and camera). In Figure 7 you can see a mission plan example with six Goto maneuvers. Their main settings can be seen in the table in the bottom of the figure, and the selected Goto1 on the right with more detail. There, the sidescan sonar setting (which is disabled for this maneuver, but enabled for the other ones) are visible.

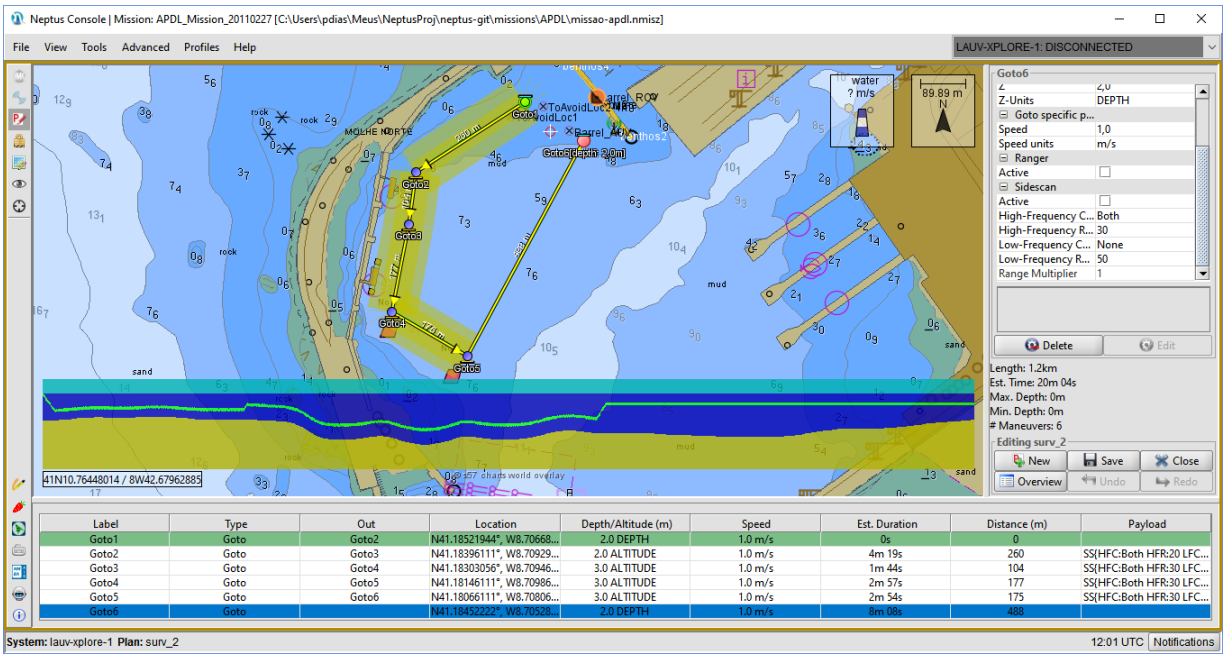

Figure 7: Planning level aids

Figure 7 depicts the planning aids in action. Right above the maneuvers table, we see a vertical cut of the water column where the plan is going to pass. In yellow you see the water bottom, the blue is the water and above the water surface in greenish. The vehicle path through the water column is the green line. We can see that the path is clear. If the path would touch either the bottom or the surface the line would turn red for that

segment alerting the operator that the plan is not safe.

Another feature that you can see in the map in Figure 7 around the plan path is the yellow shadow. This corresponds to the active sensors footprint. In this case is the sidescan sonar that is active. The operator will see the range covered by the sensor, allowing in this way to better place the maneuvers to cover the areas of interest.

The operator can also feed the plan to the vehicles' onboard simulators to better understand the behavior of the vehicle in finer detail. In this case we are feeding the plan to DUNE running in simulation profile. This profile will run DUNE with parameters target to simulating not only the navigation sensors but also, when available, the payload sensors. The data is then visible in Neptus.

3.2 Situation Awareness

During the execution of the operation the situation awareness is important especially because underwater vehicles get more vulnerable every time they resurface. For this, Neptus provides an external systems interface that can be fed with data from a number of sources. There are some sources implemented such as AIS, generic/custom NMEA strings, Web sources (e.g. Ripples LSTS Web situation awareness, NASA-AMES MTS, MBARI). Figure 8 depicts such systems. These are drawn on the map component in purple, and some information can be inspected as shown in the figure, in the same way as native Neptus systems.

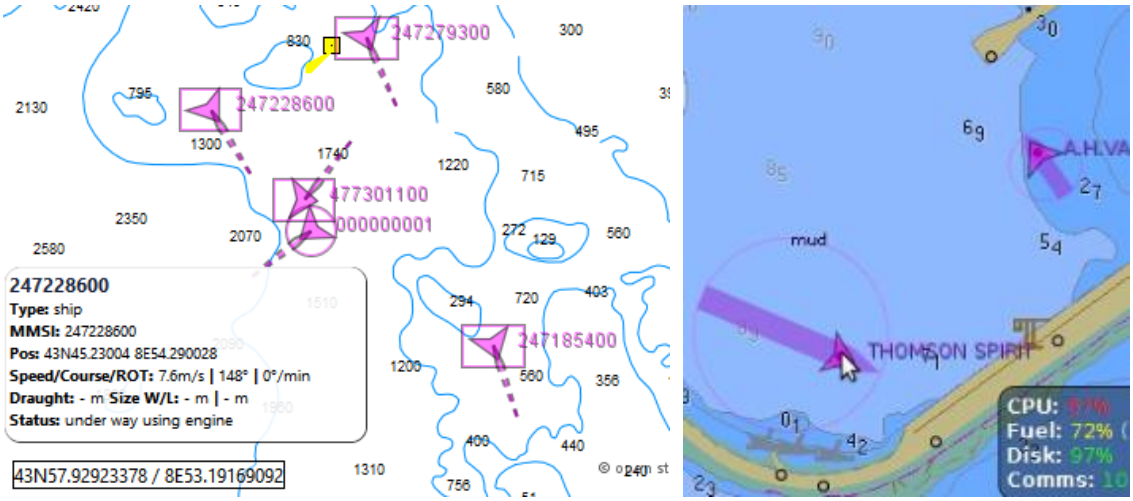

Figure 8: AIS and external systems display

With the operation of systems that can get off communication range with the monitoring station (Neptus), it is helpful to predict in some level what the system is doing and its location. In Figure 9 on the map component two systems are depicted: the LAUV-Noptilus-2 and LAUV-Noptilus-3. The 'U' shape indicates that they are underwater vehicles, and in this case they are LAUVs [9][10]. A bit bellow LAUV-Noptilus-3 (in faded color, indicating no communications) in the map, an arrow shape depicts a rough simulation of the vehicle. This is also highlighted by the text at the upper left corner of the map component indicating the time length it is simulating. By receiving system updates information, either by Wi-Fi, acoustic, GSM, or Iridium, the system simulation is adjusted. Any other system being simulated appears in this form.

These aids allows to extend the available data to support other planning and execution methods. This can extend planning onboard the vehicles with the use of planners such as with TREX [11], or EUROPtus [12] using NASA's EUROPA planner, as used while tracking sunfish at Portugal's south coast [2], NECSAVE [13] where a swarm of vehicles are tasked with diferent topologies, or MvPlanning [14], where tasks are

generated from high level objectives to a set of heterogeneous vehicles.

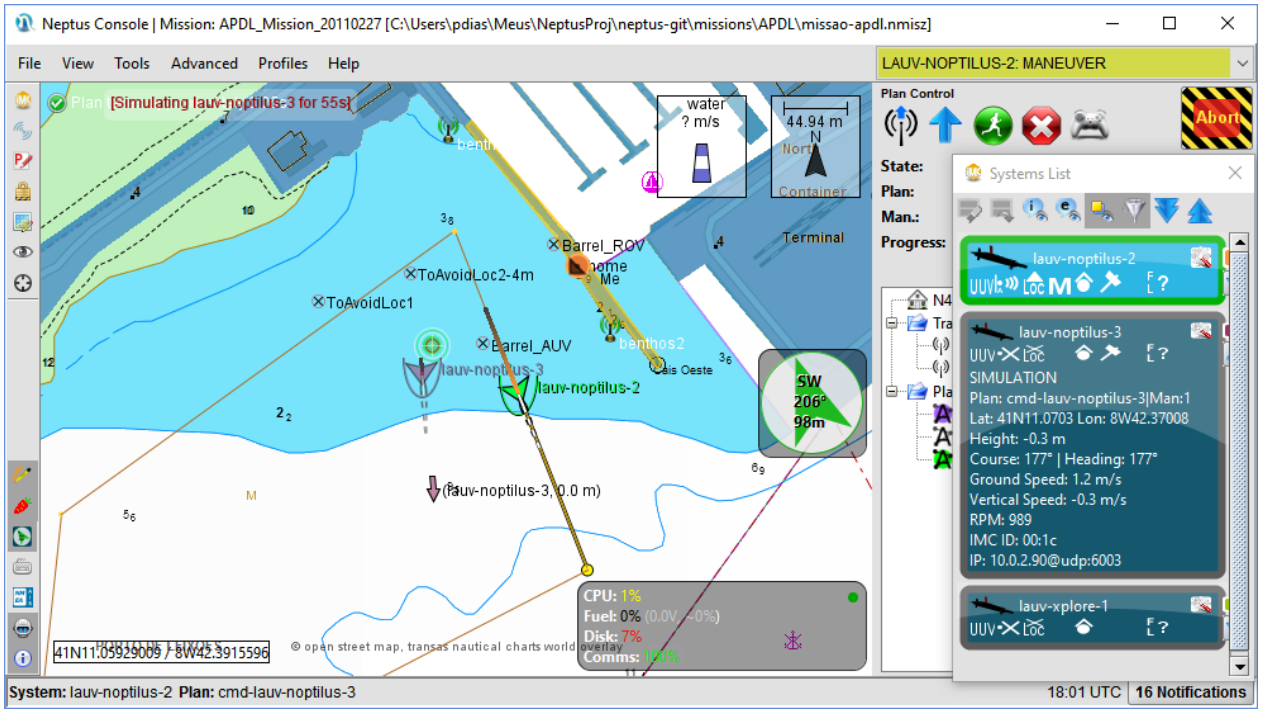

Figure 9: Execution level aids

4.0 CONCLUSIONS

We have introduced the LSTS toolchain for autonomous systems that was developed and now available freely for non-commercial purposes (commercial licenses are available on request). This paper describes some of the simulation tools available in toolchain's Neptus command and control framework that aids the operator to better plan and run autonomous systems. These aids provide environmental data that can be used at planning level to add safety while planning the mission plans. As more complex the systems, this data can also aid the execution of the mission plans either by giving more information to the operator, or by providing this data to onboard planners. Also Neptus provides situation awareness features that allow the operator to follow what is happening around the operation site in order to be able to intervene if any commanded vehicle is, or will be put, in danger.

As more and more autonomous systems are used and deployed, having tools that can provide an integrated view both in planning and execution is more than welcome. Enabling integration of data sources and allowing the operation of heterogeneous systems in the same interface, tools such as Neptus are allowing to extract more from these amazing robotic tools.

REFERENCES

[1] IOC/UNESCO, IMO, FAO, and UNDP, "A Blueprint for Ocean and Coastal Sustainability," United Nations Conference on Sustainable Development, p. 42, 2011. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/24081670.

[2] Lara L. Sousa, Francisco López-Castejón, Javier Gilabert, Paulo Relvas, Ana Couto, Nuno Queiroz,

Renato Caldas, Paulo Sousa Dias, Hugo Dias, Margarida Faria, Filipe Ferreira, António Sérgio Ferreira, João Fortuna, Ricardo Joel Gomes, Bruno Loureiro, Ricardo Martins, Luis Madureira, Jorge Neiva, Marina Oliveira, João Pereira, José Pinto, Frederic Py, Hugo Queirós, Daniel Silva, P.B. Sujit, Artur Zolich, Tor Arne Johansen, João Borges de Sousa, Kanna Rajan, "Integrated monitoring of Mola mola behaviour in space and time", PLOS ONE (Public Library of Science), PONE-D-16-10091R1, 2016.

[3] LSTS group on GitHub. [Online]. Available: https://github.com/LSTS.

[4] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. R. B. Marques, and J. B. Sousa, "The LSTS toolchain for networked vehicle systems," 2013 MTS/IEEE OCEANS - Bergen, pp. 1–9, Jun 2013. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6608148.

[5] R. Martins, P. S. Dias, E. R. Marques, J. Pinto, J. Sousa, and F. L. Pereira, "IMC: A communication protocol for networked vehicles and sensors," in OCEANS 2009-EUROPE. IEEE, 2009, pp. 1–6.

[6] IMC protocol specification and documentation. [Online]. Available: https://github.com/LSTS/imc.

[7] Rui Gonçalves, Paulo Sousa Dias, José Queirós Pinto, Gil Gonçalves, J. Borges de Sousa "A STANAG 4586 Compliant Command and Control Operational Interface for Multiple UAVs" in HUMOUS10 (Humans Operating Unmanned Systems), ISAE, Toulouse, France, April 26-27, 2010.

[8] NetCDF Climate and Forecast (CF) Metadata Conventions, [Online]. Available: http://cfconventions.org/.

[9] Luís Madureira, Alexandre Sousa, José Braga, Pedro Calado, Paulo Dias, Ricardo Martins, José Pinto, João Sousa, "The light autonomous underwater vehicle: Evolutions and networking", OCEANS - Bergen, 2013 MTS/IEEE, vol., no., pp.1,6, 10-14 June, doi: 10.1109/OCEANS-Bergen.2013.6608189, 2013.

[10] LAUV vehicle, [Online]. Available: http://www.lightauv.com.

[11] J. Pinto, J. B. Sousa, F. Py, and K. Rajan, "Experiments with Deliberative Planning on Autonomous Underwater Vehicles," in IROS Workshop on Robotics for Environmental Modeling, Algarve, Portugal, 2012.

[12] Frederic Py, Jose Pinto, Monica A. Silva, Tor Arne Johansen, Joao Sousa and Kanna Rajan, "EUROPtus: A Mixed-initiative Controller for Multi-VehicleOceanographic Field Experi-ments", International Symposium of Experimental Robotics, Tokyo, 2016.

[13] José Pinto, Manuel Ribeiro, Stefania Giodini, Patrick L'Hoir, Alessandro Sartore, J. M. Giron-Sierra, João Sousa, "Network Enabled Cooperation of Autonomous Vehicles: A Communications Perspective", in OCEANS 2017-EUROPE. IEEE, 2017.

[14] Tiago Marques, José Pinto, Paulo Dias, João Sousa, "MvPlanning: A Framework for Planning and Coordination of Multiple Autonomous Vehicles", in OCEANS'17 MTS/IEEE Anchorage, Alaska, USA, 2017, (to appear).