Mixed Initiative Control of Unmanned Air and Ocean Going Vehicles: Models, Tools and Experimentation

J. Borges de Sousa and G. Gonçalves
Departamento de Engenharia Electrotécnica e Computadores
Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto
PORTUGAL

{jtasso,gil}@fe.up.pt

ABSTRACT

The problem of mixed initiative control of unmanned air and ocean going vehicles is discussed, a formal mixed initiative control framework for networked vehicles is introduced and tools for operational deployments are presented, together with lessons learned from deployments at sea with autonomous underwater and surface vehicles. This is done in the context of the developments from the Underwater Systems and Technologies Laboratory from Porto University in Portugal.

1.0 BACKGROUND

The last decade has witnessed unprecedented interactions between technological developments in computing and communications, which have led to the design and implementation of robotic systems consisting of networked vehicles, sensors, and actuator systems. These developments enable researchers and engineers not only to design new robotic systems but also to develop visions for systems that could have not been imagined before.

Researchers and technology developers are devoting significant efforts to the development of concepts of operation for networked vehicle systems. In these systems vehicles come and go and interact through inter-operated networks with other vehicles and human operators. Surprisingly, or not, the role of human operators is receiving significant attention in the development of concepts of operation for future robotic systems. In fact, this is the reason why researchers and technology developers have introduced the concept of mixed initiative interactions where planning procedures and execution control must allow intervention by experienced human operators. In part this is because essential experience and operational insight of these operators cannot be reflected in mathematical models, so the operators must approve or modify the plan and the execution. Also, it is impossible to design (say) vehicle and team controllers that can respond satisfactorily to every possible contingency. In unforeseen situations, these controllers ask the human operators for direction.

The design and deployment of mixed initiative frameworks in a systematic manner and within an appropriate scientific framework requires a significant expansion of the basic tool sets from different areas (computation, control, communication, and human factors) and the introduction of fundamentally new techniques that extend and complement the existing state of the art. The major challenges come from the distributed nature of these frameworks and from the human factors. This is why we need to couple the development of scientific frameworks with lessons learned from field tests with human operators.

This is what we have been doing at the Underwater Systems and Technologies Laboratory (USTL) at Porto University in Portugal. At the USTL we have expertise on networked vehicles and systems, on

Borges de Sousa, J.; Gonçalves, G. (2007) Mixed Initiative Control of Unmanned Air and Ocean Going Vehicles: Models, Tools and Experimentation. In *Platform Innovations and System Integration for Unmanned Air, Land and Sea Vehicles (AVT-SCI Joint Symposium)* (pp. 9-1 – 9-14). Meeting Proceedings RTO-MP-AVT-146, Paper 9. Neuilly-sur-Seine, France: RTO. Available from: http://www.rto.nato.int/abstracts.asp.

environmental and oceanographic field studies, on interactions with oceanographic and environmental agencies, and on designing and demonstrating advanced concepts for networked vehicles systems. We have an inter-disciplinary approach for the design, implementation, and deployment of networked vehicle systems for oceanographic and environmental field studies. This involves a technology push driven by engineers at the Faculty of Engineering and an application pull driven by biologists, oceanographers, geologists and environmental experts from the Faculties of Engineering, Medicine, and Sciences. The technological push is based on a core team of Faculty members and is strengthened by an active international cooperation program with leading institutions in these fields; in the US these institutions include the University of California at Berkeley, California, the California Institute of Technology, California, and the Naval Postgraduate School, California; in Europe these institutions include the Royal Institute of Technology in Sweden, the Imperial College in London, and Verimag in France. The application pull is driven by scientists from Porto University in articulation with national and international research institutions; this articulation is done through CIMAR (the largest Portuguese center for marine research), the Portuguese Environmental agency, Porto Harbour Authority, and SMEs in the environmental services area. Research and development is targeted at developing and integrating tools and technologies for new applications.

More recently, we have also been actively working with the Portuguese Air Force and Navy in the demonstration of technologies and in the development of new vehicles, tools and concepts of operation. We report on these in this paper. First we introduce our conceptual planning and control framework. Second we present tools and technologies for the deployment of our framework. Third, we discuss operational deployments and fourth we discuss future developments.

2.0 MIXED INITIATIVE PLANNING AND CONTROL FRAMEWORK

2.1 System concepts

The idea of a system of systems seems appropriate to capture the essential aspects of operation of networked systems with mixed initiative interactions. The observation is that the components in the network are part of a system, within which new properties arise, some of them as planned, some of them emergent, and eventually leading to unpredictable behaviors. Moreover, since communication is not necessarily available, or instantaneous, the state of the system – a network of systems with evolving structure – is not accessible.

In a system of systems, a significant part of the “system” is embodied not as physical devices, such as sensors, robotic devices or communication networks, but as software applications which may be mobile, in the sense of migrating from one computer unit to another one, as part of the evolution of the system. This poses challenges to robotics, control, computer and communication scientists.

2.2 Control concepts

We use the concept of manoeuvre – a prototype of an action/motion description for a vehicle – as the atomic component of all execution concepts. We abstract each vehicle as a provider of manoeuvres and services. A simple protocol based on an abstract vehicle interface governs the interactions between the vehicle and an external controller: the external controller sends a manoeuvre command to the vehicle; the vehicle either accepts the command and executes the manoeuvre, or does not accept the command and sends an error message to the controller; the vehicle sends a done message or an error message to the controller depending on whether the manoeuvre terminates successfully or fails. This protocol facilitates inter-operability with other platforms. Actually, the same protocol is used on-board each vehicle for autonomous execution control. In this case there is a local controller, the mission supervisor that assumes the role of the external controller in this interaction protocol.

We use our vehicle abstractions in multi-vehicle controllers that may reside in some remote locations or in some other vehicles. This leads to different control configurations and strategies. We do this in the framework of high level hybrid automata. By this we mean hybrid automata whose states include sets of vehicles and links to these vehicles. Space limitations preclude discussion of how this organization and control structures can be fully utilized in a complex military scenario.

2.3 Control architecture

The control architecture consists of two main layers: multi-vehicle control and vehicle control. Each layer, in turn is further decomposed into other layers. The vehicle control architecture is standard for all the vehicles (see Figure 1). The multi-vehicle control structure is mission dependent.

The vehicle control architecture consists of the following layers: low-level control, manoeuvre control, vehicle supervision and plan supervision.

The concept of manoeuvre plays a central role in the USTL control architecture: it facilitates the task of mission specification, since it is easily understood by a mission specialist; it is easily mapped onto self-contained controllers, since it encodes the control logic; and is a key element in modular design, since it defines clear interfaces to other control elements. Depending on the type of vehicle we can find manoeuvres like: Hover, FollowTrajectory, FollowWall, Surface, Goto, Rows, Tele-operation and others. Each manoeuvre controller is encoded as a hybrid automaton. Each transition is labelled with a guard, the condition under which the transition can take place, and an event, the message sent out when the transition is taken; the two are separated by a / in the figure. The manoeuvre controller takes as input a manoeuvre specification, sends low-level control commands to the actuators in continuous time, and signals back to the vehicle supervisor the success or failure of the manoeuvre. We allow the operator to interact with the execution of some manoeuvres. This is encoded in the manoeuvre automaton as transitions and resets of variables. There is a library of manoeuvres and of manoeuvre controllers; and the addition and deletion of manoeuvre to the library does not require changes to the control architecture.

The vehicle supervisor controls all of the onboard activities and mediates the interactions between an external multi-vehicle controller or the internal mission supervisor and the manoeuvre controllers. The vehicle supervisor is encoded as a hybrid automaton. It has an internal state representing the state of all physical components in the vehicle and of all software modules. Mixed initiative control is allowed by the enabling and disabling of transitions in the automaton. This can be done by an operator if communication with the vehicle is available. The supervisor accepts manoeuvre commands (or commands to abort the current manoeuvre) and passes the manoeuvre parameters to the corresponding manoeuvre controller for execution, and signals back the completion or failure of the manoeuvre. The basic structure of the automaton encoding the vehicle supervisor is very simple. It has 4 states: Init, Exec, Error, and Idle. The vehicle supervisor is initially in the state Idle. Upon the reception of a manoeuvre specification it creates a manoeuvre controller if the enabling condition is true. When the manoeuvre is completed it goes to the Idle state again, otherwise, the transition to the Error state is taken, and it sends an err_code event to the plan supervisor, and the plan fails. The vehicle supervisor keeps a state of all the components of the vehicle and also encodes

The plan Supervisor commands and controls the execution of the mission plan. It commands the vehicle supervisor to trigger the execution of a manoeuvre specification and waits for the acknowledgment of its completion, or for an error. When it receives the acknowledgement, the plan supervisor selects the next manoeuvre to be executed. The process is repeated until the plan is successfully terminated, or it fails. The plan also has provisions for mixed initiative control by allowing the operator to enable and disable some of the transitions.

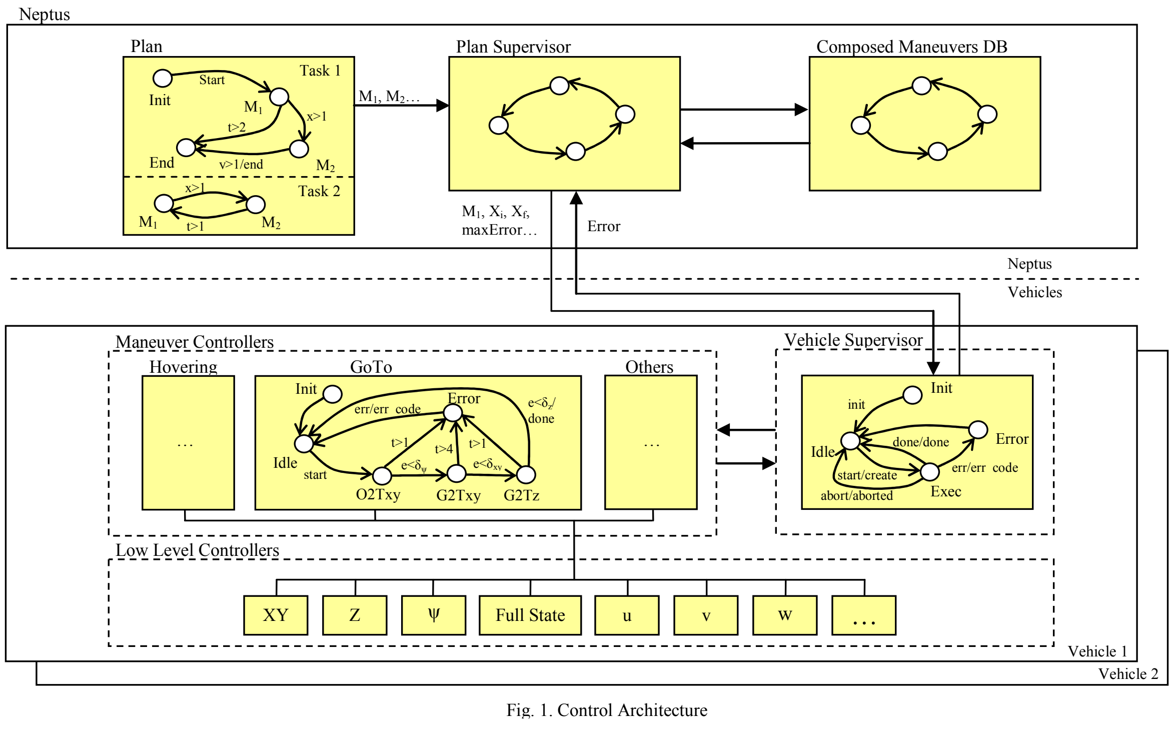


Figure 1 Vehicle control architecture

Next, we briefly describe two multi-vehicle control structures for specific applications.

In [15] we present a verified multi-vehicle control structure for executing multi-vehicle team coordination algorithms expressed as a formal specification in the automata framework. We structure the space by first decomposing it into waypoint generation and online execution control. The waypoint generation procedure generates the waypoints for a team to search for the minimum of a scalar field under dynamic and communication constraints and in accordance to a given optimization algorithm. Execution control is organized as a three level hierarchy of team controller, supervisor, and manoeuvre controller. It is shown that the controller implementation is consistent with the system specification on the desired team behaviour. This is done in a modular fashion by layering the execution control and designing each layer to ensure that the controllers produce guaranteed results under the assumption that the controllers at the adjacent layers also produce guaranteed results.

In [9] we present a multi-vehicle control structure for collaborative missions of Suppression of Enemy Air Defences. We address this problem by structuring the design in two layers: an off-line plan, and an online execution control. The planning procedure first invokes an algorithm that selects the targets and prescribes the order in which they must be attacked, keeping risk below a given threshold. As a side effect, the algorithm selects risk-minimizing nominal flight paths. The procedure next groups the targets into sub-tasks, assigns a UAV team to each target, and specifies spatial and temporal coordination points so that the target attack order is maintained. The execution control is decomposed into a hierarchy of task, sub-task, and sub-team controllers and vehicle controllers (which determine the actual flight path and weapons release). These controllers are described as interacting hybrid automata in the framework of dynamic networks of hybrid automata. The execution control framework accommodates different types of tasks of varying complexity. Given a task specification it automatically creates the initial structure of controllers, including the links or communication channels connecting them, and the information structures for task execution. It adapts the initial structure to changes in the world and to the execution requirements. It does this by creating and/or removing links and controllers while preserving structural and task invariants.

Links play an important role in this framework. A Link variable is basically a pointer to another component. Links are unidirectional: if A communicates to B it has a link to B; B is required to have a link to A to communicate with it. The components, organization, and evolution of the execution control framework are briefly described.

There is a task controller for each Task, one sub-task controller for each sub-task, one sub-team controller for each sub-team, and one vehicle controller for each UAV. Controllers execute specifications. Specifications are separated from the control code for re-use and modularity. There is one specification type per type of controller. The UAV controllers are non-mobile and reside onboard the UAV. The other controllers are mobile, i.e., they have a link to a physical location and we can change this location. The location of a mobile controller is part of its state.

The multi-vehicle control structure is a tree graph of controllers (see Figure 2 from [15]): the nodes are the controllers and the edges are the links connecting them. There are four layers in this tree, one layer for each type of controller: task, sub-task, sub-team and UAV respectively. The root node is the task controller. It is linked to the sub-task controllers. Each sub-task controller is linked to the Attacker and Reserve sub-team controllers. Each sub-team controller is linked to the UAV controllers in the sub-team.

The task, sub-task and sub-team controllers follow the same patterns of coordination. This is because the control structure is organized as a tree. Each controller has in its state a few coordination variables and links to each of its dependants (controllers). The coordination variables describe the state of its dependants and the state of execution of the controller specification. The controller receives state updates from each dependant, updates the coordination variables, and commands its dependants accordingly. This allows for distributed decision making. The task controller coordinates execution dependencies and task failures. The sub-task controller coordinates the Reserve and Attacker sub-teams. The sub-team controllers coordinate the manoeuvres of each UAV in the sub-team.

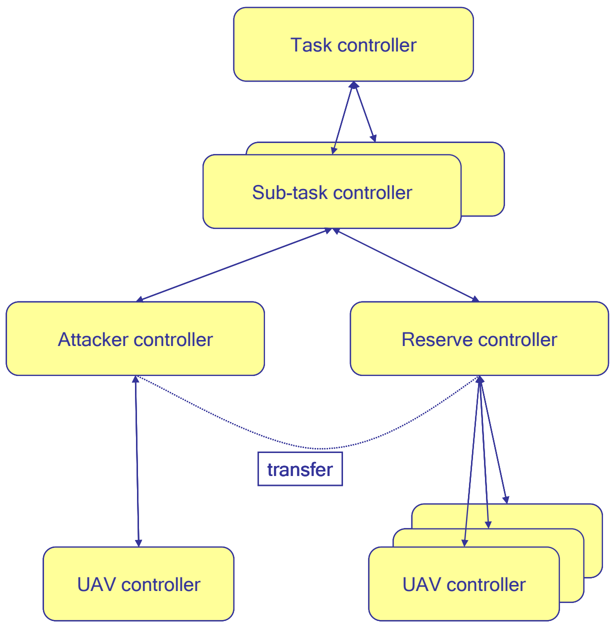


Figure 2 Multi-vehicle control architecture

3.0 TOOLS AND TECHNOLOGIES

3.1 Introduction

We are using the Neptus/Seaware/DFO/Dune tool set, developed at USTL, to support the implementation of this planning and framework. These tools and the technologies they use are described next.

3.2 Neptus command and control framework

Neptus is a distributed command and control framework for operations with networked vehicles, sensors, and human operators [3,4]. The interactions with human operators are classified according to the phases of a mission life cycle: world representation; planning; simulation; execution and post-mission analysis. There are applications for world representation and modelling, planning, simulation, execution control, and post-mission analysis.

The Neptus design facilitates mixed initiative interactions with heterogeneous vehicle systems over inter-operated networks. First, Neptus applications are built around a set of truly reusable software modules with special emphasis on modules for Graphical User Interfaces (GUI) and data management. Second, Neptus embodies the abstractions of our command and control framework with XML abstract data types and eXtensible Stylesheet Language Transformations (XSLT) technology; this leads to vehicle-interoperability and to the standardization of interactions with human operators. Third, Neptus allows the user to configure operating consoles for different vehicles. Fourth, Neptus uses the Seaware middleware framework for communications in a distributed environment; this enables the transparent inter-operability of communication networks.

We have adopted XML for data representation in Neptus. This enables us to define a grammar for every data file and to specify the exact file format to be expected from potential users. XML can also be filtered and transformed into different formats like text, HTML or any kind of native mission file formats for existing vehicles. An eXtensible Stylesheet Language Transformations (XSLT) stylesheet gives the transformation rules from XML to the vehicle's mission language. This facilitates vehicle inter-operability and the integration of new vehicles. When we add a new vehicle to Neptus we must specify the vehicle's command interface in XML format.

There is a set of modular software components – Map Editor, Mission Planner, Mission Processor, Console Builder, Variable Tree, Renderer2D, Renderer3D – which can be used by developers to build Neptus applications. This is especially useful when it comes to integrate new vehicles in the framework.

The Mission Map Editor (MME) component is a GIS-like application that allows the creation and manipulation of three dimensional world maps. Maps are stored as XML files.

The Mission Planner (MP) component is a top-level application for single and multi-vehicle mission planning. Mission planning is vehicle specific. There is a library of vehicle models and interfaces. Mission plans are stored as XML files. A mission plan is composed of world maps (links to other XML files), vehicle mission plans (a graph with nodes representing manoeuvres and transition conditions among them) and additional data like local information, checklists for operations, and specifications for tests.

The Mission Processor (MProc) component translates Neptus mission files (XML) to the native formats used by different vehicles. We use this module to generate vehicle-specific mission files. These are then uploaded to a vehicle for execution.

There are vehicle-specific and mission-specific operational consoles. We use the first to supervise single vehicle operations and the latter to supervise multi-vehicle operations. We use the Console Builder (CB) component to build operational consoles and to tailor these consoles to each vehicle and to each operator. Initially the CB application presents an empty window which serves as a canvas for adding various visual components. The visual components are then connected to variables that might be available on the network. These include, for example, the state of the vehicle, or the motor RPMs. The configurations for each console are saved as XML files for reuse.

There is a Variable Tree (VT) module in every console. This module stores the incoming network data and provides generic access to data values. The variables are stored in a tree structure. We use this tree structure to trigger typed events and the updates of dependant variables when the value of a given variable is updated. This simple scheme allows the easy specification of system alerts by defining scripts that run whenever a variable or a variable domain is updated.

The two dimensional (R2D) and three dimensional Renderer (R3D) components are used to visualize the motions of the vehicles and the state of the world. These can be used simultaneously. The Renderer components are connected to VT module in each console to subscribe to the data for visualization. The R3D version proved extremely useful to support the human operator in remotely operated vehicle (ROV) operations. This is because video from the vehicle does not provide enough visual clues for tele-operation in low-visibility areas. The R2D module is quite useful to supervise operations that take place over a large area. Additionally, R2D is also used for map edition, allowing the user to interact with the existing objects (images, paths, marks, etc.).

The Mission Review and Analysis (MRA) component provides support for the analysis of mission data. This includes provisions for replaying missions in a virtual world and also to graph mission variables.

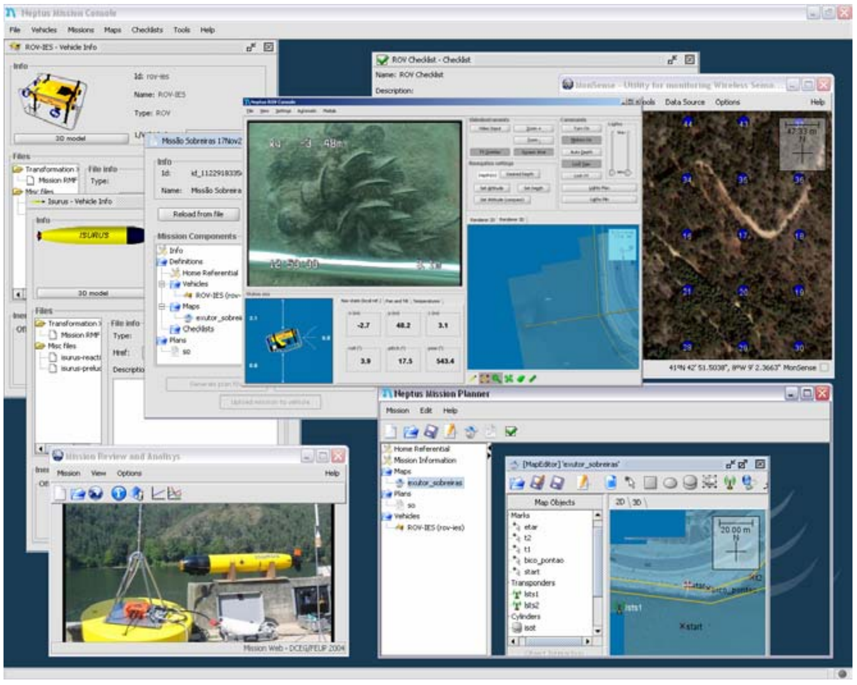


Figure 3 Neptus command and control framework

The Neptus design 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 implements a subset of the NATO standard STANAG 4586 [12] for communications with unmanned air vehicles.

3.3 Seaware publish/subscribe framework

Seaware is a publish/subscribe framework for dynamic and heterogeneous network environments oriented to data-centric network computation [7]. Publishers and subscribers communicate transparently to any node that is registered in the network. Nodes can either be vehicles that publish sensor data and receive operator commands or consoles that subscribe to the data provided by vehicles and sensors and publish operator commands. Seaware uses the RTPS (Real Time Publish Subscribe) protocol and other forms of network transport.

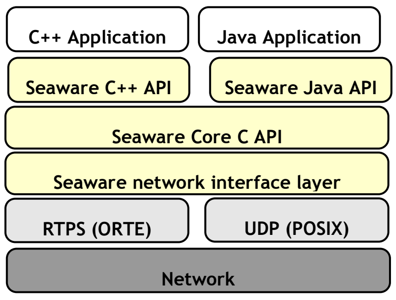


Figure 4 Seaware publish/subscribe framework

3.4 Dune onboard software infrastructure

At the core of DUNE is a platform abstraction layer, written in C++, enhancing portability among different computer architectures and operating systems. DUNE can be extended in the native compiled programming language C++ or using an interpreted programming language such as Python or Lua. The platform abstraction layer accommodates support for multiple operating systems: Linux, Sun Solaris 10, Apple Mac OS X, FreeBSD, NetBSD, Microsoft Windows 2000 or above and QNX 6.3. Different CPU architectures are easily handled by the platform abstraction layer. Currently DUNE's runs on CPUs like the Intel x86 or compatible, Sun SPARC, Intel XScale/StrongARM and IBM PowerPC.

DUNE attains loose coupling between components by partitioning related logical operations into isolated sets (or Tasks in DUNE's nomenclature). Tasks are executed in a concurrent or serialized fashion and may also be grouped into single concurrent or serialized execution entities, usually several concurrent and serialized execution tasks will coexist within DUNE. Concurrency is a configuration parameter not constrained by the task implementation. It is viable to enable or disable the concurrency aspects of any task at run-time. In a similar manner, tasks can be started or terminated at any time.

Communication and synchronization between tasks is achieved by the exchange of messages using a lock-free/wait-free message bus analogous in design to the Observer pattern. The internal message format is also used for logging purposes and communicating with external software modules over network links. As no restrictions are imposed by DUNE's core on the source and destination of a message, tasks may be scattered among different networked computers working together to achieve a common goal.

3.5 DFO framework for embedded control software specification

We are currently developing a programming language called DFO ("Data Flow Objects") for embedded control software specification [11]. DFO allows the specification of objects with "data flow". The aims of DFO are basically two-fold: firstly, to allow the definition of "data flow objects" with sound and clear semantics; secondly to provide good performance and abstract details of native support in a particular platform, operating system or use of programming language for specification of "user code".

Initially, DFO is being developed to support a set of core language constructs for: input-output data flow; mode switching (in the sense of a finite state machine); and object composition in sequential, concurrent or hierarchical fashion. The core properties we wish to attain from derived programs are determinism in execution and high performance and low memory footprint. Subsequent efforts will try to cope with distributed execution across a network, a greater challenge in terms of requirements and operational semantics.

Our primary application of interest is the use of DFO in autonomous and semi-autonomous vehicle systems. They provide a complex embedded control software scenario and are prone to a modular and hierarchical organization with highly dynamic operation. We are working on the deployment of our control architecture model with the help of DFO and Dune.

4.0 OPERATIONAL DEPLOYMENTS

4.1 Vehicles

We have been designing and operating unmanned ocean and air going vehicle systems since the early nineties. Figure 5 depicts some of the vehicles that we are currently operating.

Isrurus is a modified version of a Remus class (Remote Environment Measuring UnitS) class AUV, built by the Woods Hole Oceanographic Institution, MA, USA, for low cost and lightweight operations in coastal waters. Isurus has a torpedo shaped hull about 1.6 meters long, with a diameter of 20 cm and weighting about 35 kg in air. The maximum forward speed is 4 knots, being the best energy efficiency achieved at about 2 knots. The maximum operating depth is 200m. For navigation Isurus uses a PNI TCM2 digital compass and Long Baseline acoustic beacons (20-30 Khz). In the standard configuration, Isurus is equipped an Ocean Sensors 200 conductivity, temperature, and depth sensor, a Wet Labs optical backscatter sensor, a Marine Sonics side scan sonar and an Imagenex altimeter. It is also configurable with an Acoustic Doppler Current Profiler. The communications suite includes a Benthos acoustic modem and a 802.11 compliant radio.

Lusitânia is an UAV based on a commercial remotely controlled model airframe equipped with a OS 91-FX, 15cc, 2.9HP, 2 stroke engine. Lusitânia is equipped with the Piccolo autopilot (provided by CloudCap Technology [10]), with a small video camera and with Telos motes (with meteorological sensors optimized for use on a UAV platform). The camera can be remotely controlled, and provides the operator with a video feed in real-time. This is done through a 2.4GHz wireless transmission system with a range of 8Km.

IES is a modified Phantom 500 ROV model from Deep Ocean Engineering. The innovations include on-board power and computer systems (to minimize the number of wires in the tether cable), tele-operation and tele-programming modes and an integrated navigation system which fuses data from an external acoustic system and internal navigation sensors. The inspection package includes a video camera (Inspector, zoom 12:1) mounted on a pan and tilt unit (Imenco) with 600W of light (DSP&L). The navigation package includes a Doppler Velocity Log (Argonaut/Sontek), an Inertial Unit (HG1700 /Honeywell), a Digital Compass (TCM2/PNI) and acoustic beacons (20-30KHz).

Swordfish is a 4.5m long ASV based on an ocean-going catamaran (200kg) equipped with two Seaeye SI-MCT01 thrusters and a docking station for AUVs. Power is provided by batteries. It has a GPS unit and a miniature IMU for navigation. Swordfish is a powerful communications node with Wifi and broadband radios, GSM and a Benthos acoustic modem for underwater communications. The standard payload includes a wireless video camera and a distributed weather station based on a Mote sensor network. It is used both as a gateway buoy for underwater communications and as docking station for autonomous underwater vehicles.

Antex-M is a family of UAV platforms developed by the Portuguese Air Force Academy (AFA). ANTEX-M is a 6 meter wingspan platform with a 220cc, 22HP, 2 stroke 3W engine for a payload weight exceeding 30kg. ANTEX-M X02 is a 1:2 scale model of ANTEX-M with a 15cc, 2Hp, 4 stroke Saito100 engine, for a maximum payload takeoff weight of 7Kg. The ANTEX-M UAV family has a standard computational and sensor configuration. It is configured to fly with two different autopilots (Picollo and Micro-Pilot).

KOS is a modular ROV for underwater inspection and intervention in three basic configurations. It is made of composite materials to reduce weight and for added performance. It has advanced thrust and power control for operations in difficult environments. Dimensions: 120 x 70 x 90 cm; weight: 90 kg; 5 Seaeye SI-MCT01 Thrusters; max operating depth: 200m; Power: 3Kw. It has the same inspection and navigation packages installed on the IES ROV plus a 2-degree of freedom robotic arm for interventions.

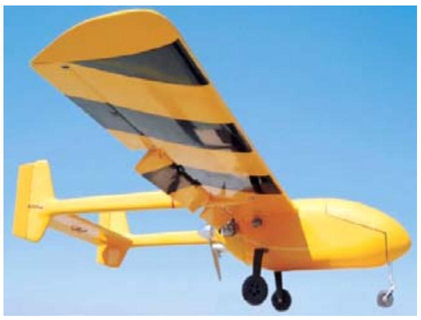
Our most recent vehicle, the Light Autonomous Underwater Vehicle (LAUV) is a prototype of a low-cost submarine for oceanographic and environmental surveys. It is a torpedo shaped vehicle made of composite materials (110x16 cm) with one propeller and 3 (or 4) control fins. The LAUV has an advanced miniaturized computer system running modular controllers on a real-time Linux kernel. It is configurable for multiple operation profiles and sensor configurations. In the standard configuration it comes with a low-cost inertial motion unit, a depth sensor, a LBL system for navigation, GPS, GSM and Wifi.

AUV Isurus

UAV Lusitânia

ROV-IES

ASV Swordfish


UAV AntexM
Portuguese Air Force Academy

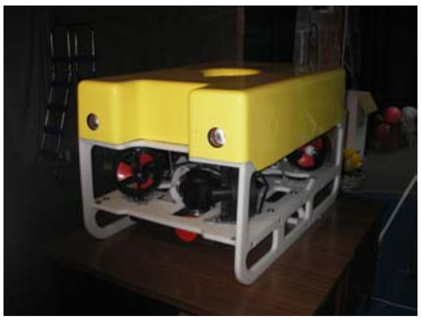

ROV-KOS

Figure 5 Vehicles

All of these vehicles use the same onboard computer system based on a PC-104 stack running a version of the real-time operating system Linux. We have a modular implementation of the on-board software architecture. This is done with the help of the DUNE framework and specified in the DFO formalism. This is integrated in the Seaware middleware engine, that encapsulates diverse forms of communication using a publish-subscribe framework. This hardware/software setup can been deployed for AUVs and control consoles using different operating systems (Linux, QNX, Windows) or programming languages (C++, Java).

4.2 Experiments

Our vehicles and toolset have been tested intensively in multiple operational deployments. We briefly describe some of these deployments.

In 2005 we used Neptus for mission planning and control of the IES ROV in the inspection of an underwater pipeline. The use of the same map for mission planning and execution greatly reduced the number of human errors. We were able to visualize the mission in simulation and to use the experience acquired in simulation to operate the vehicle in real time. The 3D visualization of the real motions in a virtual world proved quite useful for operations in waters with poor visibility. We tested the mission planning GUI and the generation of mission files through XSLT in operations with the Isurus AUV. This represented a great advance since we used to edit Isurus native mission files by hand. The number of planning errors was greatly reduced with the help of the 2D/3D maps and of the visual aids of our planning GUI. We have also built a new console to track the motions of Isurus with the help of data provided by the acoustic localization system. This console enabled us to evaluate mission performance in real-time. We had to provide consistency checks for displaying data coming from different sources.

We used Neptus to operate two Wireless Sensor Networks and two vehicles (Isurus and Roaz) in the NATO Swordfish exercise which took place in May 2006 in Tróia (Portugal). This was done in cooperation with the Portuguese Navy. There was one operator per vehicle and multiple consoles to subscribe to the data published by the vehicles and the sensors. Data was published live to the Internet.

In October 06, we conducted at rendezvous experiment between the Isurus and the Aries AUV at the Monterey Bay/CA, USA [13]. This was done in cooperation with the Naval Postgraduate School (NPS). ARIES is an AUV developed at Naval Postgraduate School [14] equipped with several navigation sensors including a ring laser gyro-based inertial measurement unit, Doppler velocity log, magnetic compass, and GPS receiver for periodic navigation corrections on the surface. Designed as a network server vehicle, ARIES carries radio and acoustic modems for inter-vehicle communications and can function as a node on a wireless network. The single AUV control scenario involved the Isurus AUV and the core control Isurus commands. Using the control and communication framework it was possible to perform to control mission execution and obtain real-time monitoring data for the vehicle state through a control console. The multiple AUV control and communication scenario involved both Isurus and ARIES. The rendezvous method considers a networked environment with one server vehicle and one or more survey vehicles, in which the server vehicle has a priori knowledge of the survey vehicles' mission profiles and may at any time be requested by one of the survey vehicles to rendez-vous for data exchange. The server vehicle then performs the necessary path-planning to achieve the requested time-optimal or energy-optimal rendezvous. In our rendezvous experiment we used Aries as server vehicle and Isurus as a survey vehicle and an acoustic message protocol. The experiment involves several types of mixed-initiative interactions: i) coordinated planning for multiple vehicles; ii) mission supervision of multiple vehicles; iii) simultaneous planning and execution for different vehicles; iv) vehicle re-tasking; and iv) operator’s handoff.

The mixed initiative interactions were mediated through Neptus. These took place at the planning state, during operations, and at the post-mission analysis stage. These stages may overlap in time, for example when it becomes necessary to re-plan the mission for a given vehicle. This happens because there will be several concurrent operations taking place at a given time: side-scan surveys; sonar interpretation and classification; and detailed sonar imaging. This level of concurrency poses a considerable burden on operators, especially when a significant number of events, each one requiring specific reactions, occur during a short period of time. To address this issue, and to balance the load on human operators, Neptus supports some levels of interoperability, namely handoff of control of a given vehicle. Neptus, also records the human-system interactions for performance analysis, and provides automated evaluation according to a set of measures of effectiveness. Performance analysis is of paramount importance to evaluate the mixed initiative command and control environment, and to develop and improve operational procedures and concepts of operation.

5.0 CONCLUSIONS

The problem of mixed initiative control of unmanned air and ocean going vehicles is discussed, a formal mixed initiative control framework for networked vehicles is introduced and tools for operational deployments are presented. We discuss these in the context of the developments from the Underwater Systems and Technologies Laboratory from Porto University in Portugal where we attempt to couple the development of scientific frameworks with lessons learned from field tests with human operators.

ACKNOWLEDGMENTS

The paper reports the results of a team work. The authors thank all of the past and present members of the Underwater Systems and Technology Laboratory for their contributions and for stimulating discussions. This research has been partly supported by Agência de Inovação under project PISCIS, by the AsasF project from Porto University and by Fundação Luso-Americana para o Desenvolvimento.

REFERENCES

[1] Almeida, Pedro; Bencatel, Ricardo; Gonçalves, Gil M.; Sousa, João B.; Multi-UAV Integration for Coordinated Missions, Encontro Científico de Robótica, Guimarães, April 2006.

[2] Almeida, Pedro; Gonçalves, Gil M.; Sousa, João B.; Multi-UAV Platform for Integration in Mixed-Initiative Coordinated Missions, First IFAC Workshop on Multi-vehicle Systems (MVS'06), Salvador, October 2006-

[3] Dias, P.S.; Fraga, S.L.; Gomes, R.M.F.; Gonçalves, G.M.; Pereira, F.L.; Pinto, J.; Sousa, J.B.; Neptus - a framework to support multiple vehicle operation Oceans 2005 – Europe, Volume 2, 20-23 June 2005 Page(s):963 - 968 Vol. 2

[4] Dias, P. S., R. Gomes, J. Pinto, G. M. Gonçalves, J. B. Sousa and F. L. Pereira (2006); Mission planning and specification in the Neptus framework. Humanitarian Robotics, ICRA 2006 IEEE International Conference on Robotics and Automation.

[5] Girard; Anouck R.; Sousa, João; Hedrick, J. Karl; A Selection of Recent Advances in Networked Multi-Vehicle Systems; Proceedings of the Institution of Mechanical Engineers, Part I, 2004 (a)

[6] IEEE Standard for Application and Management of the Systems Engineering Process, IEEE Std 1220-2005, September 2005.

[7] Marques, Eduardo; Gonçalves, Gil; Sousa, João; Seaware: A Publish/Subscribe Middleware for Networked Vehicle Systems, 7th Conference on Manoeuvreing and Control of Marine Craft (MCMC’2006), Lisboa, Portugal, 2006.

[8] NSA Standardization Agreement (STANAG), Standard Interfaces of UAV Control Systems (UCS) for NATO UAV Interoperability, 2006.

[9] Sousa, J. B., T. Simsek and P. Varaiya (2004). Task planning and execution for UAV teams. Proceedings of the Decision and Control Conference, Bahamas.

[10] Vaglienti, B.; Hoag, R.; Niculescu M., Piccolo system user guide, July 2004, www.cloudcaptech.com

[11] Marques, Eduardo; DFO – Data Flow Objects for embedded software control specification, Internal Report, FEUP, 2007.

[12] NSA Standardization Agreement (STANAG), Standard Interfaces of UAV Control Systems (UCS) for NATO UAV Interoperability, 2006.

[13] Marques, E, Pinto, J., Kragelund, S., Dias, P., Madureira, L., Sousa, A., Correia, M., Ferreira, Gonçalves, R., Horner, D., Healey, A., Gonçalves, G. and Borges de Sousa, J., "AUV control and communication using underwater acoustic networks", accepted for publication in the Proceedings of the Ocean 07 Conference, Aberdeen, Scotland, 2007.

[14] Marco, D. B., and Healey, A. J. "Current Developments in Underwater Vehicle Control and Navigation: The NPS ARIES AUV", OCEANS 2000 MTS/IEEE Conference and Exhibition, pp 1011-1016, 2000.

[15] Borges de Sousa, J., Johansson, K. H., Silva, J. and Speranzon, A. "A verified hierarchical control architecture for coordinated multi-vehicle operations", International Journal of Adaptive Control and Signal Processing, accepted for publication.