May 1, 2007

White Paper
Software Architecture and Strategic Plans for Undersea Cooperative Cueing and Intervention

Michael R. Benjamin
NAVSEA-DIVNPT, Code 2501

Abstract

The Undersea Cooperative Cueing and Intervention (UC2I) Program objectives include a heavy emphasis on developing autonomy for both unmanned underwater vehicles (UUVs) and Unmanned Surface Vehicles (USVs). The mission objective is to utilize a collection of such assets, each with different sensors, range, endurance, speed, and communications capabilities to work together to rapidly search and map a minefield area on the order of 100 x 100 kilometers in five days or under. This program explores the gain achieved by utilizing assets in a coordinated manner. Working together includes not only inter-vehicle cueing and sharing of information, but also the transportation, launch and recovery of UUVs from USVs. The possible configurations of platform assets, sensors, sensing algorithms, navigation algorithms and autonomy algorithms is enormous. Of significant concern to the Navy and the execution of this program is the ability to explore this configuration space to find the most effective system for achieving mission objectives. Practically speaking, this not only requires that software solutions are decoupled from hardware solutions, but that the collection of software modules that comprise the sensing, navigation and vehicle autonomy algorithms are also decoupled from one another. This white paper outlines the plan and software architecture ground rules for the UC2I program to meet that objective.

1. Overview

The architecture plan described here is composed of three paradigms. The first concerns separation of the vehicle autonomy system from the physical platform and platform specific core software. The second concerns the separation within the autonomy system between high-level functions, one of which is the module generating autonomy decisions. The third is the separation within the decision-making module between dedicated modules associated with mission objectives.

1. *Backseat-driver paradigm*: A division between "low-level" control and "high-level" control on the vehicle, with most likely the former residing on the vehicle's main computer, and the latter residing on a computer in a payload section that can be physically swapped out of the vehicle. The low-level control is also referred to as "vehicle control" and the high-level control as "mission control".

2. *Publish-Subscribe Middleware*: The term "middleware" here refers to the architecture software that coordinates the set of software modules collectively comprising the "backseat-driver" system running in the payload. Publish-subscribe middleware implements a community of modules communicating through a shared database process that accepts information voluntarily published by any other connected process, and distributes particular information to any such process that subscribes for updates to such information.

3. *Modular Behavior-Based Autonomy*: Behavior-based control is composed of a set of distinct, separate autonomy modules each vying for influence of the vehicle based on sensed conditions of the environment and individual objectives.

In the UC2I program, participation is sought in accordance with (1) and (2), *while leaving (3) entirely optional*. Discussion of each follows.

2. The Backseat Driver Paradigm

The backseat driver paradigm is not completely new; it has been used on several platforms typically as a special request from a vehicle customer seeking to control the vehicle with something other than the autonomy architecture and software produced by the vehicle manufacturer. The "backseat" is alternatively known as the "mission control" component, and the front seat as the "vehicle control" component. This is usually accomplished by having a separate computer in the payload section (Figure 1) that produces autonomy commands to the main vehicle computer (MVC) in the form of *heading*, *speed* and *depth* commands or *waypoint*, *speed* and *depth* commands. The "front-seat driver" converts the autonomy commands into control commands, i.e., *rudder*, *thrust*, *elevator*, typically with a set of PID controllers tuned to control that particular platform. The autonomy commands may be produced in the range of 1-10Hz, while the control commands may be updated in the 5-50Hz range. (As Figure 1 indicates, the "backseat" driver in the payload may not actually be back of the "front-seat" driver[1].)

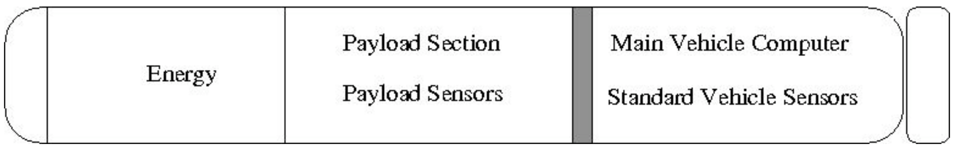


Figure 1: Simplified view of unmanned underwater vehicle components. The backseat driver notion refers to the separation of roles between high-level or mission control in the backseat and low-level or vehicle control in the front-seat. There may be sensors on either side that also may need to push information through this interface.

Flow of information: The flow of information between the two sections extends well beyond the passing of autonomy commands to the main vehicle computer. Typically the main vehicle computer has direct connection to platform native sensors such as a global positioning system (GPS), inertial navigation system (INS), Doppler velocity log (DVL), conductivity-temperature-depth (CTD), synthetic aperture sonar (SAS) and so on. The autonomy modules running in the payload section may need some or all of this information. The MVC may also be running processes that can supply fused or processed sensor to the payload section. One arrangement is to have the vehicle attitude information supplied to the payload from the MVC without specifying what sensors or fusing algorithms were used by the MVC. Furthermore, one or more of these sensors or others may be connected to the payload section instead and the information flows in the opposite direction to the MVC. External communication links to the vehicle in the form of radio frequency (RF), or acoustic communications (ACOMMS) typically have their entry point into the main vehicle computer (but not necessarily). This creates the further need to have information arriving from other platforms or from field-command propagated to the autonomy module in the payload section.

Although the MVC is acting as a virtual slave in the backseat mode to the autonomy commands produced by the backseat driver, a typical arrangement is to in fact have the MVC in ultimate control while temporarily "letting" the backseat driver have autonomy influence. The MVC may have certain veto conditions such as time-out, energy minimums, depth-limits or boundaries in the x-y plane. It may also reclaim control based on external communications. Some or all such conditions could be monitored and acted upon appropriately by the autonomy module, but it may be more suitable from the vehicle manufacturer's perspective to have such watchdog functionality performed in the MVC and treat such watchdog activity running in the backseat act as second layer of protection. Of course one such failure

[1] Although "backseat" is a word in the English dictionary, "frontseat" is not.

mode detectable only by the MVC is the outright failure of the payload section and subsequent cessation autonomy commands.

Current Issues: This paradigm has been implemented on a number of platforms (e.g., Bluefin, Hydroid, and Ocean Server) enough to show that it is a workable arrangement that provides significant benefits to autonomy users/developers. It has been tested less on surface craft, partly due to the relative lack of commercial USVs compared to UUVs. An objective of the UC2I program is to reach a further consensus on the backseat driver interface across UUV and USV platforms and manufacturers and perhaps get to the point where this mode of operation is not a special-case, special needs arrangement worked out between a platform customer and manufacture, but rather a standard choice for mode of operation, and relatively similar between manufacturers.

Benefits to the Navy: The primary benefit of operating with this separation between autonomy control and platform control is the freedom to separate autonomy development investments from platform investments. This is not to say that a platform developer may not in fact also be a suitable choice for providing autonomy solutions.

3. The Publish and Subscribe Middleware

The term "middleware" here refers to the architecture software that coordinates the set of software modules collectively comprising the "backseat-driver" system running in the payload. Publish-subscribe middleware implements a community of modules communicating through a shared database process that accepts information voluntarily published by any other connected process, and distributes particular information to any such process that subscribes for updates to such information.

The primary benefit of the publish-subscribe architecture is the separation of implementation details between individual modules. This also enables several modules to be developed (invested in) to provide the same functionality. A drawback sometimes associated with this architecture is the potential bandwidth bottleneck by requiring a single process to be involved in all information transactions. In practice this has not been a concern, especially with modern COTS computer systems. The benefits of modularity and simplicity often far outweigh such concerns.

The MOOS publish and subscribe middleware: MOOS[2] is a particular set of middleware implemented in C++ for both the Linux and Windows environment that implements a "publish and subscribe" architecture. The core functionality enables individual modules to communicate with one another and execute their primary functions at a pre-specified frequency (Figure 2).

[2] MOOS stands for Mission Oriented Operating Suite and was developed by Paul Newman as a post-doc at the MIT Department of Ocean Engineering before joining the faculty at Oxford University, where development has continued, partly with ONR support.

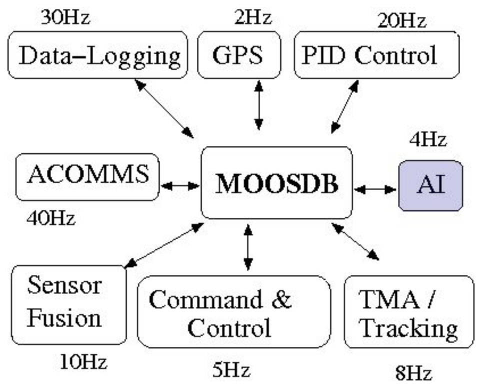


Figure 2: A MOOS community consists of a number of (typically Linux) processes running separately, perhaps on different machines, communicating via TCP/IP through a single database process (MOOSDB) in a publish-subscribe framework. MOOS also provides basic functionality for requesting the frequency in which a process executes its primary routine.

Components of a MOOS "community" include data logging, sensor processing, physical device interfaces, communication, as well as vehicle autonomy. The simple publish and subscribe interface allows one to swap in newer or competing modules without affecting other parts of the system. MOOS ([NEW03]) is an Open Source project initiated at MIT under ONR and has an extensive recent history of reliability on marine vehicles and indoor robots ([BBESB07], [EBWS07], [HN07], [PSN07], [CN07], [EBSL06], [BLCN06], [CN06], [NCH06], [BGN06], [WEL05], [EWL05], [HN05], [DPNH05], [NH05], [CHN05], [MDDPC05], [DDHMP05]).

Current Status: MOOS is an active Open Source project, partly funded by ONR, with source code fully available on the web from the Oxford University Robotics Research Group. The website is:

http://www.robots.ox.ac.uk/~pnewman/TheMOOS/

The software collection available at this site contains a critical subcomponent known as *MOOS Core,* which is the minimal code responsible for implementing inter-process communication and scheduling. It also contains a number of utilities that are essential in practice, and vehicle navigation and autonomy code used on prior robotic systems. The *MOOS Core* modules are the minimal common denominator modules – required for compatibility between MOOS users at different sites, platforms and projects. The rest is optional and open for newer, better or different versions to emerge.

Benefits to the Navy: The primary benefit derived from the publish and subscribe architecture is the ease and freedom to separately develop and invest in subcomponents of the overall software system responsible for vehicle autonomy. Furthermore MOOS is an Open Source project funded in part by ONR. The most critical piece of glue software is exceptionally small, well tested on marine vehicles, and open with no proprietary issues to hinder development of further Navy autonomy system components.

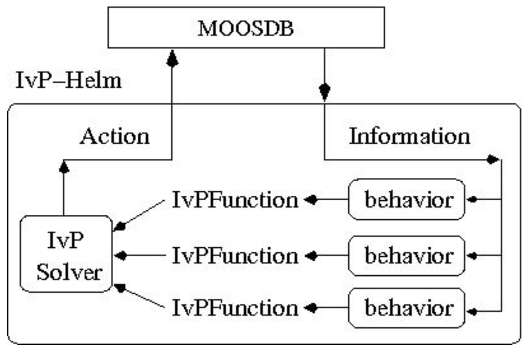
4. Modular Behavior-Based Autonomy

The MOOS community of software modules described in the previous section contains a subset of modules dedicated to autonomy that ultimately publishes *heading*, *speed* and *depth* autonomy command variables to the MOOSDB. These variables are subscribed to by another MOOS process that implements the communication to the font-seat driver as explained earlier. We describe here one such autonomy architecture, a behavior-based architecture, for implementing this autonomy functionality. It differs from the previous two discussions on MOOS and the back/front-seat driver because the autonomy architectures that may ultimately be invested in under the UC2I program need not be behavior-based or any other particular architecture. They must however have the same publish and subscribe interface of publishing the *heading*, *speed* and *depth* variables (at least 4Hz) and subscribing to whatever they need to make autonomy decisions.

The behavior-based discussion below does have one issue in common with the MOOS and back/front-seat driver discussions – the goal of modularity. The underlying objectives at those two levels is to define a minimal interface between distinct components to allow modular system development progression, yet also have that same interface be sufficiently rich to preclude limits on the capabilities of the individually developed components. In the back/front-seat driver case, this separation roughly is between the hardware and autonomy software on the vehicle. In the MOOS case, the separation is between software modules for sensing, navigation, autonomy, communication, data logging, simulation and so on. The discussion here is focused solely on the autonomy module(s) in the MOOS community. This functionality is also deeply challenging and equally benefited by modular development perhaps developed by different groups of experts with different missions and needs but common areas of overlap. As with the other cases, achieving modularity comes from settling on a particular way of doing business with a well-defined interface between modules that balances simplicity and extendibility.

Behavior-based control: Behavior-based control ([BRO86]) is composed of distinct software modules each potentially influencing control of the platform. The modules are independent and largely treated as black boxes when coordinating their output. Behavior coordination is the most contentious and difficult aspect of this architecture. Early approaches used a "layered" approach where behaviors had a strict precedence ordering and only a single behavior had control at any one time. Other early approaches used a "vector summation", a.k.a. "potential fields", method for handling two simultaneously active behaviors ([KHA85], [ARK87]). This method simply takes the average of two output vectors, and has been used effectively on many systems. It also has well-known shortcomings when taking the said average is known to be unreliable. These shortcomings were described in detail in [ROS97], [PIR98], and [RIE99], all of which endorsed an alternative style of behavior fusion based on multi-objective optimization.

Multi-objective optimization: Early research on multi-objective optimization for behavior fusion ([ROS97], [PIR98], and [RIE99]) provided virtually no computationally efficient technique for implementation, beyond brute force search of the decision space. The interval programming (IvP) method ([BEN02], [BEN04]) is a mathematical programming model for representing and efficiently solving multi-objective optimization problems, and implemented at the inner core of a marine vehicle autonomy engine ([BBESB07], [EBWS07], [EBSL06], [BLCN06], [BGN06]). The basic idea shown in Figure 3 is that each behavior produces an objective function at every iteration of the autonomy control loop (roughly 1-10Hz). The functions are piecewise linearly constructed "IvP Functions" with suitable syntactic structure balancing expressive power and computational efficiency during the combined optimization phase.



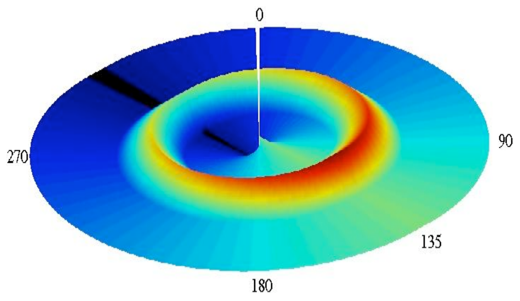


Figure 3: The IvP Helm is a MOOS module that implements a behavior-based architecture using multi-objective optimization for behavior coordination. On the left is the basic structure and flow of control and information during a single control loop iteration. On the right an objective function produce by a simple waypoint-following behavior is shown. The plot is shown in polar

coordinates for rendering purposes; with higher (red) peaks have a greater utility. In this case the waypoint is at a bearing of 135 degrees relative to the controlled vehicle.

Current Status: The IvP model and solution algorithms have been patented by the US Navy, and the full software implementation has been licensed to the general public under a pseudo Open Source license. By "pseudo" we mean that there are no restrictions for academic and non-profit entities, as well as for entities using this concept in the performance of related work for the US Government. Commercial licenses are available for discussion with NAVSEA-DIVNPT. A true Open Source license wouldn't discern between the above groups. The source code and documentation and reports of use in the field are at:

http://oceanai.mit.edu/mikerb/software

Benefit to the Navy: The primary benefit of the behavior-based architecture is the ease and freedom to separately develop and invest in subcomponents of the vehicle autonomy module. As with the middleware discussion, the important aspect is that overall system functionality is broken down into components with the "glue" being owned by the Government. The individual components may be derived from academia, a Navy lab, or be commercial proprietary products. But controlling the glue or interface allows the Government to compete and compare modules and invest in alternative approaches that share only module requirements.

5. Summary

Three paradigms were discussed here: the backseat/front-seat paradigm, the "publish and subscribe" middleware paradigm, and the behavior-based autonomy paradigm. It is worth repeating that, in the UC2I program, the first two will be considered standard for doing business while the third is optional. The motivation of all three is to ensure the Government has a sound system architecture strategy that allows for both competition between components, also allows for rapid field testing of complete systems with different hardware and software configurations to evaluate overall effectiveness, and also allows for further system growth not hampered by proprietary issues.

The reason for leaving the last paradigm as optional is that there is less consensus in the field of artificial intelligence (AI) as to what ultimately is the most effective architecture approach. It was discussed here for the following reason. If an autonomy system for unmanned marine vehicles is graded on (a) effectiveness in reliably achieving mission objectives, (b) the openness of the architecture to upgrades, verification and understanding, and (c) proprietary restrictions and cost for upgrades, then ultimately (a) will dominate the choice in system procurement. If however two alternative systems are nearly equivalent on (a), then (b) and (c) will quickly break the tie.

REFERENCES

[BBESB07] M. R. Benjamin, D. Battle, D. P. Eickstedt, H. Schmidt, A. Balasuriya, "*Autonomous Control of an Autonomous Underwater Vehicle Towing a Vector Sensor Array*", IEEE Conference on Robotics and Automation (ICRA), Rome Italy, April 2007.

[EBWS07] D. P. Eickstedt, M. R. Benjamin, D. Wang, H. Schmidt, "*Behavior-Based Adaptive Control for Autonomous Oceanographic Sampling*", IEEE Conference on Robotics and Automation (ICRA), Rome Italy, April 2007.

[HN07] K. Ho, P. Newman, "*Detecting Loop Closure with Scene Sequences*", Journal of Computer Vision, To Appear, 2007.

[PSN07] I. Posner, D. Schroeter, P. M. Newman, "*Describing Composite Urban Workspaces*", Proceedings of the International Conference on Robotics and Automation (ICRA), Rome Italy, April 2007.

[CN07] M. Cummins, P. Newman, "*Probabilistic Appearance Based Navigation and Loop Closing*", Proceedings of the International Conference on Robotics and Automation (ICRA), Rome Italy, April 2007.

[EBSL06] D. P. Eickstedt, M. R. Benjamin, H. Schmidt, J. J. Leonard, "*Adaptive Control of Heterogeneous Marine Sensor Platforms in an Autonomous Sensor Network*", IEEE/RJS International Conference on Intelligent Robots and Systems (IROS), Beijing China, October 2006.

[BLCN06] M. R. Benjamin, J. J. Leonard, J. A. Curcio, P. M. Newman, "*A Method for Protocol-Based Collision Avoidance between Autonomous Marine Vehicles*", Journal of Field Robotics, Vol. 23 No. 5, May 2006.

[CN06] M. Chandran, P. Newman, "*Motion Estimation from Map Quality with Millimeter Wave Radar*", IEEE/RJS International Conference on Intelligent Robots and Systems (IROS), Beijing China, October 2006.

[PSN06] I. Posner, D. Schroeter, P. M. Newman, "*Using Scene Similarity for Place Labelling*", Proceedings of the International Conference on Robotics and Automation (ICRA), Orlando Florida, April 2006.

[CN06] D. M. Cole, P. M. Newman, "*Using Laser Range Data for 3D SLAM in Outdoor Environments*", IEEE International Conference on Robotics and Automation (ICRA), Orlando Florida, May 2006.

[NCH06] P. M. Newman, D. M. Cole, K. L. Ho, "*Outdoor SLAM using Visual Appearance and Laser Ranging*", IEEE International Conference on Robotics and Automation (ICRA), Orlando Florida, May 2006.

[BGN06] M. R. Benjamin, M. Grund, P. M. Newman, "*Multi-objective Optimization of Sensor Quality with Efficient Marine Vehicle Task Execution*", IEEE International Conference on Robotics and Automation (ICRA), Orlando Florida, May 2006.

[WEL05] M. Walter, R. Eustice, J. J. Leonard, "*A Provably Consistent Method for Imposing Exact Sparsity in Feature-based SLAM Information Filters*", In Proceedings of the 12th International Symposium of Robotics Research (ISSR), San Francisco California, October 2005.

[EWL05] R. Eustice, M. Walter, J. J. Leonard, "*Sparse Extended Information Filters: Insights in Sparsification*", In Proceedings of the 2005 IEEE/RJS International Conference on Intelligent Robots and Systems (IROS), Edmonton, Alberta, Canada, August 2005.

[HN05] K. Ho, P. Newman, "*Multiple Map Intersection Detection using Visual Appearance*", 3rd International Conference on Computational Intelligence, Robotics, and Autonomous Systems, Singapore, December 2005.

[DPNH05] S. Dobnik, S. Pulman, P. Newman, A. Harrison, "*Teaching a Robot Spatial Expressions*", Second ACL-SIGSEM, Colchester, UK, April 2005.

[NH05], P. Newman, K. Ho, "*SLAM Loop Closing with Visually Salient Features*", International Conference on Robotics and Automation, April 2005.

[CHN05] D. Cole, A. Harrison, P. Newman, "*Using Naturally Salient Regions for SLAM with 3D Laser Data*", International Conference on Robotics and Automation, SLAM Workshop, Barcelona, April 2005.

[MDDPC05] J. Morash, R. Damus, S. Desset, V. Polidoro, C. Chryssostomidis, "*Adapting a Survey-Class AUV for High Resolution Seafloor Mapping*", 14th International Symposium on Unmanned Untethered Submersible Technology (USS), Durham NH, 2005.

[DDHMP05] S. Desset, R. Damus, F. Hover, J. Morash, V. Polidoro, "*Closer to Deep Underwater Science with ODYSSEY IV Class Hovering Autonomous Underwater Vehicle (HAUV)*", IEEE Oceans 2005 Europe, Brest France, 2005.

[BEN04] M. R. Benjamin, "*The Interval Programming Model for Multi-Objective Decision Making*", AI Lab Technical Memo AIM-2004-021, Massachusetts Institute of Technology, Cambridge MA, September 2004.

[NEW03] P. Newman, "*MOOS – A Mission Oriented Operating Suite*", Technical Report OE2003-07, MIT Department of Ocean Engineering, Cambridge MA, 2003.

[BEN02] M. R. Benjamin, "*Interval Programming: A Multi-Objective Optimization Model for Autonomous Vehicle Control*", PhD thesis, Brown University, Providence RI, May 2002.

[RWD02] J. K. Rosenblatt, S. B. Williams, H. Durrant-Whyte, "*Behavior-Based Control for Autonomous Underwater Exploration*", International Journal of Information Sciences, vol. 145, no. 1-2, pp. 69-87, 2002.

[BL00] A. A. Bennet, J. J. Leonard, "*A Behavior-Based Approach to Adaptive Feature Detection and Following with Autonomous Underwater Vehicles*", IEEE Journal of Oceanic Engineering, vol. 25, no. 2, pp. 213-226, April 2000.

[RIE99] J. Riekki, "*Reactive Task Execution of a Mobile Robot*", PhD dissertation, Oulu University, 1999.

[PIR98] P. Pirjanian, "*Multiple Objective Action Selection in Behavior Fusion*", PhD Dissertation, Aalborg University, 1998.

[ROS97] J. K. Rosenblatt, "*DAMN: A Distributed Architecture for Mobile Navigation*", PhD dissertation, Carnegie Mellon University, Pittsburgh, PA, 1997.

[ARK97] R. C. Arkin, "*Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior*", in Proceedings of the IEEE Conference on Robotics and Automation (ICRA), Raleigh, NC, pp. 264-271, 1987.

[BRO86] R. A. Brooks, "*A Robust Layered Control System for a Mobile Robot*", IEEE Journal of Robotics and Automation, Vol. RA-2, No. 1., April 1986.

[KHA85] O. Khatib, “*Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*”, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)”, pp. 500-505, St. Louis, MO, 1985.