INTERFACING THE CODA AND CADCON SIMULATORS: A MULTI-FIDELITY SIMULATION TESTBED FOR AUTONOMOUS OCEANOGRAPHIC SAMPLING NETWORKS

Erik Albert, Jonathan Bilodeau, and Roy M. Turner*
Department of Computer Science, University of Maine
5752 Neville Hall, Orono, ME 04469 USA
{albert17,bilodeau,rmt}@umcs.maine.edu

Abstract

Simulation testbeds are important for developing complex systems, such as autonomous oceanographic sampling networks (AOSNs), as they allow the development and testing of control mechanisms without incurring the cost or risk of using real vehicles. Over the past several years, two simulators for AOSNs, CoDA and CADCON, have been developed. The CoDA simulator is a multi-fidelity simulation testbed for the intelligent AOSN control mechanisms being developed at the University of Maine. While it can simulate AOSNs at a range of fidelities, it lacks truly high-fidelity simulation of the environment and of vehicle dynamics. CADCON is a high-fidelity multi-AUV simulation testbed developed at the Autonomous Undersea Systems Institute. This paper reports on work done to combine the two simulators into the CoDA/CADCON simulator, which is capable of a range of AOSN simulations, from faster-than-real-time simulations of aggregate AOSN control mechanism properties to high-fidelity, real-time simulations of an AOSN as it carries out missions.

Introduction

Work on developing complex systems is facilitated by the availability of simulators. This is especially true if the system is expensive or contains many expensive or difficult to obtain components. Such is the case for autonomous oceanographic sampling networks (AOSNs) [4], which are composed of many autonomous underwater vehicles (AUVs) and other instrument platforms.

*This work was funded in part by grants N0001-14-96-1-5009, N0001-14-98-1-0648, and N00014-00-1-0614 from the Office of Naval Research. The content of this paper does not necessarily reflect the position or the policy of the U.S. government, and no official endorsement should be inferred. Authors are listed alphabetically; contact author is Roy M. Turner (rmt@umcs.maine.edu). The authors thank Andrew Sylvia for his work on part of the simulator and also the other members of the Maine Software Agents and Artificial Intelligence Laboratory (MaineSAIL.umcs.maine.edu).

There are several existing simulators for AOSNs (e.g., [10; 16; 3]). These typically either simulate the high-level behavior of an AOSN (e.g., SAMON [10]) or the low-level behavior of the vehicles within the AOSN (e.g., CADCON [3]).

The CoDA (Cooperative Distributed AOSN controller) project [17] takes a different approach. The CoDA simulator, described in more detail elsewhere [16], is meant to allow AOSNs to be simulated at several different levels: it is a *multi-fidelity* simulator. For example, once the designers have completed an overall design for an AOSN control mechanism, they can use CoDA to quickly implement a high-level model and run simulation experiments. As experience with the design accrues, the model can be made incrementally more detailed, so that higher-fidelity simulations can be run. Different pieces of the model can be simultaneously at different levels of detail. For example, the high-level behavior of a group of AOSN agents (i.e., the vehicles and other instrument platforms) can be simulated by rules that model the overall effects of cooperation protocols, while other parts of the model (e.g., task assignment) can be modeled at a higher fidelity, perhaps even with the code that would be fielded in an actual AOSN. Multi-fidelity simulation promotes rapid prototyping and faster-than-real-time simulation (e.g., by discrete-event simulation) of overall system properties by selectively avoiding high-fidelity, continuous time-dependent pieces of the simulator, yet it also allows more detailed simulations to be done later for accuracy.

Until the current phase of the CoDA project, while the simulator has been able to simulate AOSNs at a range of fidelity levels, it has not been able to support truly high-fidelity simulations of vehicle dynamics and sensors. This paper reports on the current phase, in which work that has been done add this capability to CoDA by interfacing it with the Autonomous Undersea Systems Institute's (AUSI's) CADCON high-fidelity simulator. The combined simulator, called CoDA/CADCON,

This paper appears in the *Proceedings of the 13th International Symposium on Unmanned Untethered Submersible Technology (UUST)*, Durham, NH, August, 2003.

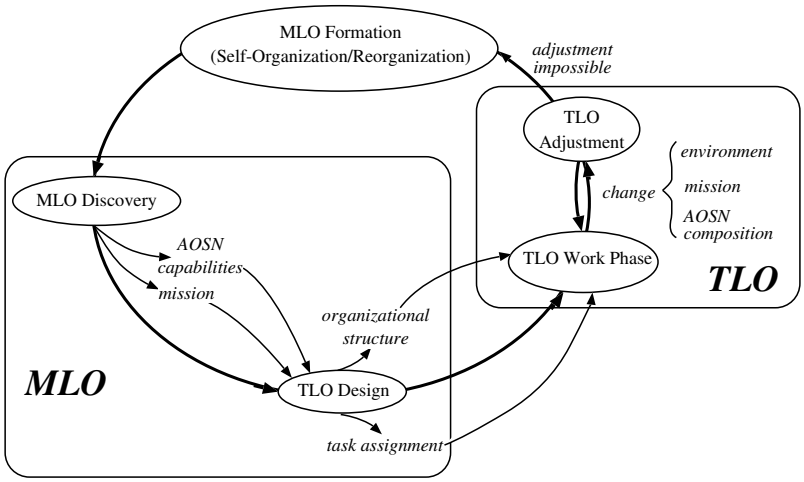

Figure 1: CoDA's two-level approach to AOSN control. [From Turner & Turner, 2001, copyright © 2002 IEEE; used by permission.]

is now able to perform at fidelity levels ranging from high-level, low-fidelity simulation of only the AOSN's aggregate properties to high-fidelity simulations which include agents' reasoning processes as well as vehicle dynamics.

In the remainder of this paper, we first discuss the two major pieces comprising CoDA/CADCON, CoDA and CADCON. We then describe the CoDA/CADCON simulator, and discuss some of the problems we faced in interfacing the two and our solutions. We then give an example of a CoDA/CADCON simulation.

CoDA

The CoDA project has as its goal developing an open, distributed, intelligent control mechanism for AOSNs. We assume that for many AOSN missions of interest, it may be impossible to pre-specify an organization for the system. For example, it may be impossible to predict ahead of time which agents will take part in the AOSN (e.g., due to attrition during transit through a hostile or dangerous area). We further assume that many missions will require the absence of contact with the AOSN, either due to mission characteristics (e.g., covert missions) or the environment (e.g., when operating under ice). Consequently, the AOSN must be able to organize itself autonomously, carry out its mission, and reorganize as needed.

Our approach treats the AOSN as a multiagent system that uses cooperative distributed problem solving (CDPS) (e.g., [5]) to organize and reorganize. No single agent is assumed to have a complete global view or complete control. Instead, the agents cooperate to organize the AOSN, carry out its mission, and reorganize it as needed.

CoDA uses a two-level approach. When the AOSN components arrive at the work site, some subset of them communicate and cooperate to form a *meta-level organization* (MLO). The MLO is a loose organization composed of those agents intelligent enough to participate in organization design. Its job is to analyze the current situation and design another organization, the *task-level organization* (TLO), that will actually conduct the mission. Once designed, the TLO takes over control of the AOSN until there is a significant problem (e.g., loss of a vehicle), at which point an MLO re-forms and repairs or redesigns the TLO. Figure 1 illustrates this process.

Cooperation protocols [17] control the interaction of the AOSN's agents. Different protocols are used in different phases of CoDA's operation and by different classes of agents. Task assignment is done based on matching an agent's capabilities with those needed for the mission. This is currently done via a constraint-based mechanism [11; 12] that is based on constrained heuristic search [6]; in the future, this will be done in a distributed manner. Organization design is currently done very simply, and the TLO is always a simple hierarchy. In the future, other organizational structures will be used as well.

The CoDA simulator is written in Allegro Common Lisp (Franz, Inc.) and CLIPS, a rule-based language developed by NASA [7], and runs on Linux and other Unix-like systems. Low-fidelity, high-level aspects of the AOSN's behavior are encoded

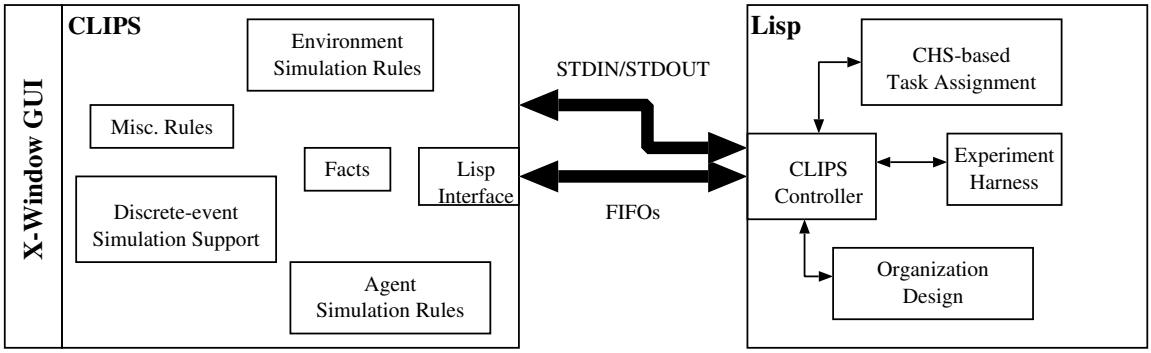

Figure 2: The CoDA simulator.

in CLIPS rules, while higher-fidelity portions of the control mechanism are written in Lisp. The simulator can be under the control of either CLIPS or Lisp, the former to use CLIPS' graphical user interface for debugging and the latter to run simulation experiments.

Figure 2 shows CoDA's structure prior to interfacing with CADCON. The CLIPS portion contains rules for a low-fidelity simulation of the AOSN's environment and agent movement as well as rules that simulate the agents' behavior when following the CoDA protocols. It also contains a discrete-event simulation (DES) component [1] that manages the simulation time. For example, when one agent sends a message via acoustic link, the time-of-flight is determined and an event is posted to cause delivery of the message at the appropriate time. If no other events are scheduled before that one, and there are no other high-priority rules triggered, the DES component will advance the system's simulated time to the time of that event.

The Lisp portion of the simulator contains a module to control CLIPS as well as modules for high-fidelity simulation of parts of CoDA's operation. Included in these are modules for constrained heuristic search-based task assignment and TLO design. This portion of CoDA also contains an "experiment harness" that generates various AOSN configurations for experiments.

Communication between the two parts of CoDA is by means of standard input/output and Unix named pipes (FIFOs). The pipes handle bidirectional I/O between the two languages during normal operation. For example, requests to generate randomly-configured AOSNs travel over one FIFO from CLIPS to Lisp, and the result travels back via the other FIFO. Lisp uses standard input/output (stdin/stdout) to control CLIPS. Basically, when Lisp is in control of the simulation, it interacts with CLIPS essentially as would a user typing at the keyboard. In this way, commands, rather than simulation messages, can be given to CLIPS and the results received by Lisp.

CADCON

CADCON is a high-fidelity, multi-vehicle simulator written in C and OpenGL that runs under the Linux and MS Windows operating systems [3]. It realistically simulates several different kinds of AUV, as well as several different kinds of sensor data. It is a distributed simulation testbed; users can connect from around the world to control the simulated AUVs. It also allows hardware-in-the-loop simulation. Clients are available via the World Wide Web[1].

Figure 3 shows the structure of CADCON. It is a client–server system. An environment server provides a simulation of the environment, including sensor information and communication (e.g., acoustic link) simulation. A visualization client provides a graphical user interface to the distributed system. One or more AUVSim clients are used to simulate AUVs, which interact with the environment server to comprise the simulated system. Additional clients are planned, as the figure shows. All agents communicate with one another via the Internet.

The Combined Simulator

In this section, we discuss the overall design of the combined simulator. In the next section, we discuss the integration issues that arose and our solutions.

The overall design of the combined CoDA/CADCON simulator is shown in Figure 4. The interface between CADCON and CoDA is via the Internet.

[1] At `www.ausi.org`.

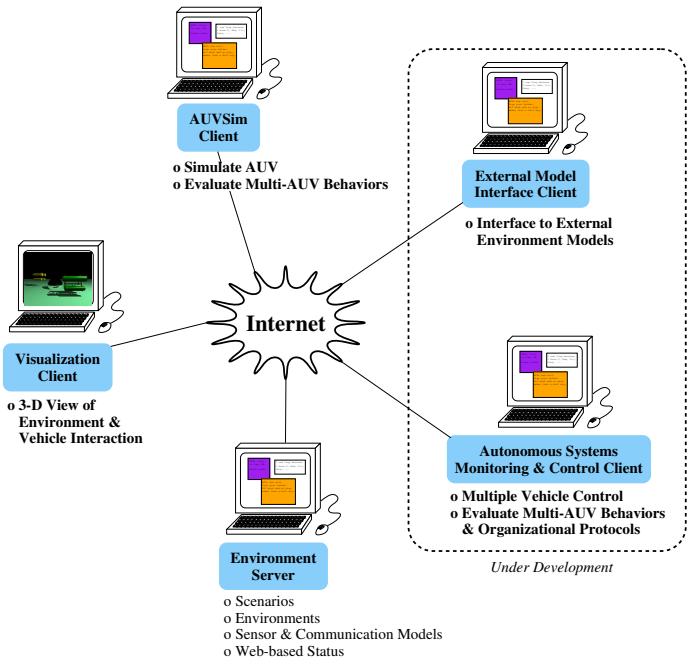

Figure 3: The CADCON Simulator. (Courtesy AUSI.)

CADCON was designed to support multi-agent simulations, but the assumption was that each piece of vehicle control software would control a single AUVSim. Vehicle control software would communicate with its AUVSim "body" via the generic behavior-based Common Command Language, CCL [15; 3]. In contrast, CoDA needs to control several or many simulated AUVs simultaneously.

One approach would have been to have CoDA control several instances of AUVSim clients via the network and CCL. However, this has the drawback of requiring CoDA to translate from its internal representation to CCL for each command sent to each AUVSim. In addition, CCL itself is still under active development, hence subject to change.

Consequently, we took the approach of implementing AUVSim functionality in Lisp. In CoDA/CADCON, the Common Lisp Object System (CLOS) class `VIPbase`[2] implements a Lisp-based AUVSim. This has several benefits:

- it avoids the overhead of translation to and from CCL;
- it decouples work on CoDA/CADCON from work on AUSI's AUVSim clients;
- it prevents a proliferation of AUVSim processes, hence making debugging and control of the simulator more straightforward; and
- it provides a Lisp-based, simple AUVSim for those investigators working in the Lisp language, for example, those looking at artificial intelligence techniques for mission-level control.

We anticipate that over time, the Lisp-based AUVSim client will become the home for more and more of the intelligent vehicle control software, as, for example, mission controllers such as Orca [13; 14] replace the simulated mission controllers currently present in CLIPS rules. We also anticipate that as CADCON and CoDA mature, `VIPbase` will become less of an AUVSim client and more of an interface to AUSI's AUVSim clients. This is part of the planned progression in CoDA from high-level simulation to fielded AOSNs.

In the remainder of this paper, we will refer to instances of `VIPbase` as AUVSim agents.

Specifications for AUVSim agents are written in a frame-based knowledge representation language. This not only provides a convenient way to describe the vehicles, but it also provides compatibility with other, frame-based systems (e.g., Orca) that will use the agents in the future. So far, specifications exist for (and there can be AUVSim agents for) EAVE vehicles [8], solar-powered AUVs [9], and some instrument moorings, in particular, a simplified version of a CONVEX mooring used to detect convective

[2]The term *VIPs* has been used in the past in our work to refer to Vehicles and Instrument Platforms.

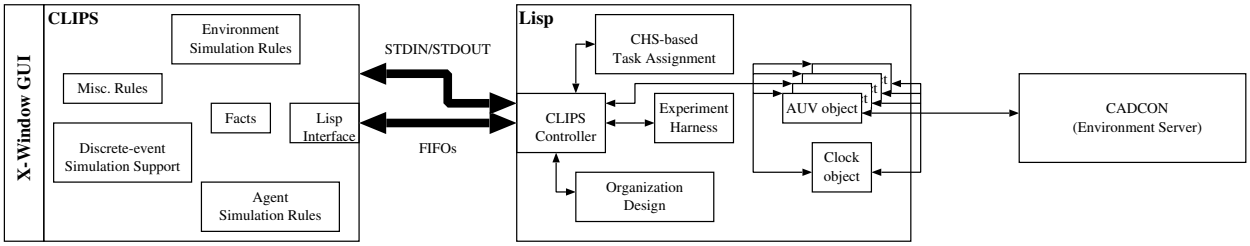

Figure 4: The CoDA/CADCON simulator.

overturn events [2]. For example, Figure 5 shows the definition of an EAVE vehicle.

To set up a simulation, CoDA either generates a sample AOSN or creates one based on a file the user specifies. The AOSN's agents, represented by Lisp AUVSim agents, are also reflected in corresponding facts in CLIPS' working memory. When each AUVSim agent is created, it registers as a vehicle with CADCON's environment server. When the simulation begins, CLIPS rules simulate the AOSN as it follows CoDA protocols to organize itself and carry out its missions. Movement and other commands are sent to the corresponding AUVSim, which communicates with the environment server. As the server sends back updated position, posture, sensor, and other data, AUVSim translates this as necessary to match what CoDA expects, then informs CLIPS, which asserts it as facts in its working memory.

Integration Issues and Solutions

The CoDA and CADCON simulators are quite different from one another, both in purpose and implementation. Consequently, integrating the two was challenging. The major issues that arose are the following:

Implementation languages. The simplest issue to deal with, thanks to CADCON's networked design, was the difference in implementation languages. Initially, the CLOS agent class used Allegro Common Lisp's (ACL's) foreign-function interface to call some AUVSim functions provided by AUSI, written in C. This worked; however, it did make the CoDA/CADCON simulator dependent on AUSI's code rather than just on the network protocols and message types. Consequently, AUVSim code was replaced by Lisp code, and the current version communicates directly with CADCON's environment server. As mentioned, in the future communication may be with AUVSim clients running separately from CoDA.

Keeping CADCON integration optional. A goal of this phase of the work was to add capability to what already existed in the CoDA simulator, not replace one capability with another. Thus, we want the user to be able to run simulations using CoDA with or without CADCON, that is, to continue to allow multiple levels of fidelity of simulation. Consequently, all of the changes to CoDA to create the CoDA/CADCON simulator needed to be optional. This means that the simulator:

- must be able either to simulate vehicle behavior and capabilities using CLIPS rules or via the AUVSim agent communicating with CoDA;
- must be able to use either discrete or continuous time (see below); and
- must be able to model, at some level of detail, the environment when it is not using CADCON's environment server.

By doing this, simulations can be run in low-fidelity mode much faster than real time to quickly test AOSN control mechanisms, then, when desired, run in high-fidelity mode in real-time to get better simulation results.

Continuous versus discrete time. CADCON uses simulated continuous time. Although it has a basic cycle rate, the passage of simulated time is roughly one-to-one with the passage of real time. This is not the case with CoDA. CoDA uses discrete time, maintained by a discrete-event simulator. Scheduled events are placed in a priority queue. When all CLIPS rules have fired that are applicable at the current time, the next event is taken from the queue, and the simulated time is advanced to that event's time. Consequently, a simulation can progress very rapidly if events are sparsely distributed over time. This mechanism for handling time is only possible because CoDA does not try to model such things as vehicle motion to any level of detail: when an AUV needs to move from (x_1, y_1, z_1) to (x_2, y_2, z_2), the simulator's rules determine the travel time and simply post an event representing the AUV's arrival at its destination.

```
(defframe eave (^coptor)
     (cadcon-implementation-class VIPBase)

     (masss 300)          ;; placeholder value
     (bouyancy 0.0)       ;; placeholder value
     (drag 0.25)
     (position (0.0 0.0))
     (min-turn-radius 0 (aspects (units meters)))
     (length 2 (aspects (units meters))) ;; approx.
     (width 1 (aspects (units meters)))  ;; approx.
     (height 1 (aspects (units meters))) ;; approx.
     (power-system -
        (aspects (isa ^battery-power-system (constraint t))))
     (propulsion-system -
        (aspects (isa ^eave-propulsion-system (constraint t))))
     (sensor-package -
        (aspects (isa ^eave-sensor-package) (constraint t))))
```

Figure 5: Frame definition of an EAVE AUV.

This mismatch in time representation was one of the thorniest issues in creating the CoDA/CADCON simulator, especially since we wanted (1) to keep the ability to post events (e.g., to simulate the loss of an AUV) and (2) to keep the DES ability so that we could still run CoDA apart from CADCON when needed.

The first solution we considered was to create a process that would run on the Lisp side of CoDA constantly send time information to CLIPS. We rejected this idea because CADCON has its own system time maintained by the environment server, and one typically wants each part of a simulation to agree on the state of the current simulation environment. Different time values could lead, for example, to CADCON and CoDA having different velocity/acceleration values for an AUV.

We solved this problem by using the system time from CADCON. The packets CoDA receives from the environment server include timestamps. The Lisp side of CoDA passes this information to CLIPS, where it is asserted as the new system time. The CLIPS rules that determine which events should take place simply check that the system time is greater than or equal to the time an event should take place. Because this mechanism for performing a scheduled event is independent of the mechanism for advancing time, simply making this alteration to how time advances causes each scheduled event to happen in real time (within the resolution of CADCON's time cycle). A problem in implementing this solution was that vehicle bodies are not created in CADCON at the start of the simulation, but rather as they enter the AOSN work area, via events defined ahead of time in the problem statement. Without vehicle bodies connected to the environment server, there are no packets (and therefore no timestamps) being received from CADCON. Our solution was to create a new vehicle body type called a *clock-vip*. Clock-vips are massless agents that the server treats like any other agent, yet have no detectable physical presence within the simulation. Their only function is to receive the time from the server and pass this information to CLIPS. One of CoDA's first actions is to create an instance of `clock-vip`.

Controlling multiple simulated AUVs simultaneously. A simulated AOSN obviously requires multiple simulated AUVs and instrument platforms. In our approach, all AUVSim agents are controlled by CoDA rules that move them to satisfy the needs of the protocols and mission. Vehicles can be added and deleted when necessary (e.g., to simulate vehicle failure or a new AUV wishing to enter the AOSN), and they can be paused and resumed when the user needs to pause/resume the simulation.

Handling telemetry and sensor data. CoDA needs information about the state of its vehicles as well as data from their sensors. In prior versions of the simulator, CLIPS rules simulated this information. In the CoDA/CADCON simulator, however, much of this information comes from CADCON's environment server via the network. When a packet arrives, the AUVSim agent unpacks the information and translates it into a form that CLIPS rules can use; this information is then sent via the FIFOs to be asserted in working memory.

```
(defrule raster-trigger
  ?r <- (raster ?vip ?x ?y ?z ?x1 ?y1 ?z1 ?resolution)
  =>
  (retract ?r)
  (assert (waypoint (vip ?vip) (x ?x) (y ?y) (z ?z) (phase 4)
                (dx (- ?x1 ?x)) (dy ?resolution) (end FALSE)
                (startx ?x) (starty ?y) (startz ?z)))
  (assert (waypoint (vip ?vip) (x ?x) (y ?y1) (z ?z) (end TRUE) (position left)))
  (assert (waypoint (vip ?vip) (x ?x1) (y ?y1) (z ?z) (end TRUE) (position right)))
  (move-vip ?vip ?x ?y ?z))
```

Figure 6: A CLIPS rule to move an AUV to the first waypoint in a raster search.

The environment server provides information about a vehicle's position and attitude. Some information is provided about the local environment as well, such as magnetic flux, current velocity, temperature, and so on. This information is the actual data from the simulated environment, not sensor data. The AUVSim agent, like the "real" AUVSim client, must simulate sensors that view this information and provide simulated data. For now, most of this information is simply passed through as data to CLIPS; in the future, the agent will be able to modify the data via various noise models, etc. Sensors whose information is not provided by the environment server are simulated completely by the agent, then their information is reported to CLIPS.

Issuing commands. For now, CLIPS rules simulate most of the intelligence in the AOSN. This includes what corresponds to the agent programs (e.g., intelligent mission controllers) that would, in a real system, control the vehicles.

Commands to move, to communicate, or (ultimately) to activate effectors are sent from CLIPS rules to the appropriate agent in Lisp via the FIFOs. For example, Figure 6 shows a rule that initiates a raster search of an area by setting a waypoint and telling the corresponding AUVSim to move to that waypoint. To move, the AUVSim agent computes the appropriate desired velocity vector and angular velocities, then sends this information to the environment server. For simulated communication (e.g., via an acoustic modem) or effector commands, messages are also sent to the environment server. The environment server uses the desired velocities to move the vehicle.[3]

Complex instrument platforms. Because CADCON's environment server does not directly support one VIP having multiple sensors at various positions, as is needed for (e.g.) a CONVEX mooring, we needed to create a system where AUVSim clients can create and control other AUVSim clients in order to support complex instrument platforms. A complex AUVSim that controls other AUVSims is specified in its frame representation as having "intelligent" components. When instantiated, each of its intelligent components is instantiated in turn and their output functions (normally connected to CLIPS) are redirected to send sensor and telemetry information to their controlling AUVSim. In addition, the methods that the main AUVSim client supports (i.e., connect, disconnect, transit, etc.) must call the corresponding method in each of its intelligent components so that CLIPS can treat the compound AUVSim as a single entity.

Network issues. CADCON communicates with its clients using packets which are essentially nested C structures. In Lisp, we needed to represent these structures twice: once as a Lisp structure whose slots are native Lisp types, and again as a foreign type which can be directly sent to and received from a binary stream. Each pair of structures has corresponding functions to convert between them and also handle other conversion issues, such as converting data to/from network byte order, replacing Lisp `nil` values with a default value specific to each data type, and truncating strings that are too long.

Speed mismatch. Not only do CoDA and CADCON operate with different representations of time, their speed differs. CADCON's environment server, for example, generates a position (etc.) packet every second. Given the complexity of what CoDA is doing, it cannot guarantee that it will respond within that time, although it usually will. Consequently, the AUVSim agent must manage any mismatch. This is done by allowing the agent to send information to CLIPS at a slower rate than it receives it from CADCON. Newer information that arrives at the agent before it is ready to send information to CLIPS replaces older information, so that CLIPS always gets the most current information.

[3]This is essentially how the real AUVSim clients currently work, as well. In the future, desired accelerations or even thruster output might be used instead.

```
(defproblem demo-problem
  (end (time (minutes 20)))             ;end simulation in 20 min.
  (tasks
   (background-survey                   ;surveying task
    (methods survey-alt1 survey-alt2))
   (convex                              ;CONVEX task
    (methods convex-default))
   (communication-relay                 ;radio mooring, e.g.
    (methods stationary-comm))
   (LBL1                                ;long-baseline nav transponder
    (methods LBL))
   ...)
  (methods
   (survey-alt1                         ;requires two surveys with two instruments
    (capabilities survey-magnetometer survey-side-scan-sonar))
   (stationary-comm                     ;comm. task needs a radio
    (capabilities radio))
    ...)
  (VIPs
   (EAVE-Ariel                          ;an EAVE vehicle
    (caps CDPS survey-magnetometer loiter transit search acoustic-link
          manage manage manage manage)  ;simplistic representation for mult. caps
    (resource-units 3) (location 0 0 0) (heading 0 0 0)
    (entry                              ;when the EAVE shows up -- 5 s, std. dev. 0.3 s
     (time (seconds 5 .3) (distribution normal)))
    (exit                               ;exits 100 seconds later
     (time (seconds 100 10) (distribution normal))
     (type sudden)))                    ;without notification
   (EAVE-Arista                         ;another EAVE
    ...))
  (events
   (failure                             ;AUV1 fails sometime during
    (vip AUV1)                          ; the TLO-work phase of simulation
    (type clean)
    (phase "TLO-work"))
    ...))
```

Figure 7: A portion of a problem definition.

Example

To use the CoDA/CADCON simulator, the user first defines the vehicles and instrument platforms by defining frames to represent them (see Figure 5), then either defines a problem or else lets the simulator build one or more problems.

To specify a problem, the user creates a file containing a problem definition. Figure 7 shows one such problem definition. Currently, the parts of a problem definition are:

- Task definitions: one or more tasks are named and alternative methods are specified for accomplishing each.
- Methods: each method specifies the vehicle/instrument platform capabilities it needs (e.g., CTD, raster search, etc.).
- VIPs: vehicles and instrument platforms are specified; for each, its capabilities, resources, and location is specified, as well as when it is to enter or leave the system.
- Events: this portion specifies events that are to be scheduled to occur during the simulation, for example, the failure of an AUV.
- Simulation properties: simulation end time, etc.

The current problem specification mechanism will be extended in the future to allow the specification of full AND–OR task trees [12], task locations, time, and resources, and constraints between task elements (e.g., that two capabilities are needed at the same time, on the same vehicle).

The user next makes an instance of the simulator, telling it the location of the problem file. When the simulator is run, the problem definition specifies which AUVSims are created and when they are to be created (i.e., as the simulated vehicles arrive at the work site). These then register with CADCON's environment server.

Figure 8 shows two screenshots from CADCON's visualization client when CoDA/CADCON is running. On the right is a simple problem containing a solar AUV, an EAVE, and a CONVEX mooring. On the left is a more complex simulation containing seven AUVSims: six solar AUVs and a CONVEX mooring. (A mooring with multiple sensors, such as CONVEX, currently shows up in the simulation as multiple AUVSim clients.)

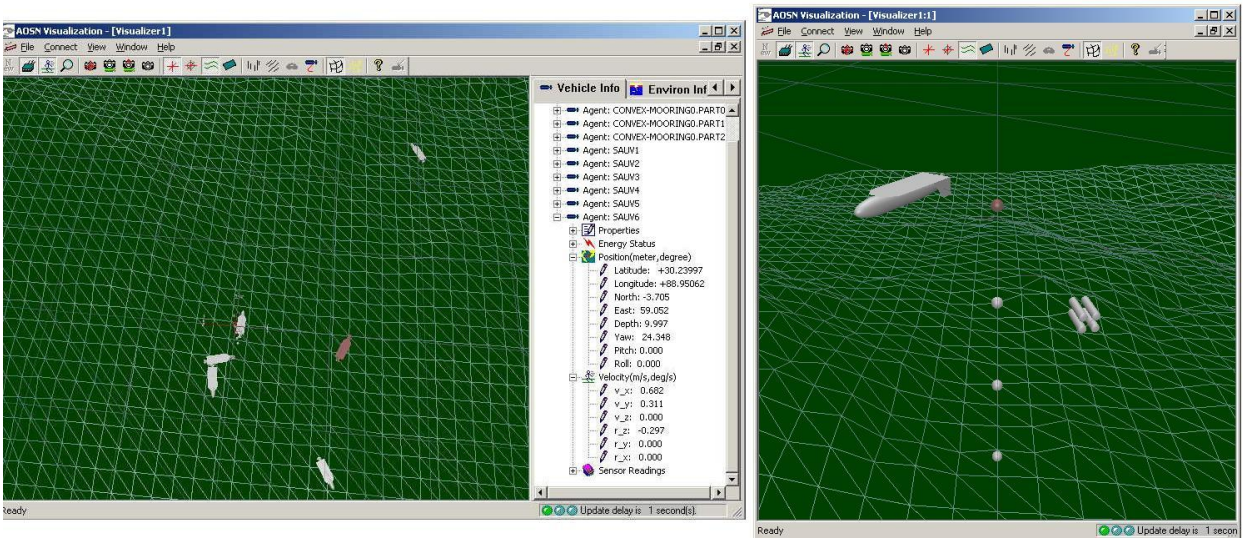

Figure 8: CoDA/CADCON screen snapshots.

Conclusion and Future Work

Multi-fidelity simulators facilitate the development of complex systems such as AOSNs. Such simulators allow simulations to be run at several different levels, depending on the needs of the current development efforts, from high-level, faster than real-time simulations of aggregate AOSN properties to highly-realistic simulations of vehicle movement and behavior.

The purpose of this phase of the CoDA project was to extend the CoDA multi-fidelity simulator to support highly-fidelity simulations of AUVs via interfacing with the CADCON simulator. Currently, CoDA/CADCON simulations can range from aggregate property simulations to those involving very realistic AUV behavior.

In addition, this phase of the project supports its sister project, Orca, and projects building on Orca. The mechanisms developed for utilizing CADCON from within Lisp will shortly also be used by the Orca intelligent mission controller, allowing Orca to drive simulated AUVs in CADCON simulations. This replaces an older, much cruder simulation testbed.

Future work in the CoDA project will focus on issues related both to CoDA protocols and to simulation. Most of the future work will focus on further developing the mechanisms and protocols for AOSN operation, in particular for autonomous organization/reorganization and task assignment. Work will also focus on improving the rather primitive means CoDA now uses to describe problems and, to a lesser extent, vehicles and instrument platforms. Currently, for example, tasks require particular capabilities, but there is no principled way to specify when and where those capabilities will be needed, nor is there a way to specify resource needs. Constraints between capabilities are also not well-supported; at the current time, for example, one cannot specify "capability A needs to be done by the same vehicle as capability B".

With respect to the CoDA/CADCON simulation, most of our future work will be on the CoDA side. We anticipate moving more of the CoDA control mechanism into Lisp and out of CLIPS as we move from aggregate property simulation to agent-based simulation. To do this, vehicle controllers (e.g., Orca) will be linked to our Lisp AUVSims (`VIPbase` objects). Also, as we mentioned above, as work on CADCON progresses, `VIPbase` will contain less vehicle simulation code; instead, `VIPbase` will be modified to interact with AUSI's AUVSim clients. This will allow higher-fidelity simulations as well as the option of using real hardware. This will be an important step along the path toward eventually moving CoDA out of simulation and into the real world.

References

[1] J. Banks, J. S. Carson, II, and B. L. Nelson. *Discrete-Event System Simulation*. Prentice Hall, second edition, 1996.

[2] F. Bub, W. Brown, P. Mupparapu, K. Jacobs, and B. Rogers. Hydrographics survey report: Convective overturn experiment (CONVEX): R/V *endeavor* cruise en–291. Technical

report, University of New Hampshire Ocean Process Analysis Laboratory, 1997. (ekman.sr.unh.edu/OPAL/CONVEX/EN291/en291_report.html).

[3] S. G. Chappell, R. J. Komerska, L. Peng, and Y. Lu. Cooperative AUV Development Concept (CADCON) – an environment for high-level multiple AUV simulation. In *Proceedings of the 11th International Symposium on Unmanned Untethered Submersible Technology (UUST99)*, Durham, NH, August 1999. The Autonomous Undersea Systems Institute, Lee, NH.

[4] T. Curtin, J. Bellingham, J. Catipovic, and D. Webb. Autonomous oceanographic sampling networks. *Oceanography*, 6(3), 1993.

[5] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Cooperative distributed problem solving. In A. Barr, P. R. Cohen, and E. A. Feigenbaum, editors, *The Handbook of Artificial Intelligence*, volume IV, pages 83–147. Addison–Wesley Publishing Company, Reading, MA, 1989.

[6] M. S. Fox, N. Sadeh, and C. Baykan. Constrained heuristic search. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 309–315, Detroit, MI, August 1989.

[7] J. C. Giarratano. *CLIPS User's Guide*. NASA, Information Systems Directorate, Software Technology Branch, Lyndon B. Johnson Space Center, Houston, TX, 1993.

[8] J. C. Jalbert. EAVE–EAST field test results. In *Proceedings of the 1984 Conference of the IEEE Oceanic Engineering Society (Oceans '84)*, Washington, DC, 1984.

[9] J. C. Jalbert, P. Irazoqui-Pastor, S. Miles, D. R. Blidberg, and D. James. Solar AUV technology evaluation and development project. In *Proceedings of the Tenth International Symposium on Unmanned Untethered Submersible Technology*, pages 75–87, Durham, NH, 1997.

[10] S. Phoha, J. Stover, R. Gibson, E. Peluso, and P. Stadter. Autonomous ocean sampling mobile network controller. In *Proceedings of the Tenth International Symposium on Unmanned Untethered Submersible Technology (UUST)*, pages 362–374, Durham, NH, September 1997.

[11] E. H. Turner. Task assignment in AOSNs: A constraint-based approach. In *Proceedings of the 10th International Symposium on Unmanned Untethered Submersible Technology (UUST)*, Durham, NH, 1997.

[12] E. H. Turner and R. M. Turner. A constraint-based approach to assigning system components to tasks. *International Journal of Applied Intelligence*, 10(2/3):155–172, 1999.

[13] R. M. Turner. Context-sensitive, adaptive reasoning for intelligent AUV control: Orca project update. In *Proceedings of the 9th International Symposium on Unmanned Untethered Submersible Technology*, Durham, NH, September 1995.

[14] R. M. Turner. Context-mediated behavior for intelligent agents. *International Journal of Human–Computer Studies*, 48(3):307–330, March 1998.

[15] R. M. Turner, D. R. Blidberg, S. G. Chappell, and J. C. Jalbert. Generic behaviors: An approach to modularity in intelligent systems control. In *Proceedings of the 8th International Symposium on Unmanned Untethered Submersible Technology (AUV'93)*, Durham, New Hampshire, 1993.

[16] R. M. Turner and E. H. Turner. Simulating an autonomous oceanographic sampling network: A multi-fidelity approach to simulating systems of systems. In *Proceedings of the Conference of the IEEE Oceanic Engineering Society (OCEANS'2000)*, Providence, RI, September 2000.

[17] R. M. Turner and E. H. Turner. A two-level, protocol-based approach to controlling autonomous oceanographic sampling networks. *IEEE Journal of Oceanic Engineering*, 26(4), October 2001.