Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

SIMULATION ENVIRONMENTS FOR THE DESIGN AND TEST OF AN INTELLIGENT CONTROLLER FOR AUTONOMOUS UNDERWATER VEHICLES

Michael W. Roeckel
Robert H. Rivoir
Ronald E. Gibson
Stephen P. Linder

Applied Research Laboratory
Pennsylvania State University
P.O. Box 30
State College, PA 16804 U.S.A.

ABSTRACT

Intelligent controllers usually consist of a hybrid system that includes both discrete and continuous processes. This hybrid construction poses difficulties in validating and verifying their design. As the use of intelligent controllers proliferates throughout society, the development of simulation techniques that support both the construction and testing of these controllers becomes increasingly important. At the Applied Research Laboratory (ARL) of the Pennsylvania State University we have gained insight over the last ten years into the design, implementation and testing of intelligent controllers for Autonomous Underwater Vehicles (AUV). However, as AUV missions become more complex, simulation environments must be provided that achieve complete state coverage of the discrete processes of the controller while still fully exercising the continuous processes with high fidelity Monte Carlo simulations. As an illustrative example, this paper describes the current utilization of simulation in the development and testing of intelligent mission controllers for AUVs using ARL's own intelligent control architecture. Then our new paradigm for simulation based design and testing for intelligent controllers is formulated and discussed.

1 INTRODUCTION

The Applied Research Laboratory has a history in the field of autonomous underwater vehicles going back to the end of WW II. Over the years various methods have been used to develop and test the controllers for these vehicles. As the control problem and operating environments became more complex and the in-water testing became more expensive, a paradigm emerged to use simulation throughout the development phase, starting with individual component testing, followed by system integration testing, and lastly to determine expected performance over a wide range of operating conditions. In most cases the same simulation is also used in post run analysis of the mission controller to determine the exact cause of any anomalies that may have occurred in the in-water runs.

As missions have become more complex the mission controllers have been implemented as hybrid intelligent controllers (Nerode and Kohn 1993), (Passino and Ozguner 1991). Over the past decade ARL has pioneered the Prototype Intelligent Controller (PIC) architecture to create AUV mission controllers. PIC is a behavior-based perception/response system (Stover et al. 1996), (Jarriel et al. 1999), (Roeckel et al. 1999) with continuous perceptions and actions at the lower levels, while complex behaviors are implemented as discrete event-based systems. Intelligent control, and PIC in particular, is a software engineering paradigm for managing the complexity of software required to implement a complex mission. As with all software, the thorough verification and validation of the intelligent controller is extremely important.

Until recently, the missions achieved by ARL's AUVs have been of short duration and of a complexity that can still be understood by a single designer. This single design guru could develop a suite of simulations that insured that each behavior was fully exercised and tested and that the collective behaviors achieved the stated mission objectives. First, low fidelity simulations were used to test the discrete events system of the individual behaviors, while higher fidelity simulations were used to test an integrated controller. However, now with the newer more complicated missions that ARL is undertaking, this design and test strategy based on a single design guru is quickly becoming obsolete. As with many AI systems PIC is subject to the curse of dimensionality; whereas the number of behaviors increases linearly, the number of interactions between behaviors

increases exponentially. Therefore a new simulation-based paradigm for design and testing of intelligent controllers is now required. Because the actual cumulative behavior of the controller is often quite complex, systematic methodologies must be used to initially test individual behaviors, insuring complete test coverage of all states of a behavior. Next, interactions between behaviors must be fully tested, by insuring that possible paths between behaviors are exercised.

This paper details how ARL implements an AUV controller based on the PIC architecture. Our current simulation environment is described and a discussion of the enhancements we feel are necessary to our design and test paradigms for us to achieve continued success implementing more complex mission controllers is provided.

2 CONTROLLER

The ARL implementation of an AUV controller divides the controller into two main components: the intelligent **mission controller** performs the high level autonomous control while the **vehicle controller** performs the low level control of vehicle subsystems. A block diagram showing data flow of a typical controller in both the in-water and simulation environment is presented in Figure 1. The mission controller component is networked to the simulation and tested on a workstation development platform. The same controller code is compiled onto the target platform, networked to the simulation, and tested for consistency. Reliability of the in-water vehicle controller has been greatly enhanced by using the same Ada source code for both the simulation environment and the in-water vehicle. The target platform hardware and software are then used in the actual in-water vehicle. This assures that the control code used in the simulated environment is the same code used in the vehicle, eliminating the uncertainties associated with translating the code from the development environment to the hardware environment.

Some of the subsystems communicate via ethernet, while some of the sonars use specialized interface cards to communicate. Although Figure 1 shows the configuration for a pure real time (left side) or simulated environment

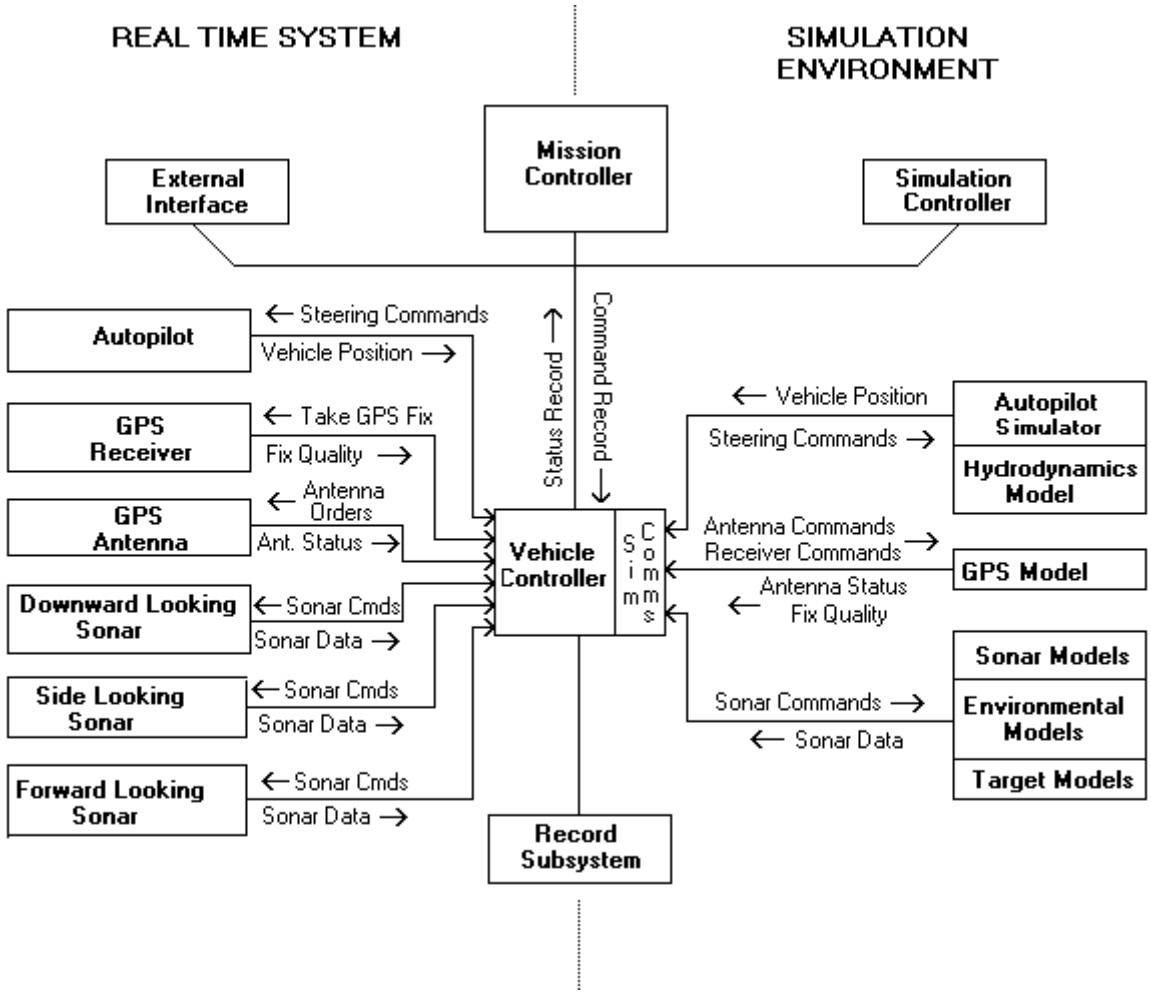

Figure 1: Block Diagram of the Mission Data Flow for both an In-Water and Simulated Environments

(right side) in practice simulation runs are often a hybrid of real time hardware and simulated components. Organizationally, in the systems developed to date the mission controller and vehicle controller have resided on separate boards. This allows the subsystem communications portion of the vehicle controller to be used with subsystem test drivers early on in the development.

Data is passed between the mission controller and the vehicle controller in one of two messages; a vehicle command message from the mission controller to the vehicle controller and a vehicle status message from the vehicle controller to the mission controller. The vehicle controller breaks the command message into separate messages for each of the subsystems, restructures them as appropriate, and forwards them to the target subsystems. It returns subsystem status, sonar data, and system time in the vehicle status message to the mission controller.

2.1 Mission Controller Architecture

The block diagram in Figure 2 shows the architecture of a typical PIC-style mission controller, implemented in Ada. The Perception Layer builds internal representations of real-world objects from external detection reports. The Response Layer performs situation assessment and makes tactical decisions based on the Perception Layer models. The Resource Layer provides a set of high level interfaces to vehicle hardware (sonars, autopilot, GPS receivers, etc.) and external data (e.g. environmental models, post-processed sonar reports). The Vehicle Controller Interface layer handles all communications with the vehicle computer.

The Vehicle Controller Interface layer decouples the remainder of the mission controller from the structure of the messages exchanged with the vehicle controller. This layer is organized hierarchically, gathering/distributing information to/from internal resources at the top level and exchanging external messages with the vehicle controller over a network link at the bottom level. The Resource Layer further isolates the Response and Perception Layers from the specifics (and even the existence) of the vehicle controller interface and supports interaction with the corresponding devices at the level appropriate to application requirements.

A thorough discussion of the internals of the Response Layer is beyond the scope of this paper (see Stover et. al. for details), but a brief look at its high level structure (see Figure 3) is warranted. The Response Engine sequences Response Layer operations and issues feedback (via Response Directives) to the Perception Layer. Response activity is organized in terms of three levels of objects. The Mission Manager formulates a high level tactical plan as a list of Behaviors to be executed at the next lower level. Similarly, Behaviors formulate plans at the next level as sequences of Actions required to accomplish their goals. The Actions at the bottom level issue commands to effector subsystems.

2.2 Mission Controller Simulation Testing

In developing a PIC-style controller for a new application we often follow a modified spiral software development model, first building and testing a fully operational controller implementing key features of the desired end product, then iterating to evolve the full-up system. We subject each iteration of the mission controller to multiple levels of simulation testing. At the start of integration testing we use a low fidelity simulation (idealized vehicle dynamics with instantaneous turns, noise-free sonar returns, etc.) to identify gross errors. As testing continues, we exercise the controller against simulation models of greater and greater fidelity.

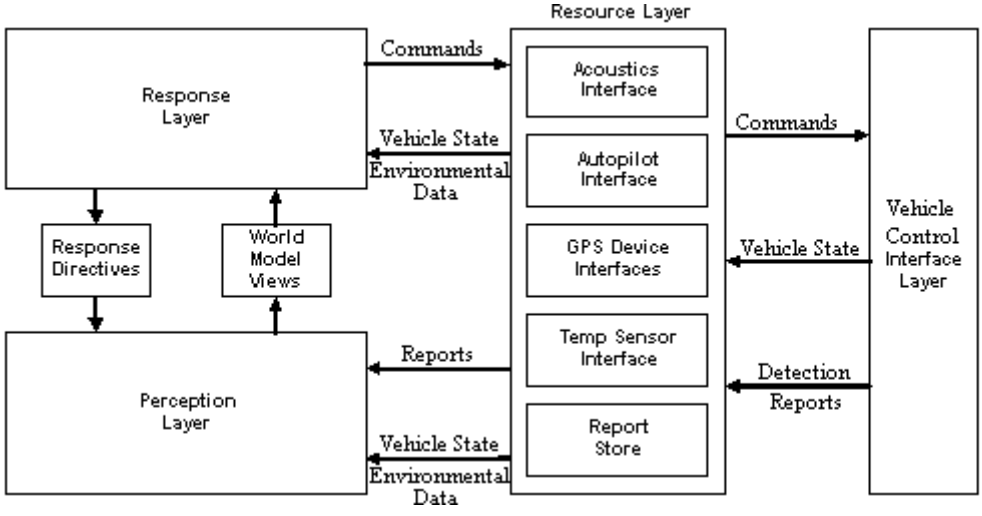

Figure 2: PIC Mission Controller Architecture

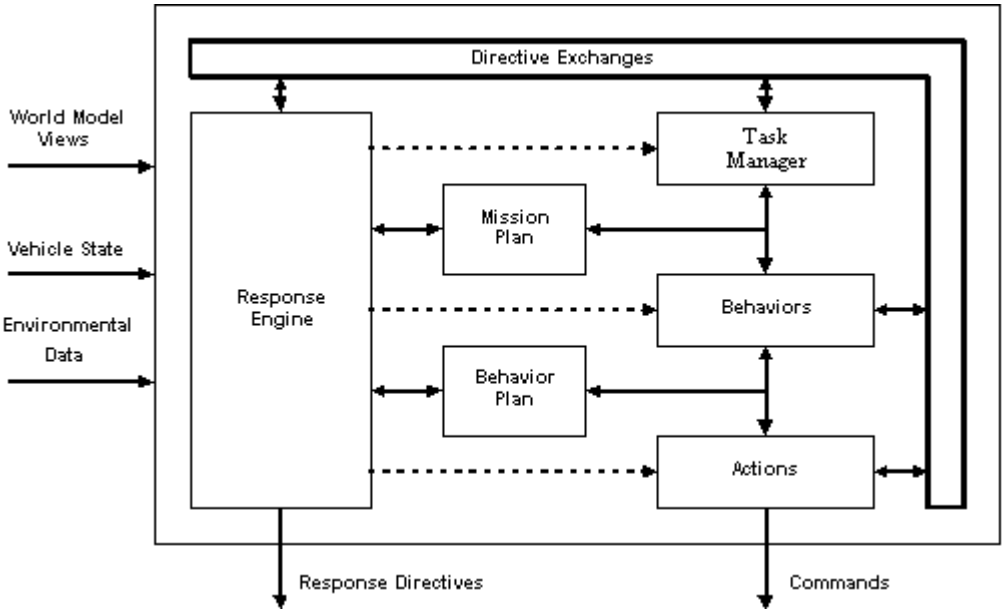

Figure 3: PIC Response Layer Architecture

Actions and Behaviors in the controller Response Layer are often associated with specific resources (e.g. GPS antenna or receiver), enabling parallel development and test of the corresponding simulation models, vehicle computer interface segments and Response Layer elements in the same iteration. Because the controller's external communications are confined to the message-based interface provided by the Vehicle Controller Communications layer, it is straightforward to capture this message traffic for playback. This playback capability enables controller performance during in-water, simulation and anechoic tank tests to be readily analyzed offline using standard profiling and debugging tools.

3 SIMULATION

The simulation is capable of modeling everything external to the mission controller. The interface between the simulation and the mission controller is well defined and, because the mission controller uses the vehicle controller time, the simulation is not constrained to run real time. The simulation runs with a fixed time step although individual subsystem models can run with a different time step (if they are time step simulations) as long as they maintain synchronization. All the vehicle subsystems and the external world effects, including vehicle hydrodynamics and sound propagation are modeled. The subsystem models can range from low to high fidelity. During the controller development stage low fidelity simulations are used to expedite turn around. When the controller is complete and in-water tests are being developed higher fidelity models are often needed. A discussion of the modules representing the simulation half of Figure 1 follows.

3.1 Simulation Controller

The simulation control module performs the simulation management duties. It reads simulation input files and writes mission/vehicle controller input files on startup, sets random number seeds to generate Monte Carlo results, and generates any debugging output during the run. The simulation controller also does any initialization required by any of the subsystem models. Both the mission controller and the vehicle controller read separate initialization files and the mission controller downloads any orders file.

3.2 Vehicle Controller

The vehicle controller used in the simulation is composed of the actual vehicle controller with modifications to the code that communicates with the non-networked subsystems and with the code that generates system time. Since the mission controller is designed with a high level interface to the vehicle controller, no changes are required in the mission controller when it is switched between the simulated vehicle controller and the actual in-water controller. The vehicle controller interface module is responsible for translating the high-level mission controller output commands into the appropriate command structures used by the various subsystems and for combining vehicle status data into the appropriate command structure for the mission controller. Some of the subsystems have specialized interface cards for communications. Modeling these subsystems includes modeling the communications interface and modifying the module in the vehicle controller that does the communi-

cation. For subsystems that communicated via a standard network (ethernet or CAN are currently used) no changes to the vehicle interface are necessary to the vehicle controller code to substitute a subsystem model for the actual subsystem.

The other change required to the vehicle controller simulation concerns system time. The vehicle controller time function normally returns real time acquired by reading an on-board real time clock. For simulation purposes the time function obtains simulation time from the simulation control module.

3.3 Autopilot Simulation

The autopilot simulation serves two purposes. First it receives steering commands from the vehicle controller (turn, pitch, speed up, go to way point, etc) and models the autopilot actions to those commands. Second, it models the vehicle movement through the water and computes a vehicle state vector (positions, velocities, accelerations, angles, turn rates, and turn accelerations) based on a selectable hydrodynamics model. A basic 3 degree of freedom model is sufficient for most testing but a 5 and 6 degree of freedom model is used when a more accurate vehicle response is required.

3.4 GPS Receiver / Antenna Simulation

The GPS antenna is modeled probabilistically rather than simulating the actual operation. When it receives an antenna up or antenna down command from the vehicle controller, the time it will take to perform the command is computed as a fixed time interval ± a random delta time. After that time delay, the antenna is modeled as having some probability of successful operation that is set by the user at startup. The GPS receiver is also modeled similarly in terms of the probability of getting a quality GPS fix.

3.5 Environmental Models

The environmental models simulate the effects of the medium on sound received by the AUV. That sound can be actively transmitted by the AUV, sound actively transmitted by another object in the medium, or sound passively radiating from another object in the medium or environmental noise. The sound is propagated through the medium from the source to the AUV receiver. Beam patterns of the AUV transmitter and receiver are modeled analytically as are radiation or reflection patterns of other objects in the medium. Sound forward or back scattered off the surface or bottom are attenuated as a function of sea state or bottom type.

The environmental models can assume iso-velocity propagation, a single sound velocity profile (SVP) for an entire run, or multiple, spatially distributed SVPs. The bottom can be flat or have a spatially varying depth. The bottom scattering can be based or a single bottom type or the bottom type can also vary based on location. The surface scattering is based on wind speed or wave height and is constant throughout a run.

The extent of environmental modeling depends upon the level of fidelity required at each stage of development. In early design and system integration testing these models may be quite simplistic. For the performance evaluation and post-run analysis these models need to be as accurate and as realistic as possible. It is also very important to validate these models with data from in-water runs and to make changes if needed. This process allows one to catch any modeling errors as early as possible and to build faith in the results provided by the model.

4 DISCUSSION

The earliest digital simulations of AUVs developed at the Applied Research Laboratory were done as analysis tools for previously developed U.S. Navy systems. During the late 1980's and throughout the 1990's ARL was funded to perform three different Guidance and Control Advanced Technology Demonstrations (ATDs) which complimented continued 6.1, 6.2, and 6.3 Guidance and Control efforts funded through the Office of Naval Research (ONR). This combination of guidance and control work required the development of highly sophisticated AUV controllers with which to showcase the technology demonstrations. The methods presented in this paper evolved to meet the challenge of developing, testing, and maintaining these controllers in a constantly changing research environment.

The digital simulations previously developed provide the basis for the environmental and hydrodynamic models presented in this paper. These models have evolved over the past thirty years and represent a tremendous investment of time and a vast compendium of knowledge. As the PIC controller moves in new areas outside the domain expertise of ARL, creating high fidelity simulations becomes more difficult. A work model of combining our simulation skills with the knowledge of an expert in the new domain to create a high fidelity simulation has worked on small projects but will require more consideration.

An issue not addressed in our current work is reachability of individual behaviors. Because the controller starts in a default startup state, there is no mechanism to start in a behavior other than the default behavior. Often the logic path to an individual behavior is direct (i.e. a behavior can be ordered to ask for control at a particular time) but in some cases the only path is through other behaviors (i.e. when a behavior asks for control only when directed by another behavior). In other cases, a behavior may be designed to address a rare event that may be difficult to simulate.

The techniques described in this paper have been used at ARL for the development of numerous PIC based AUV

controllers. New areas identified as candidates for PIC controllers are medical, oceanographic sampling, and ship damage control. As the PIC controller architecture is applied to new applications, the challenge will be to develop the simulations, both low fidelity and high fidelity, needed to model the new applications and the capability of more completely testing the controllers.

ACKNOWLEDGMENTS

The development of the Prototype Intelligent Controller and much of the simulations described herein have been supported by the Office of Naval Research.

REFERENCES

Jarriel, M.,D. Adams, M. Hissa, and F.Vasques. 1999. An effective transfer of autonomous control technology. In *Proceedings of the 1999 - 14th IEEE International Symposium on Intelligent Control / Intelligent Systems and Semiotics*, IEEE, Piscataway, NJ.

Nerode, A.and W. Kohn. 1993. Models for hybrid systems: automata, topologies, controllability, observability. In *Hybrid Systems*, vol. 736, *Lecture Notes in Computer Science*, A. Nerode and W. Kohn, Eds. Berlin: Springer-Verlag.

Passino, K. M. and U. Ozguner. 1991. Modeling and analysis of hybrid systems: examples. In *Proceedings of the IEEE International Symposium on Intelligent Control*, 251-6. IEEE, Piscataway, NJ.

Roeckel, M. W., R. H. Rivoir, and R. E. Gibson. 1999. A behavior based controller architecture and the transition to an industry application. In *Proceedings of the 1999 - 14th IEEE International Symposium on Intelligent Control / Intelligent Systems and Semiotics*, IEEE, Piscataway, NJ.

Stover, J. A., D. L. Hall, and R. E. Gibson. 1996. A fuzzy-logic architecture for autonomous multisensor data fusion. In *IEEE Transactions on Industrial Electronics, Vol. 43, No 3*, IEEE, Piscataway, NJ.

AUTHOR BIOGRAPHIES

MICHAEL W. ROECKEL is a Senior Research Engineer in the Autonomous Control and Intelligent Systems Division of the Applied Research Laboratory, Pennsylvania State University. He received a M.S. degree in Acoustics and a B.S. degree in Computer Science from the Pennsylvania State University. In addition to his long standing interest in simulation, Mr. Roeckel's current research direction is into intelligent controller technology.

ROBERT H. RIVOIR is an Associate Research Engineer in the Applied Research Laboratory at the Pennsylvania State University. He received a B.S. degree in Mathematics from Stanford University and an M.S. degree in Computer Science from Pennsylvania State University. He is a member of the IEEE Computer Society.

RONALD E. GIBSON is a Research Engineer in the Autonomous Control and Intelligent Systems Division at the Applied Research Laboratory, Pennsylvania State University. He has an M.S. in Computer Science from the Pennsylvania State University. He has 30 years of experience in autonomous control architectures and co-holds a patent in "A Processing Architecture for Autonomous Systems".

STEPHEN PAUL LINDER is an assistant professor of Computer Science at SUNY Plattsburgh. Steve received his B.S. degree in Mechanical Engineering from the Massachusetts Institute of Technology, and a M.S. and Ph.D. from Northeastern University, Boston in Computer Systems Engineering. Currently, Steve is working on the automated catching of ground balls. More details can be found at http://www.coe.neu.edu/~spl.