Intelligent Control of Autonomous Underwater Vehicles: The Orca Project*

Roy M. Turner
Marine Systems Engineering Laboratory
Marine Science Center
Northeastern University
East Point, Nahant, MA 01908 (USA)
Phone: (603)862–2980 FAX: (603)862–3493
E-mail: rmt@cs.unh.edu WWW: http://cdps.cs.unh.edu

Abstract

Establishing a useful presence in the ocean is becoming increasingly important to science, industry, and the military, yet the undersea environment is hostile to human presence. Autonomous underwater vehicles (AUVs) offer a solution. Before widespread use of AUVs is practical, however, mechanisms for intelligent control must be developed. In this paper, we report on the Orca project, which has the aim of creating a robust, intelligent, mission-level controller for long-range ocean science AUVs. Orca is now being built and tested in simulation; in the future, it will undergo in-water tests aboard the Marine Systems Engineering Laboratory's EAVE-III vehicles. In this paper, we discuss the motivation behind the project, the Orca program, and our current status and future work.

The next few decades will see humans striving to establish an increased presence in and on the oceans. Military and industrial motivations for this are obvious, though we will likely be surprised by the range of new applications developed. Global change monitoring and other environmental research dealing with the ocean is becoming increasingly urgent, as is basic ocean science research, as we strive to understand our global environment and, in some cases, stave off or remediate environmental catastrophes. Mariculture is increasingly important, and the future is even likely to see interest in floating or submerged habitats, airports, or other large ocean structures.

Common to all of these is the need for tools to help humans work in the challenging and dangerous undersea domain. Remotely-operated vehicles (ROVs) fill some roles in this area, but suffer from problems of limited range, tether entanglement, and the need for constant supervision by human pilots. Many tasks could be better and more efficiently performed by undersea vehicles that can operate on their own power and use their own initiative, free from tethers and the need for human supervisors. Such vehicles, called autonomous underwater vehicles (AUVs), could remain on-station for time-series studies, traverse long distances or work in teams to acquire simultaneous spatially-distributed data, perform routine inspection and maintenance tasks on undersea structures, an in general extend humanity's reach into the ocean.

The current state of the necessary hardware and basic control theoretic technology is such that AUVs can be built and fielded for simple tasks; indeed, many have been [4]. However, complicated, under-specified, or long-duration missions remain essentially out of reach. Unfortunately, this covers most realistically-useful missions in ocean science and ocean engineering. Missions cannot in general be completely specified in detail up front, nor can one expect the ocean environment or vehicle hardware to cooperate with the goals of the mission. Instead, missions often can at best be defined in general terms (e.g., "sample data in regions of interest in the area," "look for leaks," etc.); our knowledge of the ocean environment is incomplete, sensors are imprecise, real hardware will not perform ideally, and an AUV will encounter unknown processes and other, unpredictable agents during its missions. What is required is an *intelligent* mission controller for AUVs, one that can flesh out skeletal mission specifications, accept new commands, handle unanticipated events, and in general adapt to the demands of its mission and environment.

Autonomous underwater vehicles have been under development for many years (see [4] for a survey). The Marine Systems Engineering Laboratory (MSEL) has been a pioneer in this effort, developing a line of AUVs beginning with the EAVE-EAST [12] vehicle and progressing through our two current EAVE-III vehicles and our long-range AUV that is under development. Our aim has been twofold: to develop useful AUV technology for ocean science and ocean engineering applications, and to extend the state of the art in intelligent control. Our most recent effort is the Orca project [22; 19], whose goal is to create an intelligent, adaptive, robust mission controller for long-range ocean science AUVs. Orca is an adaptive, *schema-based reasoner* [19].

In this paper, we discuss Orca program and the current status of the Orca project.

AUV CONTROL

*This work is funded in part by grant BCS–9211914 from the National Science Foundation. The author is also affiliated with the Department of Computer Science, University of New Hampshire, and thanks the CDPS Research Group there for its helpful comments throughout this work.

Aside from the pragmatic considerations of developing an intelligent AUV controller, the task provides an interesting and challenging domain for AI research. It is at once a real-world domain without real-time constraints so tight that a planning program could not reasonably be expected to operate within them.

Real-world domains provide many challenges for AI programs, including incomplete information, uncertainty, and the possible presence of unpredictable processes and agents in the world. The result is unpredictability and lack of guarantees that plans or other commitments to future actions will actually succeed.

The AUV domain is more ornery than most real-world domains. We have relatively little knowledge about the undersea environment, so naturally the knowledge we can give an AUV is limited. Compounding the usual uncertainty and incomplete knowledge problems are the notoriously poor sensor modalities available to AUVs operating in realistic settings. Vision is practically useless except in the clearest of water, and most desirable locales for AUV missions are far from having clear water—e.g., the North Atlantic, with its high density of organisms and particulate and organic matter in the water column. The most usual sensor for AUVs is sonar, which has a wealth of problems, including lack of spatial resolution at any significant range and multipath errors.

On the other hand, AUVs are not cruise missiles. The EAVE-III vehicles, for example, have a cruising speed of roughly 2 knots. This, combined with the fact that things happen slowly underwater in general, means that an AUV controller is not bound by hard real-time constraints. On vehicles such as EAVE that have lower-level control software on board [3], the controller can reduce its time constraints further by intelligently presetting the low-level software ahead of time to handle most situations that call for quick responses.

The AUV control task imposes several requirements on any approach seeking to solve it. First, the controller needs the ability to create and execute plans of action. This is a contentious statement in AI currently [e.g., 1]; however, it is unclear how any purely reactive approach and/or any approach relying on emergent behavior to achieve the users' goals will succeed. Typical missions for AUVs include such goals as: "collect data at location x at time t"; "rendezvous with the support ship at x at t"; "perform the set of steps S to construct the underwater structure"; and so forth. These require commitments to future actions; they require planning and resource management to ensure that these commitments can be met. This is especially true in cooperative settings, for example when several AUVs work together to sample or photograph an area or construct an underwater structure.

Though planning is required, "traditional" AI planning—i.e., create a complete, detailed plan of action, then execute it—simply won't work. As in most realistic domains, what is needed here is what has variously been called *reactive planning* [e.g., 10] (though not the extreme form that involves *no* planning!) or *adaptive reasoning* [19]. The basic idea is that proposed by McDermott [13] many years ago: interleave planning and plan execution, so that the planner does not over-commit to details of the plan, only to be sabotaged by changes in the world or its own ignorance.

The controller also needs to be interruptible—that is, to be able to interrupt a plan it is working on to handle unanticipated events or possibly to work on another mission, then perhaps later resume its original plan. The importance of unanticipated events cannot be overestimated in this domain. Due to the AUV controller's incomplete, uncertain knowledge, and the possible presence of other agents, it will quite often be surprised, often unpleasantly. Some of these events directly impact the mission plan and can be handled by changing it. However, some only indirectly impact the plan, though they may impact the overall health (or survival) of the AUV; these may require interrupting the current plan, handling them, then resuming it. A controller also needs to be interruptible when working with others, for example, when on a cooperative sampling mission. In these cases, goals or requests from other agents will arrive asynchronously and should be handled.

Another requirement is that the AUV controller be context-sensitive. This seems obvious: all reasoners need to adjust their actions to the exigencies of the situation they are in. How to get them to easily do this, however, is problematic and has only recently become a research area in its own right [e.g., 5; 6]. Below and elsewhere [18; 19], we describe one approach to context-sensitive reasoning that we believe holds promise for real-world systems.

RELATED WORK

Many others have also attacked problems similar to AUV mission-level control. Purely reactive approaches, such as Agre & Chapman's [2] or Brooks' [7], have the drawback of not being able to commit to future plans or actions; this complaint can also be made about rule-based systems [9]. Hybrid architectures, employing both reactive and planning modules [e.g., 8; 14], can suffer from the incommensurability between "representations" used by the pieces: what is the shared vocabulary of a reactive module and a planning module, for instance, and how are they to communicate their states and requests to the other? Integrated approaches that seek to combine reactive and deliberative elements in a seamless whole [e.g., 10; 11] are much closer to our approach. We extend and bring to these existing approaches increased attention to event handling and context-sensitive reasoning. Other schema-based approaches also exist that are related to our approach, for example JULIANA [15] and JUDIS [17]. These should be viewed as complementary rather than competing; JULIANA, though not a real-world controller, can serve as inspiration for how to autonomously build our approach's knowledge structures, and our research group is beginning to investigate integrating ideas from JUDIS and Orca in the domain of cooperative distributed problem solving.

ORCA

Orca is a schema-based adaptive reasoner. It is designed to reside in the top-most level of the MSEL/EAVE

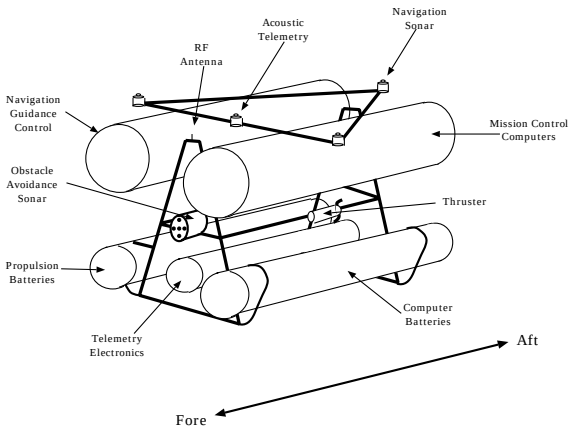

Figure 1: The EAVE-III AUV.

software architecture [3], a hierarchical architecture for controlling AUVs. The initial target AUV is the EAVE vehicle, a diagram of which is shown in Figure 1; the ultimate target is MSEL's planned long-range AUV, which will be a hydrodynamic vehicle capable of extended missions that require diving to full (average) ocean depth.

Orca's knowledge structures are *schemas*, explicitly-represented packets of related problem-solving knowledge. *Procedural schemas* (p-schemas) control how Orca takes actions to achieve its goals; they are capable of representing plans, scripts, or rules. *Contextual schemas* (c-schemas) represent kinds of situations Orca might reasonably expect to find itself in; they provide information to the various pieces of Orca to ensure that the program's behavior is always appropriate for its context.

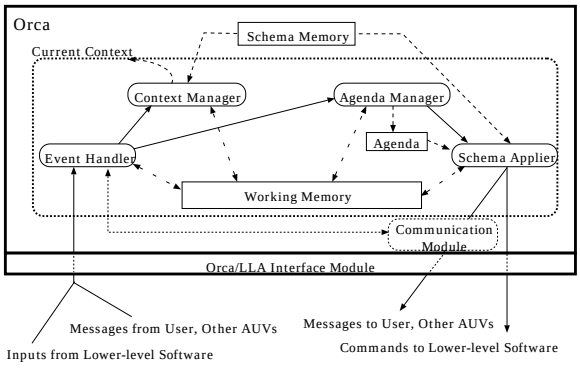

Figure 2: Orca.

Figure 2 shows Orca's internal modular structure. Agenda Manager is the module responsible for selecting what goal or goals Orca will work on at each point during a mission. Schema Applier is responsible for finding a p-schema fitting the current goal, then interpreting (applying) it to take actions directed at achieving the goal. Actions can be internal (e.g., inferences), commands to the low-level software (e.g., "move to (x,y,z)"), or messages to others sent via the communication module. All incoming information enters Orca via the Event Handler, which is responsible for assessing the state of the evolving problem-solving situation and detecting and handling any events. Context Manager is responsible for maintaining Orca's view of what its current context is (via its c-schemas) and for coordinating with Orca's other modules to ensure context-appropriate behavior. It builds a knowledge structure, the *current c-schema*, that represents the current situation; as the figure is intended to suggest, this structure provides a backdrop for all of Orca's reasoning. Long-Term (schema) Memory holds and organizes Orca's schemas, and Working Memory contains information about the current situation.

SCHEMAS

Orca's primary knowledge structures are *schemas*. Procedural schemas (p-schemas) are hierarchical plan-like in nature. They contain a description of the situation and goal they are appropriate for, a set of steps, and a set of ordering constraints or directives. The steps of a p-schema can be either goals, other p-schemas, or executable actions ("xacts"), providing a range of commitment to plan details from low to high, respectively. Steps are organized by the ordering information. This takes the form of directives specifying how steps should be sequenced, or constraints that relate one step to others. The ordering vocabulary is sufficiently rich to allow sequential, parallel, and non-deterministic execution of steps; it also provides if–then–else and looping structures, as well as mechanisms to suspend p-schema application, to explicitly handle some plan failures, and, in future, to represent resource- and time-related inter-step constraints.

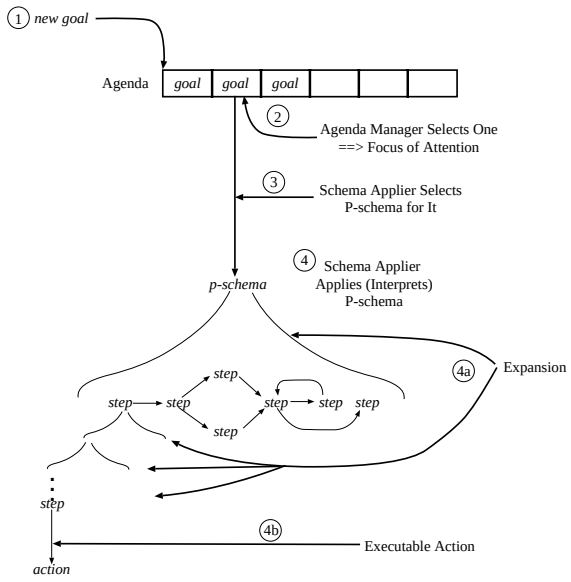

Figure 3: The schema application process.

Contextual schemas contain both descriptive and prescriptive information about the contexts they represent. The descriptive information can be used as a source of top-down expectations about unseen features the agent's situation and to help it make sense of those features it has observed. The prescriptive information is used to ensure that the agent's behavior is appropriate for its current situation. Such information includes: event-handling infor-

mation; attention-focusing information; schema-selection information; and "standing orders", suggestions for things that should automatically be done when entering a new situation. These latter take the form of goals to activate and behavioral parameter settings to put in force.

Both types of schemas are organized in a content-addressable memory so that features of the situation in effect at the time a schema is needed can directly index the schema; this kind of memory has been used in other schema-based [19] and case-based [e.g., 16] reasoning programs. The memory is organized such that more general schemas organize, or "index", more specific ones. This has the salubrious effect that the most specific schema possible is always found for a given situation. If a highly-specialized schema is available (e.g., a p-schema for cooperative adaptive sampling missions), then that will be returned; however, if not, the agent is not paralyzed. Other, more general schemas will always be available (e.g., a p-schema to perform cooperative missions, or one to perform adaptive sampling) that can likely be made to fit the situation. In the worst case, highly general schemas for from-scratch problem solving—i.e., that compose other p-schemas—may be found and used, though this has yet to be implemented. The result is the reasoner gains access to strong methods where possible, but has the ability to fall back on increasingly weak methods as needed.

SCHEMA APPLICATION

The process of schema application is shown in Figure 3. When new goal arrives from the user, another agent, or Orca itself (e.g., as a result of handling unanticipated events), it is placed on the agenda. When Agenda Manager focuses attention on that goal, Schema Applier (SA) begins by finding an appropriate p-schema for the goal. It may be aided in this by suggestions from the current c-schema about what p-schemas are appropriate in this context; these suggestions allow SA to short-cut some or all of the memory search that would otherwise be necessary.

Once a p-schema is found, then application begins. SA creates a knowledge structure to hold the instantiated p-schema and expands the p-schema by creating a plan network to represent its steps. The first step is then identified. If it is an xact, then SA takes the action specified in its representation (at the current time, by executing a Lisp function). This will result either in some internal action (e.g., an inference being made), a command to the low-level architecture (e.g., "move to (x,y,z)"), or a message sent to another agent via the communication module.

If the step is not an xact, however, then additional work must be done. If the step is a goal, then a p-schema is found with which to achieve it, and that p-schema becomes the current step. If it is a p-schema, then SA seeks first to find a specialization of the p-schema better fitting the current situation, then the process of application is begun recursively by expanding that p-schema into its steps, etc., until an xact is ultimately found.

Specialization at run-time of steps is done to tailor the schema more closely to the actual current problem-solving situation. Also, by not expanding those portions of plans it does not have to, SA puts off committing to future details that may be obviated by changes in the world or caused by its own actions. It will likely make better choices for how to expand future steps by waiting, since it will have more current information at that time. All of these aspects of the schema application process help to ensure that Orca's behavior will be well adapted to its current situation.

The schema application process can be interrupted as needed to allow Orca to respond to changes in the situation or new goals that arise. After each step is executed, SA checks a semaphore associated with the agenda to see if Agenda Manager wishes control of the agenda, e.g., to refocus attention. If so, SA suspends what it is doing; a record of the state of application of the current p-schema is always kept, so later execution can pick up where it left off.

Work on a goal can also be interrupted when the execution of an xact extends over a significant amount of time. Orca can suspend the action's parent p-schema[1] until the action is done or until predictions about the results of the action are met. P-schemas can also explicitly call for their application to be suspended for some period of time or until some condition is met. Both of these interruption mechanisms allow interleaving of actions in service of multiple goals.

ATTENTION FOCUSING

Attention focusing is done by Agenda Manager (AM). Upon any significant change to the world or the agenda, or when requested by another module, AM assesses the relative importance of all goals currently on the agenda, assigning them a priority measure. It then selects the one with the highest priority as the current focus of attention.

The assessment process is done[2] in the current version of Orca by a fuzzy rule-based system (RBS) whose rules come from both a default set and from the current c-schema. Thus the assessment process is strongly colored by the context Orca finds itself in.

All actions Orca takes apart from simple things such as setting parameters are mediated by goals on the agenda. This includes response to unanticipated events, even very high-priority responses such as "abort mission". The major benefit of this is that all potential activity is compared using the same criteria by the same module, thus assuring that the measures of importance for doing one thing rather than another are commensurate and correspond to what is appropriate in the current context.

EVENT HANDLING

In any real-world system, events will occur that are unpredicted, or at least that cannot be so thoroughly planned for that they do not need to be handled as they arise. Event Handler (EH) is the Orca module respon-

[1] Actually, only the portion of the p-schema that depends on the outcome of the action is suspended.

[2] Or will be, by the time of publication.

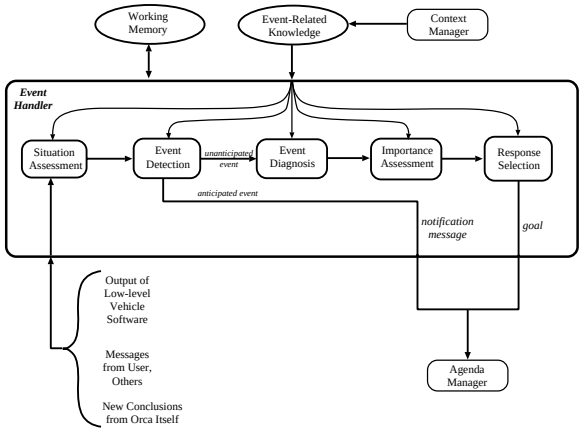

Figure 4: Orca's Event Handler module.

sible for assessing the current state of the world and for detecting and handling all events. Figure 4 diagrams EH's event-handling process.

Two kinds of events are handled by EH. Anticipated events are those that SA predicted based on its activities, either by a p-schema explicitly posting a prediction to the Working Memory (WM) or by an xact's anticipated results being posted as predictions. When an anticipated event is detected, a message to that effect is sent to Agenda Manager, which then does whatever has been requested by Schema Applier.

Unanticipated events are not necessarily novel, just unpredictable. For example, an AUV controller can to some extent expect to run out of power, however, when and where are in general unknowable.

An unanticipated event requires more processing than one that was anticipated. First, it is more difficult to detect, since EH does not know what to look for in advance. Second, it must be diagnosed to arrive at its meaning, that is, the underlying cause. For example, the surface form of an event might be "motion stopped"; this could be caused by many things, including: caught in a net, entangled in kelp, aground, loss of power, or thruster malfunction. Third, the importance of the underlying event is assessed to determine if a response should be taken. If so, the the fourth aspect of event handling is to select and activate a response. In our approach, all responses are couched in terms of goals that are then sent to AM for activation; this way, whether or not a response is immediately taken depends not only on EH's view of the situation, but on AM's knowledge of the entire context of the mission, including other goals.

Most of EH's work is done by instances of fuzzy rule-based systems. As with AM's RBS, these get their rules both from a default ruleset as well as from the current c-schema.

We recognize the limitations imposed on Orca's event-handling abilities due to the linear nature of EH's processing. Elsewhere, we have proposed a different approach, drawing on work in AI in medicine, that is less linear and that has provision for tentative diagnosis of, assessment of, and response to events [20]. This will be a topic of future research.

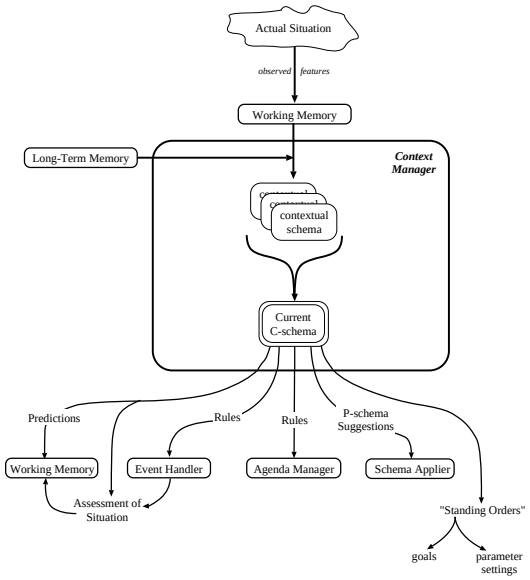

Figure 5: Orca's Context Manager.

CONTEXT MANAGEMENT

Throughout this paper, we have stressed the importance of context-sensitive behavior. Context Manager (CM) is the Orca module responsible for this. Figure 5 is a diagram of CM's operation.

When CM notices a change in the situation, it asks the schema memory (LTM) to find all c-schemas that are good matches for the changed situation. These are then merged to form the *current c-schema*, a composite knowledge structure that in essence represents Orca's commitment to what the current situation is. By merging c-schemas, Orca avoids an explosion in the number of c-schemas needed to represent all situations in which it might find itself. Instead, only those contexts need be represented whose implications for behavior cannot be captured by combining existing schemas. For example, suppose an AUV is operating in Puget Sound during an incoming tide on a search mission when power becomes low; this exact situation is not likely to have ever occurred in the past. However, the context can be captured by retrieving and merging schemas representing the contexts of: being in Puget Sound, operating in the presence of currents, being on a search mission, and having low power.

Once the current c-schema is formed, CM can parcel out information from it to help Orca's other modules. For example, predictions and an assessment of the situation based on the contextual knowledge are sent to EH, as are rules for event handling; rules are sent to AM for attention focusing; and suggestions for p-schemas appropriate for the current context are sent to SA. "Standing orders", that is, goals to activate and parameters to set based on the context, are activated directly by CM.

We anticipate work on implementing CM to be well underway by the time of publication. An earlier implementation of context management has given us insight into many of the problems and requirements [19], and context

merging will be a promising area of near-future research.

CONCLUSIONS AND FUTURE WORK

AUV control, like other real-world tasks, requires an adaptive reasoner, one whose behavior is always in close congruence with its context, and one that can plan, yet not over-commit, and that can react to changes in the problem-solving situation. The Orca project is aimed at creating such a program for controlling ocean science AUVs.

When complete, Orca will be a context-sensitive, robust, adaptive problem solver capable of controlling AUVs on useful missions. At the time of writing, an initial version of Orca, with simple EH and AM modules and no CM, is being used in simulation by another research group. Orca 2.0 is under construction; this version will have a Context Manager and more competent versions of EH and AM that make use of fuzzy rule-based systems to do their jobs. A future version of Orca, to be completed within a year, will focus on uncertainty management and spatiotemporal and resource-based reasoning.

During this phase of the project, all development and evaluation is being done in our simulation testbed, SMART [21]. In a future phase of the project, Orca will be ported to vehicle computers to control actual AUVs during in-water tests and missions.

Simultaneously, we will be exploring the applicability of the schema-based reasoning mechanism for other intelligent agents, in particular intelligent Internet agents ("softbots"). We believe that our past experience with SBR in MEDIC, our current experience in Orca, and our future experience in the softbot area will allow us to evaluate SBR's usefulness and applicability as a general approach to intelligent agent research.

References

[1] P. E. Agre. Computational research on interaction and agency. *Artificial Intelligence*, 72:1–52, 1995.

[2] P. E. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268–272, 1987.

[3] D. R. Blidberg and S. G. Chappell. Guidance and control architecture for the EAVE vehicle. *IEEE Journal of Oceanic Engineering*, OE–11(4):449–461, 1986.

[4] D. R. Blidberg, R. M. Turner, and S. G. Chappell. Autonomous underwater vehicles: Current activities and research opportunities. *Robotics and Autonomous Systems*, 7:139–150, 1991.

[5] P. Brézillon, editor. *Proceedings of the 1993 IJCAI Workshop on Using Knowledge in its Context*, Chambéry, France, August 1993. IJCAI.

[6] P. Brézillon and S. Abu-Hakima, editors. *Working Notes of the 1995 IJCAI Workshop on Modelling Context in Knowledge Representation and Reasoning*, Montreal, Canada, 1995. IJCAI.

[7] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA–2(1):14–23, 1986.

[8] P. R. Cohen, M. L. Greenberg, D. M. Hart, and A. E. Howe. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3):34–48, Fall 1989.

[9] R. Davis, B. G. Buchanan, and E. H. Shortliffe. Production rules as a representation for a knowledge-based consultation system. *Artificial Intelligence*, 8:15–45, 1977.

[10] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning: An experiment with a mobile robot. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 677–682, Seattle, Washington, 1987.

[11] B. Hayes-Roth, R. Washington, R. Hewett, M. Hewett, and A. Seiver. Intelligent monitoring and control. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 243–249, Detroit, Michigan, 1989.

[12] 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.

[13] D. McDermott. Planning and acting. *Cognitive Science*, 2:71–109, 1978.

[14] R. B. Schudy and C. N. Duarte. Advanced autonomous underwater vehicle software architecture. In *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology*, pages 9–22. The Oceanic Engineering Society of the IEEE, 1990.

[15] H. S. Shinn. Abstractional analogy: A model of analogical reasoning. In *Proceedings of the DARPA Workshop on Case-Based Reasoning*, pages 370–387, Clearwater Beach, Florida, 1988.

[16] K. Sycara. *Adversarial Reasoning in Conflict Resolution*. PhD thesis, School of Information and Computer Science, Georgia Institute of Technology, 1987. (Technical report #GIT–ICS–87/26.).

[17] E. H. Turner and R. E. Cullingford. Using conversation mops in natural language interfaces. *Discourse Processes*, 12(1):63–91, 1989.

[18] R. M. Turner. When reactive planning is not enough: Using contextual schemas to react appropriately to environmental change. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, pages 940–947, Detroit, MI, 1989.

[19] R. M. Turner. *Adaptive Reasoning for Real-World Problems: A Schema-Based Approach*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1994.

[20] R. M. Turner, P. S. Eaton, and M. J. Dempsey. Handling unanticipated events in single and multiple AUV systems. In *Proceedings of the IEEE Oceanic Engineering Society Conference (Oceans '94 OSATES)*, volume II, pages 125–130, Brest, France, 1994.

[21] R. M. Turner, J. S. Fox, E. H. Turner, and D. R. Blidberg. Multiple autonomous vehicle imaging system (MAVIS). In *Proceedings of the 7th International Symposium on Unmanned Untethered Submersible Technology (AUV '91)*, pages 526–536, Durham, NH, 1991.

[22] R. M. Turner and R. A. G. Stevenson. ORCA: An adaptive, context-sensitive reasoner for controlling AUVs. In *Proceedings of the 7th International Symposium on Unmanned Untethered Submersible Technology (UUST '91)*, pages 423–432, 1991.