Published in *Proceedings of the Seventh International Symposium on Unmanned Untethered Submersible Technology (UUST)*, Durham, NH, September, 1991.

Orca: An Adaptive, Context-sensitive Reasoner for Controlling AUVs

Roy M. Turner
Marine Systems Engineering Laboratory
& Department of Computer Science
Marine Programs Building
University of New Hampshire
Durham, NH 03824
rmt@cs.unh.edu

Robert A.G. Stevenson
Department of Computer Science
Kingsbury Hall
University of New Hampshire
Durham, NH 03824

Abstract

Controlling an autonomous underwater vehicle is a very difficult task that stretches the limits of existing artificial intelligence (AI) technology. For very limited missions of short duration, pre-planning approaches may be sufficient. However, for realistically complex and/or long-duration missions, an AUV needs to have an onboard reasoning system that can adapt the AUV's behavior appropriately for the evolving mission, system status, and environment. The reasoner must cope with uncertain, incomplete knowledge, as well as a world in which unknown, possibly unpredictable, processes and agents exist. It must be capable of making commitments to future actions to achieve mission goals, yet be reactive to demands arising from unanticipated events. Its responses to events, its decisions about which of its goals to attempt to achieve, and other facets of its behavior must be appropriate for the current problem-solving context.

This paper describes ORCA, an adaptive, context-sensitive reasoning system being developed to control AUVs. ORCA is a reactive planner, interleaving plan generation and execution. It makes use of plan-like structures (called *procedural schemas*) to commit to future courses of action, yet remains ready to handle unanticipated events as they arise. ORCA is also a context-sensitive reasoner. It makes use of *contextual schemas* to represent prototypical problem-solving situations. ORCA uses these to decide, in a context-specific way, how to respond to unanticipated events, focus attention, pre-set reflexes, and set operating parameters ("standing orders"). ORCA is currently being designed and implemented to control the UNH Marine Systems Engineering Laboratory's EAVE and long-range AUVs.

An autonomous underwater vehicle (AUV) may be preprogrammed to carry out very limited missions of short duration. For realistically complex missions, however, and especially for missions of long-duration, this is impractical. Once the mission is underway, an *a priori* plan may quickly become obsolete and fail due to the uncertain, changing, and unpredictable nature of the underwater environment. What is needed is an onboard intelligent controller capable of creating and executing a plan for carrying out the mission. Not only must the controller be able to plan ahead (e.g., for sequencing tasks and coordinating with other agents), it must be able to react to changes in its environment and to unanticipated events that may result from changes or from incomplete and uncertain knowledge.

In this paper, we describe one approach to the problem of creating an intelligent AUV controller. Our approach, called *schema-based reasoning* [Turner, 1989a; Turner, 1989b], has been previously demonstrated in the domain of medical consultation systems, a domain in which the problem-solving environment is mildly unpredictable and changing. We are now implementing and testing our ideas in the much more demanding domain of AUV mission control. Our testbed for this work is ORCA, a schema-based, context-sensitive controller for the EAVE and long-range AUVs under development at the UNH Marine Systems Engineering Laboratory (MSEL). ORCA uses schemas to represent both compiled plans of action and kinds of situations

it knows about. Both kinds of schemas are used in a reactive manner, to allow nearly automatic, context-sensitive behavior while remaining sensitive to the demands of a changing, unpredictable environment.

AUV Control Issues

AUVs have many uses in ocean science and engineering, the military, and industry. Many of the missions that are being undertaken and that will be undertaken in the future are quite complex, involving large numbers of actions, some or all of which may have to be performed in some particular order. Many missions may be carried out in cooperation with other agents, or in situations where coordination with the actions of another is important. Consequently, there needs to be some mechanism for creating a plan for the AUV to carry out to accomplish its mission.

A naive approach is to have a human or automatic planner create a plan for the mission, load a representation of that plan into the AUV, and send it off. The problem of automatic plan generation is one that has received a great deal of attention in artificial intelligence (AI). Good planning programs exist; for example, one of the most capable current AI planners is SIPE [Wilkins, 1984], which can take into account resource constraints as well as goal interactions. A planner can be designed to create a plan that has a branching structure to take into account some uncertainty about conditions that may occur during the mission.

Unfortunately, this approach has many serious drawbacks. In any real-world domain, a planner (whether human or machine) suffers from incomplete knowledge of the world and uncertainty; this is especially true of the underwater domain. In addition, the real world is constantly changing in ways that are difficult or impossible to predict due to unknown, uncontrollable processes or other agents operating in the world. What this means for the AUV is that any plan it is given ahead of time is likely to fail or at least need modification at run-time: assumptions on which the plan was based are likely to be violated due to uncertainty, incomplete information, or a changing world. If the plan is branching, then the number of branches necessary to cope with the possible contingencies can rise explosively—assuming the planner can predict those contingencies at all, which is problematic. This is a particularly severe problem for complex missions, missions in complex environments, long-duration missions, and missions involving other agents—in other words, for most interesting missions. Preplanning approaches are hindered as well by the fact that much of the information needed to create the plan may not become available (via, perhaps, the AUV's own sensors) until some time during the plan's execution.

These complications argue for an AUV to have an intelligent mission controller on board, one that can create and execute plans of actions as well as react to unanticipated events caused by uncertainty, incomplete knowledge, and a changing world. Given a mission description consisting of goals, constraints, partial (high-level) plans, and information about the worksite and environment, such a planner would then create a plan for the mission at some level of detail. As necessary and possible, the planner would refine its initial plan during the mission based on new information gained about the environment via its sensors. As the mission progresses under the control of the planner, the planner would monitor the plan's execution. When the plan's execution deviates from the desired results, the planner would take action to correct its plan. When an unanticipated event occurs (e.g., a leak, prop fouling, etc.), the planner would decide if the event is important, how to handle the event, and whether or not the overall plan needs to be changed in response. The planner should remain cognizant of its context at all times so that its plans, assessment of events, and reactions to events are appropriate for the context in which it finds itself. In addition, the design of an intelligent controller must take into account the real-time constraints of the mission control task: its behavior should be as automatic as possible, while still remaining appropriate for the context.

Functional Overview of Orca

ORCA is designed to be the topmost level of the MSEL software architecture for AUV control [e.g., Blidberg & Chappell, 1986], as shown in Figure 1. This architecture has been designed to control MSEL's EAVE AUVs. ORCA will combine the functions of mission-level situation assessment, planning, and plan execution.

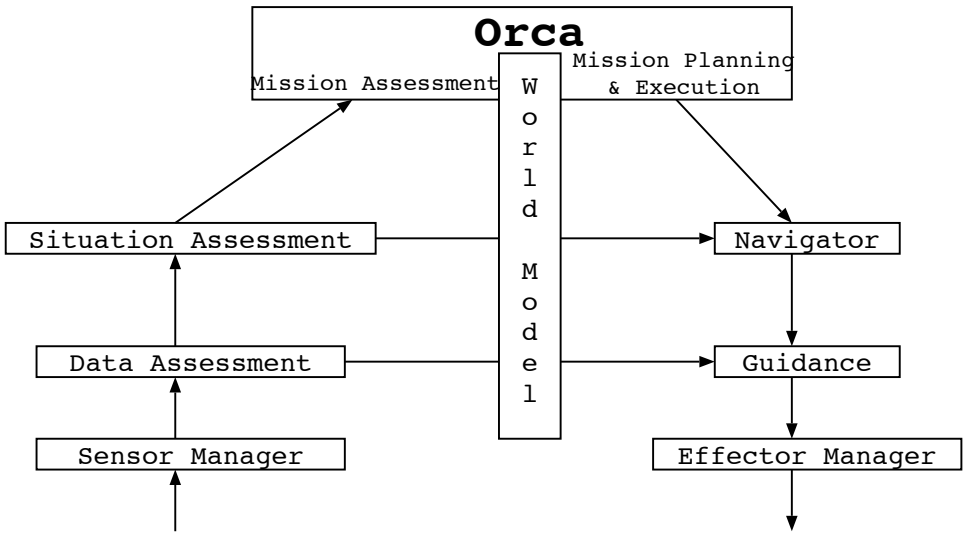

Figure 1: Orca's place in the MSEL architecture.

Input to ORCA consists of:

- from the user (and/or other AUVs): goals, constraints, environmental description (e.g., maps), and possibly partial plans; and
- from Situation Assessment: symbolic information providing an environmental-specific, mission-independent assessment of the situation.

Output from ORCA consists of:

- to the user (and/or other AUVs): information, requests (e.g., for clarification or other information), description of the mission, and status reports;
- to the Navigator: commands (e.g., "move to (x,y,z)"); and
- to both Situation Assessment and the Navigator: parameter settings and reflex presets.

To control an AUV, ORCA must be able to create plans and execute them, and it must react appropriately to unanticipated events. The question is, which technology to use to accomplish these functions?

Obviously, traditional AI planning techniques are insufficient for the reasons discussed above. Purely reactive planners, such as PENGI [Agre & Chapman, 1987], the RAP planner [Firby, 1987], and Brooks' [1986] subsumption architecture, also have disadvantages, since they have no facility for planning ahead. Rule-based expert systems have the same problem.[1]

The best approach is one that is both deliberative—i.e., able to plan ahead—and reactive. Several researchers, eschewing the purely reactive approach, have developed reactive planners that are capable of planning ahead while retaining reactivity. PRS [Georgeff & Lansky, 1987], for example, uses hierarchical plan-like structures to provide some commitment to a future course of action while remaining able to react to changes in the current situation. The few reactive planners of this type, however, have some problems with respect to controlling real physical devices, such as AUVs, in complex environments. One problem is that since the focus has usually been on the mechanisms of reactive planning *per se*, little attention has been paid to the software and hardware with which the planner must interface; yet this is crucial if the planner is to actually control that software and hardware. In addition, as pointed we have out elsewhere [Turner, 1989c], existing reactive planners do not commit to an assessment of the context they are in. Without such an assessment, each time they must make a context-sensitive decision, they must engage in reasoning about the context. A better approach is one that automatically tailors its behavior to the context: i.e., a context-sensitive reasoner.

[1]Note that this is not a problem with the rule-based methodology *per se*, just with their typical use in expert systems. Rule-based systems can be used to *implement* a reactive planner, but that is a somewhat different issue.

ORCA will be a schema-based reasoner, based in part on the MEDIC program [Turner, 1989b; Turner, 1989a; Turner, 1989c]. A schema-based reasoner (SBR) represents almost all of its knowledge as schemas, including:

- knowledge about domain objects, represented as frames [Minsky, 1975];
- procedural knowledge, represented as *procedural schemas* (p-schemas); and
- contextual knowledge—i.e., knowledge about kinds of problem-solving situations and how to behave in them—represented as *contextual schemas* (c-schemas).

Schemas

Schema-based reasoning was originally motivated in part by the common-sense observation that since patterns exist in the world and in problem solving, a problem solver acting in the world should capitalize on those patterns instead of ignoring them [Turner, 1989a]. We call the internal representation of such patterns *schemas*.

Schemas have been found in human problem solving by many researchers. For example, prototypes [Posner & Keele, 1968], scripts [Schank & Abelson, 1977], and motor schemas [Schmidt, 1975] are all schema-like structures that have been suggested to exist in humans. Chi *et al.* [1982] have proposed that structures somewhat resembling our c-schemas are used in the organization of experts' knowledge. Schemas seem to be useful for helping humans make sense of the world by organizing knowledge in a form corresponding to patterns occurring in the world. Schemas also seem to function to hold stereotypical patterns of behavior (e.g., motor schemas and scripts) so that the patterns do not have to be "recomputed" each time they are needed.

Schemas have been shown to be useful in AI as well. Frames have found widespread use in AI as a means of sensibly organizing a reasoner's knowledge about the world, and planning programs have often recorded their plans in a schematic form for later use; this includes both plans they are given [Wilkins, 1984; Alterman, 1988] and plans they derived [Fikes *et al.*, 1972]. One benefit of schematic organization of planning knowledge is that plans, once devised, do not have to be created again when a similar situation arises: the old plan can be retrieved and used again.

Procedural schemas are plan-like structures similar to the hierarchical plans used by other AI planners [e.g., Wilkins, 1984; Georgeff & Lansky, 1987]. ORCA achieves its goals by selecting and applying p-schemas. Each p-schema is a frame-like structure that has slots describing the goal(s) the p-schema can be used to achieve, features of the situations in which the p-schema is useful (similar to the filter mechanism in STRIPS-like operators), preconditions that must be satisfied before the p-schema can be applied, and expected results of applying the p-schema. In addition, a p-schema contains a partially-ordered set of steps to carry out to achieve its goals. Figure 2 shows a p-schema used to take a photograph of an underwater site in cooperation with another AUV.

```
P-takepicture:
    parameters: ?auv, ?obj, ?x, ?y, ?z
    preconditions: (operational ?auv)
    start: com-move
    steps:
        com-move: (communicate ?auv (move ?auv (+ ?x ?delta)
                        (+ ?y ?delta) ?z))
        moveself: (p-move ?self (- ?x ?delta) (- ?y ?delta) ?z)
        com-shine-light: (communicate ?auv (shine-light ?auv ?x ?y ?z))
        takepic: (photograph ?self ?x ?y ?z)
    order: (unordered: com-move, moveself), com-shine-light, takepic
    result: (havepicture ?self ?x ?y ?z)
```

Figure 2: A procedural schema.

Each step in a p-schema is either a primitive action, a goal, or another p-schema (subschema).[2] These different kinds of steps allow the reasoner to have varying levels of commitment to the actions it will carry out when applying the p-schema. If all the steps are primitive, then the p-schema will be very stereotypical and script-like. If they are all goals, then there will be little commitment to the steps until run-time. And if they are p-schemas, then there will be a level of commitment in between. There is a tradeoff between efficiency and flexibility, with increased commitment tending toward increased efficiency and decreased flexibility; p-schemas allow whatever degree of commitment that makes sense for the particular situation.

A contextual schema (c-schema) represents a kind of problem-solving situation. For example, the c-schema shown in Figure 3 represents the situation of "being in a harbor". C-schemas contain a description of the features of the situations they represent. The features can be used to recognize the AUV's current situation and to make predictions about as yet unseen aspects of the situation. In addition, c-schemas contain prescriptive information; that is, information about how to behave in the context represented by a c-schema. There are several parts of this information: "standing orders"; event-handling information; attention-focusing information; and information to help select actions.

Context description:
descriptor: harbor
water column depth: shallow
surface characteristics: surface traffic present
⋮
Standing orders:
decrease top of depth envelope
Event information:
severe leak ∨ ... ∨ power failure ⇒ catastrophic failure
catastrophic failure ⇒ land and release buoy
⋮
Attention focusing information:
don't pursue survival goals at expense of damage to other vessels
⋮
P-schema suggestions:
goal to surface ⇒ p-schema to check for surface traffic first
goal to determine position ⇒ p-schema to take GPS fix carefully
⋮

Figure 3: A contextual schema representing being in a harbor.

We will first concentrate on the basic process of p-schema application, then examine the role of c-schemas in this process.

Schema-based Reasoning Process

ORCA's overall process is shown in Figure 4. Ignoring unanticipated events for the moment, the basic process is as follows. A user first gives the reasoner a description of the mission to carry out. For example, the description of a mission to gather data at several locations might contain:

- goals, such as: sample the water column with instrument I_1 at location (x_1, y_1, z_1), sample with instrument I_2 at (x_2, y_2, z_2), etc.;
- constraints, such as: accomplish the mission in 2 days time, remain below the surface as much as possible, etc.;

[2] See [Turner, 1989a] for a discussion of other kinds of steps used in MEDIC, e.g., loops and conditionals.

- a description of the work site(s), such as a map of the area and information about the area, e.g., there is ice over subarea S_1 and subarea S_2 is a harbor.

In addition, some partial plan may also be given by the user (e.g., accomplish goal G_1 before goal G_2) or particular goals or actions may be anchored in time (e.g., achieve G_2 between times t_1 and t_2).[3]

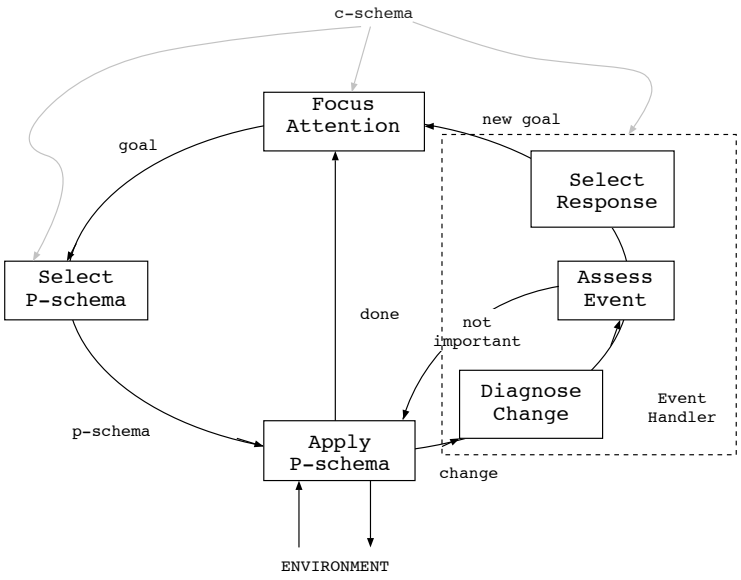

Figure 4: The schema-based reasoning process.

Once the mission is understood by ORCA, it begins trying to accomplish it. Goals, along with dependencies between them (e.g., relative time to accomplish each) are put on a *task list*.[4]

ORCA next selects one of the active tasks to achieve; this process is called *focusing attention*. Selecting an appropriate task to work on depends on assessing the relative priority of the tasks currently on the task list. The priority of a task, as discussed below, depends on features of the current situation. Once the relative priorities are determined, the task with the highest priority that can be worked on is selected as the current focus of attention.

Once a task is selected as the current focus of attention, ORCA tries to accomplish that task. The mechanism by which this happens, as discussed in more detail below, is that of *p-schema application*. A procedural schema is retrieved from ORCA's schema memory based on the task to be accomplished and relevant features of the current situation.

P-schemas are hierarchical plan-like structures, and their application can be a rather complex process; however, the ultimate result is the execution of primitive actions called *executable acts*, or xacts, such as "move", "communicate", "turn on light", etc. Executable acts are accomplished by executing a piece of (Lisp) code associated with the act; this may result in an internal action (e.g., an inference), a command to the Navigator (e.g., to move to a location), or a communication with another agent (e.g., sending information to another AUV, a human colleague, or user). The steps of p-schemas are ordered, either partially or fully, and can have a branching or looping structure, as described elsewhere [Turner, 1989a].

Once a p-schema is finished, then provided the task has been accomplished, the task is removed from the active task list. The cycle then repeats starting with attention focusing. When there are no more tasks on

[3]For the purposes of this paper, we ignore actions Orca may need to take to query the user about aspects of the mission, etc.

[4]We will in the future explore the difficult question of when to treat goals as separate and when to explicitly treat them as conjunctive.

the task list, the AUV is done with its mission.[5]

Goal achievement rarely proceeds so smoothly, of course. Often there will be failures and unanticipated events due to uncertainty, incomplete knowledge, and a changing world. When an unanticipated event (which might be a failure) occurs, ORCA does three things:

1. It attempts to determine the cause of the event, if the event itself is not a "treatable" cause—that is, it attempts to diagnose the symptoms represented by the event and other known information to determine the underlying problem.
2. It assesses the importance of the event (or its root cause) to determine if it should respond.
3. If the event is important enough, then it selects a response to the event. The response will usually be a goal to achieve (e.g., "move up in the water column") or a script to run (e.g., "surface and signal for help").

Event handling proceeds in parallel, to some extent, with p-schema application, unlike in MEDIC. This is due to the nature of the domain: actions (e.g., moving to a particular location) often cannot be easily interrupted without adversely impacting the mission. Consequently, p-schema application is not interrupted unless the event is important enough to warrant it. It is also due to the nature of the AUV, which can do several things at once.

Events can come from a variety of sources. The simplest case is when the lower-level architecture (i.e., Situation Assessment) detects an event and notifies ORCA: for example, a leak may be sensed directly from a leak detector. ORCA may also infer events from information provided by the lower level; for example, the current power level combined with knowledge of the remaining mission may lead to the conclusion that the mission is in jeopardy. Failing actions should also be treated as events to be handled, as should resource contention between two or more actions that are occurring simultaneously.

When an event is important enough to respond to and the response is a goal to achieve or p-schema or script to apply, then the goal, p-schema, or script is added to ORCA's task list. At this point, attention is focused on the most important task to work on. If the current task remains in focus, then there is no need to interrupt the p-schema applier. However, if the new task is the focus, or if the event has sufficiently changed the world that a different task becomes the focus, then the p-schema applier is interrupted and the current task's state is saved; the applier then begins (or resumes) work on the task in focus. Attention focusing can also be triggered by events that do not generate responses if ORCA believes that the world has been changed enough to warrant it; determining this is a topic of future research.

Using Contextual Schemas

Contextual schemas play an important role in several of ORCA's functions. Attention focusing, event handling, and action selection are all context-dependent. Consequently, c-schemas are used to condition all of these functions.

ORCA's *context manager* constantly monitors the state of the world (via changes to ORCA's working memory). As changes occur, it retrieves c-schemas from the schema memory that share features with the current situation. For example, if the water is shallow, the bottom is cluttered with debris, and there is a high volume of surface traffic, then the context manager might retrieve a c-schema representing the context of being in a harbor (see Figure 3). This c-schema would then represent ORCA's commitment to which context it is in: that is, in this case ORCA could be said to believe it was in a harbor with some degree of certainty.

Several c-schemas might be retrieved; for example, the current situation might resemble being in a harbor, operating with low power, performing a search mission, and working with other AUVs. When this happens, the context manager merges knowledge from these c-schemas as needed to provide ORCA with a coherent picture of its current context[6]—in this example, performing a cooperative search mission in a harbor while suffering low power. This has the advantage of reducing the number of distinct c-schemas that must be stored, as many rarely-occurring ones can be constructed on the fly by merging other c-schemas.

[5] Actually, a default, or "hotel", task should always be present on the task list to set up new missions and handle requests and information from other agents.

[6] How this merger proceeds is a topic of future research.

Contextual schemas can provide predictions of other, as yet unseen features of the environment. For example, the c-schema in Figure 3 predicts that the water will be turbid and that there are likely to be anchor lines trailing down to the bottom. In addition to predictive knowledge, however, c-schemas also contain prescriptive knowledge useful for problem solving.

The context manager uses its contextual knowledge to impact several aspects of ORCA's behavior. One area affected is attention focusing. As discussed elsewhere [Turner, 1989c; Turner, 1989a], the process of focusing attention is context-dependent. For example, an AUV may have the goal of handling the event of power becoming low: in the context of a routine survey mission, it is reasonable to focus attention on this goal, perhaps resulting in the AUV aborting its mission and returning to base; however, in the context of a rescue mission, this goal might be inappropriate to focus attention on. Since focusing attention is context-dependent, ORCA makes use of information from one or more contextual schemas to decide which task to work on at a particular time.

The process of finding a p-schema to use to achieve a goal is also context-dependent. Some ways of achieving a goal are more appropriate in some contexts than in others. For example, an AUV may have the goal of recovering from a power failure: in the context of there being a power source (e.g., a refueling station) close at hand, it makes sense to dock with it and recharge; otherwise, it may make more sense to plot a course back to the base or support ship and attempt to return home. When the p-schema applier needs to find a p-schema to achieve a goal, the context manager can often provide some suggestions of p-schemas (or classes of p-schemas) that are useful for that goal in the current context.

Event handling is also context-dependent. The context-manager can provide contextual information to help diagnose an event, assess its importance in the context, and select an appropriate response for it. Note that the event-handling information present in a contextual schema represents some prediction that the event will occur in that context; hence, the event is not, precisely speaking, unanticipated. However, we use "unanticipated" synonymously with "unplanned-for". For example, when driving a car, a person is aware of the possibility that he or she may have to stop to avoid hitting a pedestrian, but he or she would not plan for such a reaction ahead of time, since it is unlikely to occur predictably. The same argument can be made for unanticipated events in the AUV domain.

Contextual schemas also contain information we refer to as *standing orders*. This information consists of directives about what to do when in a situation described by the c-schema. In this way, a c-schema can suggest goals or tasks that should be achieved. It can also condition the lower levels of the software hierarchy to behave appropriately in the context. It does this by setting parameter values (e.g., depth and speed envelopes) at the lower levels as well as by controlling, to some extent, the way lower levels respond to events detected at their level. An example of the usefulness of the latter is turning off the obstacle avoidance "reflex" when in the context of trying to dock with the support ship.

Current Implementation: Orca 1.0

The ORCA project is proceeding in a phased fashion. The current phases are being worked on in the context of a simulator for the MAVIS (Multiple Autonomous Vehicle Imaging System) project [Turner *et al.*, 1991]. The MAVIS project is being undertaken as a cooperative venture between the University of Hawaii and the Cooperative Distributed Problem Solving (CDPS) research group at UNH. MAVIS concentrates on a two-vehicle CDPS system for taking underwater photographs: one AUV has a camera, one has a light. A copy of ORCA will be in control of each vehicle.

The first version of ORCA, called ORCA 0, was built using a version of the NOAH planner [Sacerdoti, 1977] as the plan generator. Obviously, NOAH is not reactive, and consequently much work was done to make ORCA 0 reactive. This route was taken to quickly get a mission controller built while the schema-based version of ORCA was being designed.

At the time of this writing, the current version of ORCA, called ORCA 1, is being built according to the description given in the previous section. Its event handler and attention focuser are currently just simple rule-based systems.[7] Its working memory is a data storage area with an associated resolution theorem

[7]Built using the simple TMYCIN [Novak, Jr., unpublished] rule-based system tool.

proving system to infer information not explicitly stored in the memory. Its p-schema applier is a finite state machine (FSM) [Hennie, 1968] designed to select and execute p-schemas for the goal in focus. The current implementation may only perform one step at a time and is unable to process loops and conditionals in p-schemas.

In the near future, the use of contextual schemas will be added to ORCA 1. This will entail implementing a context manager to determine which context the AUV is currently in. The event handler and attention focuser will make use of this contextual knowledge in performing their functions. Since they are rule-based systems, c-schemas will contain packets of rules useful for attention focusing and event handling in situations they represent. These rules will be used preferentially over default rules by the rule-based systems; this will involve modifying their conflict resolution mechanisms. In addition, the context manager will handle standing orders by setting parameters and reflex presets in lower levels of the architecture.

When complete, ORCA 1 will be a fully-functional schema-based reasoner, capable of controlling simulated AUVs in the MAVIS simulator. Some or all of this version of ORCA should be usable by our EAVEs and long-range AUVs.

Future Work

There are several shortcomings of the current version of ORCA that will be corrected in future versions. The development of as complex a program as an AUV controller requires the efforts of many people. One problem with ORCA 1 is that its non-uniform module interfaces and design hinders cooperative program development. This is also a problem for its role as a testbed, since one of the purposes of developing a mission controller is to have a platform for developing and testing new planning technologies. Another problem is that the program is not as interruptible as might be desired: the current version of ORCA is only interruptible when the p-schema applier is idle, between executable actions, or waiting. Finally, from an implementational standpoint, the current implementation makes use of several processes running concurrently in Lisp, and it requires a great deal of processing power and memory. Given the limitations put on processing and memory capabilities by realizable AUV hardware, it would be advantageous to explore alternative, more efficient implementations before porting ORCA to our AUVs.

To address these shortcomings, we have begun designing the third phase of the ORCA project, called ORCA 2. ORCA 2 will be a schema-based reasoner implemented as a blackboard system [Nii, 1989]. Blackboard architectures have many attractive features. There is a uniform mechanism for information interchange between pieces of the program (i.e., one or more blackboards or blackboard spaces), and there is a uniform mechanism for implementing and executing the pieces (i.e., the pieces would each be implemented as one or more knowledge sources, or KS's). This promotes both the development of ORCA in a group project as well as the use of ORCA as a testbed system. If a module, such as the p-schema applier, is implemented as a group of KS's (e.g., a KS for p-schema selection, one for expansion, etc.) rather than one monolithic unit, then that module can be interrupted between any two KS invocations, thus increasing ORCA's interruptability. Finally, there exist blackboard shells for writing blackboard systems. This not only frees up time on the part of ORCA's developers, shells tend to be more efficient and occupy less memory than custom-coded software written by developers. Consequently, the system is more likely to fit the hardware and software constraints of real AUVs.

Conclusion

In this paper, we have argued that an AUV controller needs both deliberative and reactive capabilites to handle missions of any real complexity in a world that is unpredictable and constantly changing. Schema-based reasoning meets these needs. By using schemas to represent its knowledge, a reasoner makes commitments to future expectations; in our approach, the reasoner also remains ready to react to changes in its world as needed. Context sensitivity is assured by the use of contextual schemas to represent kinds of problem solving situations and how to behave when in them.

Schema-based reasoning has been used previously with good results in a mildly reactive domain. We are

now beginning to test these ideas in the much more demanding and realistic domain of AUV control. When complete, ORCA will provide a reactive, context-sensitive controller for MSEL's EAVE and long-range ocean science vehicles, both when they are acting alone and when they are involved in multi-AUV missions.

References

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

Alterman, R. (1988). Adaptive planning. *Cognitive Science*, 12(3):393–421.

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

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

Chi, M. T. H., Glaser, R., & Rees, E. (1982). Expertise in problem solving. In Sternberg, R. J., editor, *Advances in the Psychology of Human Intelligence*, volume 1, pages 7–75. Lawrence Erlbaum Associates, Hillsdale, NJ.

Fikes, R. E., Hart, P. E., & Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288.

Firby, R. J. (1987). An investigation into reactive planning in complex domains. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 202–206.

Georgeff, M. P. & Lansky, A. L. (1987). 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.

Hennie, F. (1968). *Finite-State Models for Logical Machines*. John Wiley & Sons, Inc. New York.

Minsky, M. L. (1975). A framework for representing knowledge. In Winston, P. H., editor, *The Psychology of Computer Vision*, pages 211–277. McGraw–Hill, New York.

Nii, H. P. (1989). Blackboard systems, parts a and b. In Barr, A., Cohen, P. R., & Feigenbaum, E. A., editors, *The Handbook of Artificial Intelligence*, volume IV, chapter XVI. Addison–Wesley Publishing Company, Reading, MA.

Novak, Jr., G. S. (unpublished). TMYCIN expert system tool.

Posner, M. I. & Keele, S. W. (1968). On the genesis of abstract ideas. *Journal of Experimental Psychology*, 83:304–308.

Sacerdoti, E. D. (1977). *A Structure for Plans and Behavior*. Elsevier, Amsterdam.

Schank, R. C. & Abelson, R. (1977). *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Schmidt, R. A. (1975). A schema theory of discrete motor skill learning. *Psychological Review*, 82(4):225–259.

Turner, R. M. (1989a). *A Schema-based Model of Adaptive Problem Solving*. PhD thesis, School of Information and Computer Science, Georgia Institute of Technology. Technical report GIT–ICS–89/42.

Turner, R. M. (1989b). Using schemas for diagnosis. *Computer Methods and Programs in Biomedicine*, 30(2/3):199–208.

Turner, R. M. (1989c). 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.

Turner, R. M., Fox, J. S., Turner, E. H., & Blidberg, D. R. (1991). 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.

Wilkins, D. E. (1984). Domain-independent planning: Representation and plan generation. *Artificial Intelligence*, 22(3).