Handling Unanticipated Events During Collaboration

Roy M. Turner
Department of Computer Science
University of New Hampshire
Durham, New Hampshire 03824
rmt@unh.edu

Peggy S. Eaton
Department of Computer Science
University of New Hampshire
Durham, New Hampshire 03824
pse@cs.unh.edu

Abstract

Handling unanticipated events during problem solving is difficult enough when an agent is operating by itself. When the agent is part of a cooperative distributed problem solving (CDPS) system, the task's difficulty increases dramatically. Now the agent is forced to consider the effect of the event not only on itself, but also on others and the group as a whole. It must also consider who should handle the event and the likely impact that actions taken to diagnose the event or respond to it may have on other agents. In this paper, we discuss preliminary work aimed at developing a process for handling events during multiagent cooperative problem solving. The domain in which the work is being done is cooperating multiple autonomous underwater vehicles (AUVs). However, the approach should have broader applicability to almost any real-world cooperative problem solving task involving autonomous or nearly autonomous agents.

Introduction

Autonomous agents, whether humans or computer systems, must cope with the fact that the world is unpredictable. This has a variety of causes, most of which can be traced to uncertainty or incomplete information about the environment, unknown processes operating in the world, other agents' actions, or the actions of the agent itself. Unpredictability manifests itself in the form of *unanticipated events* that occur during problem solving: unexpected states or action outcomes, including failures and unanticipated interactions with others. Unless an agent can recognize and handle these events, it will be incapable of prolonged useful activity in the real world. Its plans will fail beyond recovery, and it will fail to seize unexpected opportunities.

Handling unanticipated events is difficult. Meaningful events are often difficult to pick out from the background of changes taking place in the world and the agent; even when detected, it is often hard to adequately diagnose the event's cause and select an appropriate response for the current problem-solving situation. The problem is compounded when an agent is part of a larger problem-solving ensemble, as in the case of a group of agents participating in cooperative distributed problem solving (CDPS). Here, agents must disentangle those events affecting only themselves from those affecting the broader group. In addition, at all points during the process of handling an event, an agent must keep in mind its role in the overall group and the fact that it may need to coordinate its information or responses with other members.

In this paper, we discuss event handling during multiagent cooperative problem solving. Our domain is cooperating autonomous underwater vehicles (AUVs) (Turner *et al.*, 1991); however, much of what we discuss should be applicable to any group of autonomous agents collaborating to solve problems in the real world. The work reported is largely a part of the ORCA project (Turner, 1994; Turner & Stevenson, 1991), whose goal is the creation of a robust, intelligent controller for AUVs for use in ocean science and CDPS systems.

Unanticipated Events and Multiagent Systems

Elsewhere, we have defined unanticipated events as all unpredictable changes to the agent's problem-solving situation (Turner, 1994). We do not necessarily mean only those things that are novel or completely unexpected, however (though we do not rule those out). Instead, we mean "unpredictable" in the sense that a person stepping in front of a car is unpredictable: although it is likely that, if asked, the driver would have been able to predict that at *some* time *some* person could step in front of his or her car at *some* place, the exact combination of time and person and place was utterly unpredictable in advance.

Unanticipated events in the context of multiple agents force us to consider not only events that impact a single agent's problem-solving activities, but also those that impact one or more other members of the group or the group as a whole. This is hampered by the essentially local view that any particular agent has of its world. Agents build their view of the world based on input from sensors, their understanding of the effects of actions in the world, knowledge of the way agents interact (including the overall organizational structure and agents' roles in it), and knowledge about what other agents may believe concerning a particular situation. Unfortunately, this local view is inherently incomplete and uncertain.[1] In addition, the agent cannot ignore the effects of its actions and changes in its beliefs on other agents with which it cooperating. These two things, a local world view and interdependence of agents' actions and beliefs, impact all phases of event handling.

Handling Events in Multiagent Systems

Our overall process for handling events during collaborative problem solving is shown in Figure 1. In many ways, it is similar to the process described by Turner (1994) for event handling when there is only a single agent involved: event detection, followed by diagnosis, importance assessment, and

[1] Perhaps this is just as well, as bounds on the agent's processing abilities would most likely preclude using the knowledge even if it *were* complete and certain.

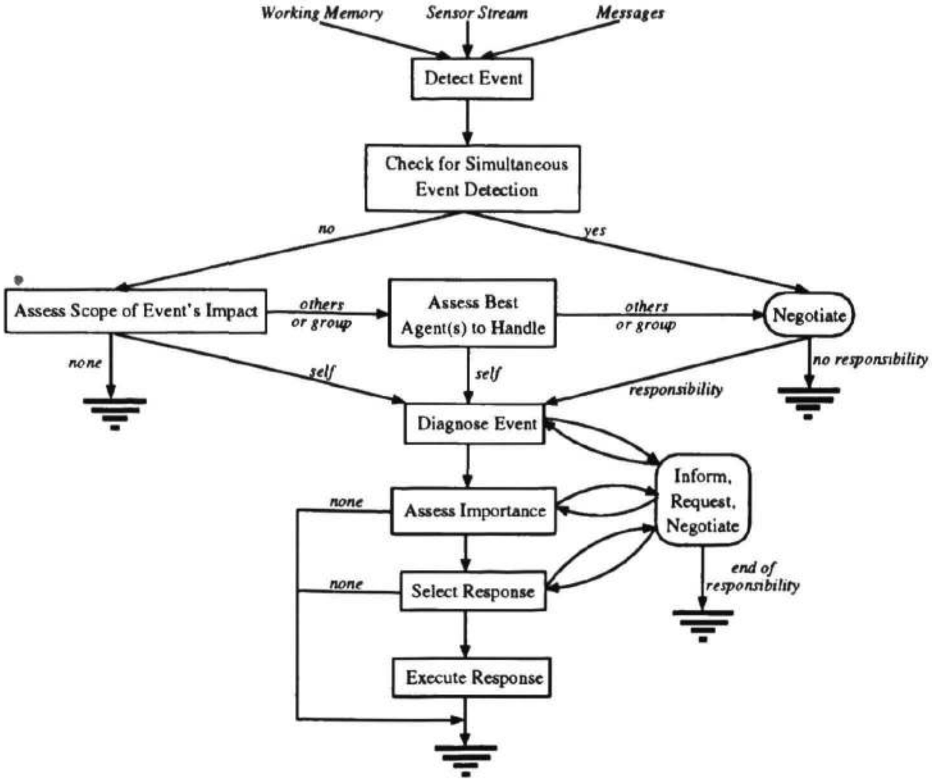

Figure 1: Process of handling events in a multiagent system.

response selection. There are, however, some notable differences, as we discuss below.

Event detection. Detecting an event fundamentally relies on comparing the agent's *sensor stream*—that is, the information coming from its sensors (possibly after being processed by other software, in the case of software agents)—with what the agent believes about the world, other agents, and the likely results of its own actions. Messages from others can also serve as a source of information to use in event detection; for example, another AUV might tell an agent "You have run aground."

The knowledge necessary for event detection includes a good model of the environment around the agent, so that differences between what the sensors report and what the agent believes can be meaningful. An agent also needs to know the organization of the group of which it is a part, including its own and others' responsibilities and commitments to actions (i.e., their *intentions* (e.g., Durfee & Lesser, 1987; Georgeff & Ingrand, 1988)). This allows it to predict the likely impact of features of the environment (or itself, or others) that it detects on its and others' actions and plans. For event notification messages, the agent needs to know how reliable the other agent is, not only with respect to the sender's processing and sensor abilities, but also its beliefs and knowledge about the agent and the group. The agent also needs to maintain a history of past sensor information, beliefs, and so forth, so that trends and intermittent events can be detected (e.g., power slowly failing).

Checking for simultaneous event detection. After an event is detected by an agent, it must decide if others are likely to have also detected (or will soon detect) the event. It would not do to have several agents simultaneously handling a single event, unaware of others' actions. Not only are "race conditions" likely, but the duplicated work would probably be unproductive or even counter-productive.

To do this, an agent must have a good model of the other agent's activities, position, sensor capabilities, and of *their* internal models of the world. The former three are needed so that the agent can determine whether the others are at vantage points to witness the event and whether they have the necessary sensory and attentional resources to detect the event when it occurred. The latter is needed so that the agent can predict whether the event looked like an event to the other agents. For example, suppose during a cooperative underwater photography task (e.g., the MAVIS task (Turner *et al.*, 1991)), AUV A notices that AUV B's light turned off when, from A's standpoint, it should not have. AUV A should predict that B likely (but not necessarily) also detected the event; it should also predict that another of the group's AUVs, C, that is not involved in the current task, would likely not have noticed—to it, seeing B's light go off would not mean that something unexpected had happened. Consequently, A needs only be concerned with B having simultaneously noticed the event.

If an agent believes that others have also noticed an event, then it needs to enter into negotiation (see, e.g., Sycara, 1989) to decide which of them will handle the event. The possibilities are: the agent, the agent and others (possibly including the one being negotiated with), or others and not the agent. If the agent has no responsibility for the event, then its processing of it can terminate.[2]

Assessing the scope of the event. When an agent decides that no other agents are likely to have noticed an event, then it has *de facto* responsibility for it. Part of this responsibility entails deciding whether or not to notify others and/or to get help. This relies on an assessment of the scope of the event: Does the event impact only me, or does it impact others in the group (or the group as a whole)?

If the agent determines that the event impacts only itself, then it can proceed with handling it. If it decides that the event affects no one, then it can terminate processing the event. (For example, if an agent notices a rockslide, but neither it nor any other agent is predicted to go near the rubble, then it may be safe to ignore the event.) If the event is predicted to impact others, then the agent must decide who is best able to handle the event.

The knowledge necessary to support assessment of an event's scope again relies on environmental knowledge as well as a model of the group's organization and models of others' actions, plans, and beliefs.

Determining the best agent to handle the event. Determining what agent can best handle an event can be done in a number of ways, the one selected depending both on the organization of the CDPS group as well as on the agent's knowledge. If the group's organization is hierarchical, for example, then it may be a standing policy that an agent noticing an event must notify its superior, which will then (possibly after notifying *its* superior, etc.) select the agent(s) that will handle the event. In other situations, for example, when there is a "flat" agent organization (e.g., in PGP systems (Durfee & Lesser, 1987)), agents may negotiate among themselves as to who should handle the event.

The agent's own knowledge impacts the determination as well. If it has sufficient knowledge, it may be able to predict which of its fellow agents is in the best position, from the standpoint of both knowledge and ability, to handle the event. To do this, it needs knowledge of other agents' abilities, intentions, knowledge, and current workload. The latter is needed because an agent may be in the best position to handle an event, yet be too busy.

Diagnosing the event. Event diagnosis is a difficult task. Among other things, it is difficult to know when to stop: Should the event be diagnosed to the ultimate cause or just to the level at which a response can be made to it? Elsewhere, we have argued that the latter is most appropriate (Turner, 1994); Rubin (1975) made a similar argument for the diagnosis of the surface form of medical "events" (i.e., signs and symptoms).

It may be that an agent in a multiagent system needs to contact other agents to perform event diagnosis. One reason for this may be to obtain information that it does not have. For example, an AUV may notice that it is drifting off course; one reason for this may be that it is in a current. If another AUV has recently traveled through the same region, it can ask it if it, too, experienced difficulty maintaining its heading—if so, then it's likely that there is a current present.

Event diagnosis may also involve taking actions; this is true, for example, in medical diagnosis (e.g., perform a lab test) as well in the AUV domain (e.g., attempt to back up to see if the vehicle is trapped in a net). An agent may need to ask others to take actions on its behalf to help it diagnose an event it has noticed. For example, an AUV may ask another to visually inspect its thrusters to look for damage after it notices that its motion has stopped. When such coordination is necessary, the agent faces the usual quandary of how much to tell its partner(s) about why it has requested the actions: Should it simply ask for an action to be performed, or should it give a rationale for the action, so the other agent can use its own knowledge and initiative to better help it?

Even if the agent needs no help from others, if it must take actions to diagnose an event, it must first assess the actions' impact on others. It may, depending on the actions to be taken, need to inform or even negotiate with others.

The knowledge needed for diagnosis in a sense goes somewhat beyond the knowledge of others and their actions, etc., necessary for the other phases of event handling. Event diagnosis also requires the kinds of associational/causal or model-based knowledge needed for other diagnostic tasks.

Assessing the event's importance. Once an event has been diagnosed to a treatable cause, then the agent needs to determine if it is important enough to respond to.[3] As we discuss below and elsewhere (Turner, 1993; Turner, 1994), an event's importance depends to a very large extent on the current problem-solving context; whether or not a response is selected depends not only on the event's importance, but how reactive the reasoner chooses to be in the current situation. Both of these factors can depend on other agents or the overall problem-solving group. For example, the failure of an AUV's light may have little importance for the AUV or its plans; however, if it knows that a camera-bearing AUV is likely to ask it to illuminate targets at some point during the mission, the event becomes more important.

If an agent predicts that an event may have importance to the broader problem-solving system, it may need to inform others of the event, ask for information to help with the assessment, or even negotiate with other agents to settle on a mutually-agreeable importance estimate. The knowledge needed for this is similar to that for the other phases of event handling.

Selecting a response. The appropriate response for an event may be different in a multiagent context than in a situation involving a single agent. For example, when encountering a current, an agent operating by itself would likely maneuver out of the current and update its internal model of the environment so it would not stray into the current in future; in a multiagent situation, however, it should also consider broadcasting information about the current to its co-workers so that they can avoid it.

[2] This is not strictly true; the agent should monitor the handling of the event so that at some later time it can help, if necessary.

[3] This phase occurs after diagnosis because importance often depends more on the cause (e.g., a disease) than the surface event (e.g., a symptom).

In a multiagent situation, an agent may also be able to ask other agents to take actions on its behalf to respond to an event. For example, an AUV that has landed on a soft bottom and has subsequently become stuck may ask another agent to try to pull it off.

Even when an agent's response does not directly involve others, it may still need to coordinate with them if the response impacts their current or future intentions. For example, if the only AUV with a light begins to lose power, it may decide to return to the support vessel; it should tell its peers, however, as they may need to use the light later in the group's mission.

In addition to needing a model of the domain and of others' actions and beliefs, the agent must be able to project the likely effects of its response on the environment and the others.

Coordination between agents. At several points during the event handling process, agents need to communicate with one another. This may involve informing agents of events or responses, asking for help or information, or negotiating to iron out inconsistencies in information or to agree on a course of action.

In many domains, including the AUV domain, communication is not as straightforward as it might seem. For example, in the AUV domain communication most often takes place by broadcasting messages over an acoustic link. Issues arise of limited communication channel bandwidth and overheard messages giving rise to unintended event-handling behavior on the part of the "eavesdroppers". In addition, communication needs to be incorporated into the broader context of group problem solving in a coherent manner. Turner and Turner (1991) describe one mechanism for this. In general, much work remains to be done.

Role of context. In all of the preceding discussion of knowledge required for multiagent event handling, much of what we have been talking about can be described as *contextual knowledge*: knowledge of the environmental, problem-solving, and social/cooperative context the agent is in. Contextual knowledge is crucial in appropriately conditioning an agent's behavior to the current problem-solving context. Contextual knowledge, however, comes not only from the agent's sensor stream and messages received. It also comes from the agent's knowledge about problem-solving situations in general, which may be the product of its past experiences.

Elsewhere, we have discussed a mechanism for context-sensitive reasoning in real-world domains (e.g., Turner, 1989; Turner, 1993; Turner, 1994), including handling unanticipated events. We believe that this mechanism, which relies on retrieving and merging *contextual schemas* into a coherent picture of the current context, can be extended to facilitate multiagent event handling as well.

Complications Arising from Time-Consuming Actions. Above, we have discussed the process of event handling as if actions, including communication actions, take negligible time. This is not the case in real world domains, and especially not in relatively slow domains such as AUV control. For example, suppose the agent must move in order to diagnose an event; this could take seconds to minutes. Similarly, communication is slow underwater; the current generation of available acoustic modems have a bit rate of about 1200 baud, and there is a significant time lag arising from the speed of sound in water.

During the time the agent is carrying out actions in support of event handling, the world does not stand still. Information continues to stream into the agent from outside, the world continues to change, and others continue their tasks. The agent cannot, then, suspend event handling while waiting for actions to complete; new events will demand attention, and should be handled.

Two possibilities suggest themselves. One solution is to make the state of event handling explicit, so that the agent can interrupt the handling of one event, then later resume where it left off after the necessary actions have been taken. A problem with this is that until the actions complete, the agent has actually done nothing about the event, which could be catastrophic: consider an event that suggests an AUV may be drifting below its crush depth. The second solution is to allow event handling to finish without waiting for actions, communication, and/or negotiation to finish—but to make only a provisional diagnosis, assessment, and action recommendation. Later, when additional information arrives as a result of the actions or communication, the agent would reevaluate the event in light of the new information. We will investigate the latter solution as this work progresses.

Multiagent Event Handling in Orca

Figure 2 shows the internal structure of the current version of ORCA. It is divided into several modules, including a Communications Module that is not shown.

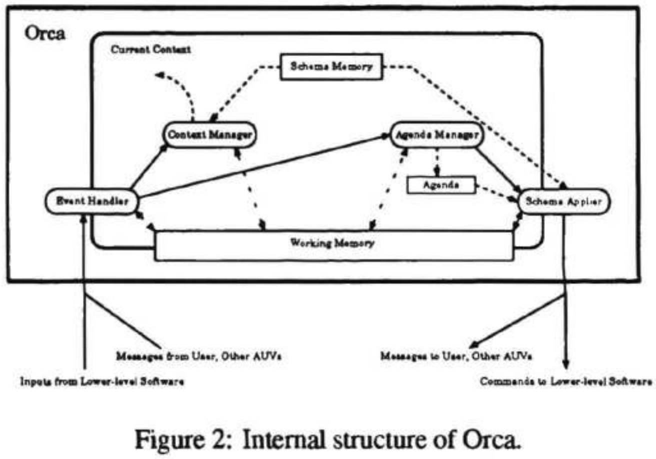

Figure 2: Internal structure of Orca.

Event Handler accepts all incoming sensor information and (parsed) messages from other agents, including the AUV's users. It is responsible for detecting if the input, together with past input, signals an event. If not, it routes the input appropriately (e.g., to the working memory, in the case of data, or Agenda Manager, in the case of goals). If so, it diagnoses the event, assesses its importance, and selects a response (a goal), which it then sends to Agenda Manager. Agenda Manager is responsible for maintaining ORCA's focus of attention, ensuring that it is always working on an appropriate set of goals for the current problem-solving situation. Schema Applier uses ORCA's library of *procedural schemas* to achieve the goals on the agenda, as described elsewhere (e.g., Turner, 1994). Context Manager watches the evolving problem-solving situation and decides what the current context is; it does this by building a *current contextual schema* from its library of contextual schemas, each of which represent one class of situations. The current contextual schema represents both a commitment to what the context is and a repository for a great deal of context-specific problem-solving knowledge the other modules use to guide ORCA's actions.

The multiagent event-handling process shown in Figure 1 does not cleanly fit into any of the modules shown; nor should it, necessarily, as the process of multiagent event handling is quite different than for a single agent. Rather, it will have to be distributed over the pieces of ORCA, with most of it residing in Event Handler. Other pieces of the process are done by Schema Applier and Agenda Manager. For example, both will be involved when actions are necessary to commuicate with other agents to carry out negotiation; both will be involved when event diagnosis requires actions to be taken that impact ORCA's other actions. All the modules will continue to rely on information supplied by Context Manager.

Related Work

Many planning systems meant for single-agent problem solving have had to deal with unanticipated events. For example, PRS (Georgeff & Ingrand, 1988; Georgeff & Lansky, 1987) and MEDIC (Turner, 1989; Turner, 1994) have both explored the problem of events occurring during the execution of a single agent's plan. PRS, for example, uses malfunction procedures and meta-knowledge to handle failures that have been anticipated and domain specific meta-knowledge to reason about multiple unrelated failures; the development of a general solution for reasoning about unexpected (unrelated) failures is left for future work. MEDIC uses contextual schemas to provide information about an unanticipated event's importance in a particular situation and about how to handle the event appropriately. ORCA (Turner, 1994; Turner & Stevenson, 1991), which is based partly on MEDIC, is extending MEDIC's event-handling mechanism to cope with real-world domains; the work reported here will ultimately be folded into ORCA.

Others have looked at the problem of handling events in multiagent situations. For example, Phoenix (Cohen *et al.*, 1989), a multiagent system operating in the firefighting domain, has agents that interleave planning and execution, are capable of reactive responses to anticipated events, and can perform deliberative planning. However, unanticipated events are not handled by individual agents, which have only a local view of the problem-solving situation, but rather by a central fire-boss with a global view of the situation. Individual agents do not reason about the effect their actions may have on others, and the system uses a set of predefined events a single agent may encounter. Work on Partial Global Planning (PGP) (e.g., Durfee & Lesser, 1987) also deals with events in a distributed AI environment. However, the agents we are interested in are more autonomous than the typical PGP agents; consequently, they have more flexibility to handle events on their own. Cammarata *et al.* (1983) were concerned with agents that are semi-autonomous (airplanes) in their air traffic control system. However, they dealt with a relatively impoverished set of event types, and relied on planning by a single agent in most cases to handle the event.

Conclusion

Handling events during multiagent cooperative problem solving is a very difficult task. Events can arise due to a single agent or a group of agents; an event can be detected by the agent that gave rise to it or by others. Deciding who should handle the event, as well as the process of diagnosing it, assessing its importance, and selecting a response may all need to be done cooperatively.

In this paper, we have sketched an approach to handling events in multiagent cooperative systems. The process relies on knowledge about the world, oneself and one's intentions, and others' beliefs and intentions, as well as on additional *a priori* contextual knowledge, possibly built from past experiences in similar situations. The process also depends on knowing when and what to communicate with others, based on what an agent knows about their current beliefs and intentions, to inform them of pertinent information, to ask for information or assistance, and to negotiate about handling the event.

The work reported here is preliminary. Though we will be developing and evaluating the approach in the AUV domain as part of the ORCA project, we believe it is applicable for handling unanticipated events in most cooperative problem-solving situations in the real world.

Acknowledgments

The authors would like to thank the National Science Foundation, which provided support for part of this work under grant BCS–9211914. We would also like to thank the rest of the UNH Cooperative Distributed Problem Solving research group for their helpful comments and suggestions.

References

Cammarata, S., McArthur, D., & Steeb, R. (1983). Strategies of cooperation in distributed problem solving. In *Proceedings of the 1983 International Joint Conference on Artificial Intelligence*, pages 767–770.

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

Durfee, E. H. & Lesser, V. R. (1987). Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the 1987 International Joint Conference on Artificial Intelligence*, pages 875–883.

Georgeff, M. P. & Ingrand, F. F. (1988). Research on procedural reasoning systems. Final Report, Phase 1, AI Center, SRI International, Menlo Park, California.

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.

Rubin, A. (1975). Hypothesis formation and evaluation in medical diagnosis. Technical Report AI–TR–316, Artificial Intelligence Laboratory, MIT, Cambridge, MA.

Sycara, K. P. (1989). Multiagent compromise via negotiation. In Gasser, L. & Huhns, M. N., editors, *Distributed Artificial Intelligence*, volume II, chapter 6, pages 119–138. Morgan Kaufmann Publishers, Inc., San Mateo, CA.

Turner, E. H. & Turner, R. M. (1991). A schema-based approach to cooperative behavior. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*.

Turner, R. M. (1989). 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. (1993). Context-sensitive reasoning for autonomous agents and cooperative distributed problem solving. In *Proceedings of the IJCAI Workshop on Using Knowledge in its Context*, Chambery, France.

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

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 Underwater Submersible Technology (AUV '91)*.

Turner, R. M. & Stevenson, R. A. G. (1991). ORCA: An adaptive, context-sensitive reasoner for controlling AUVs. In *Proceedings of the 7th International Symposium on Unmanned Untethered Underwater Submersible Technology (AUV '91)*.