Many Robots Make Short Work

Report of the SRI International mobile robot team at AAAI96

Didier Guzzoni
Swiss Federal Institute of Technology
Adam Cheyer
Luc Julia
Kurt Konolige
Artificial Intelligence Laboratory, SRI International

1. Holding a meeting

At the SRI Artificial Intelligence Lab, we have a long history of building autonomous robots, from the original Shakey (remember the STRIPS planner?) through Flakey [Congdon 1993] and more recently, the Pioneer class of small robots. Our current research focuses on realtime vision for robots, and multirobot planning using an agent-oriented architecture.

For the AAAI contest, we wanted to showcase our research, especially the ability to control multiple robots using a distributed set of software agents on the Internet. The agent technology, called the Open Agent Architecture (OAA), was developed at SRI as a way of accessing many different types of information available in computers at different locations.

In the "hold a meeting" event, a robot starts from the Director's office, determines which of two conference rooms is empty, notifies two professors where and when the meeting will be held, and then returns to tell the Director. Points are awarded for accomplishing the different parts of the task, for communicating effectively about its goals, and for finishing the task quickly. Our strategy was simple: use as many robots as we could to cut down on the time to find the rooms and notify the professors. We decided that three robots was an optimal choice: enough to search for the rooms efficiently, but not too many to get in each other's way or strain our resources. We would have two robots searching for the rooms and professors, and one remaining behind in the Director's office and tell her when the meeting would be. We were concerned that leaving one robot behind as a mobile telephone was stretching things a bit, so we cleared our strategy with the judges well before the contest.

The two search robots are Pioneer-class robots, portable robots first developed by SRI for classroom use, and now manufactured by Real World Interfaces, Inc. They run SRI's Saphira control software, which navigates the robots around an office environment, keeping track of where they are using perceptual cues from the robot sensors. Each Pioneer robot has seven sonar sensors, a Fast Track vision system from Newton Labs, and a portable computer on top with a radio ethernet for communication to a base station. The Fast Track system is an interesting device: it consists of a small color video camera and a low-power processor. It can be trained to recognize particular colors, and will indicate the position of any object with that color. We decided to use the vision system to find people in the conference rooms, and trained it to recognize red. If the judges would wear red shorts, the vision system would easily pick them out.

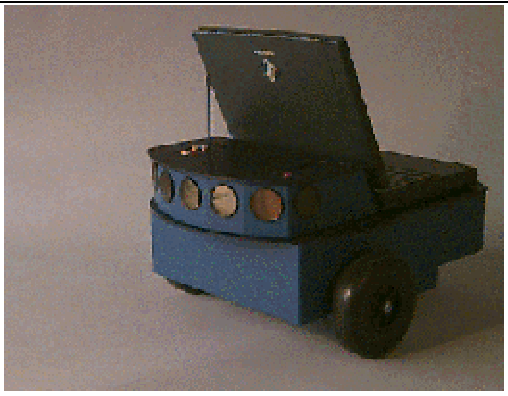


Figure 1 A Pioneer Robot with Laptop Host Computer

The robot in the Director's office didn't have to move, just relay information to the Director. We used a Koala robot, a small 6-wheeled vehicle under development at the Swiss Federal Institute of Technology. The Koala has infrared sensors, which enable it to avoid obstacles, but make it difficult to map the environment because they do not give a reliable range estimate. So we just kept the Koala stationary, with a portable PC on top to communicate with the other robots and talk to the Director.

Each robot, by virtue of the radio ethernet, is a node on the internet and an agent in the OAA. Other agents, residing on the base station, included a database agent for holding information relayed back by the robots, a mapping agent for determining and displaying where the robots are, a strategy agent for coordination, and interface agents (speech, pen gestures) for giving the robots commands. If we had a connection to the outside world, we could have run the robots from anywhere in the world!

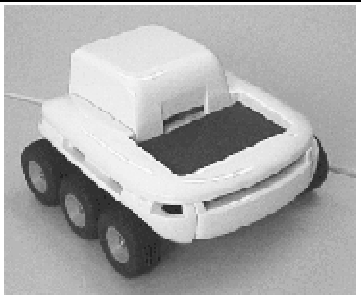


Figure 2 A Koala Robot from the Swiss Federal Institute of Technology

One of the interesting aspects of the agent architecture is that the robots are capable of a good deal of autonomy. For example, the strategy agent might tell them where to go, and even give them a path. The robots are responsible for navigation, avoiding obstacles, and finding the correct goal position. If they fail, they contact the strategy agent about the problem and wait for instructions. The connection between the robots and the rest of the agents can be very low bandwidth.

This was our strategy; now we had to execute it. We arrived at the contest late, during the preliminary rounds, and hastily set up our base station and put in the map of the office environment. All did not go smoothly --- several unanticipated coordination problems between the mapping agent and the robots caused us some frustrating moments. We hadn't realized the planner would happily plan paths through rooms with two doors, something the robots didn't like because they get lost easily in rooms and do much better in the corridors. And one of our robots was injured in a fall as we were packing up at SRI; fortunately RWI was there, and loaned us another Pioneer. And so, late in the evening of the second day, we had a perfect preliminary round run. The map of Figure 3 diagrams the paths of the two Pioneer robots. Each of them started out going to a different conference room. The green line robot arrived at the first room, found it occupied, and then started heading for a professor's office. Meanwhile, the red line robot reached the second conference room, and after a short time decided there were no people present. It then went to the second professor's office.

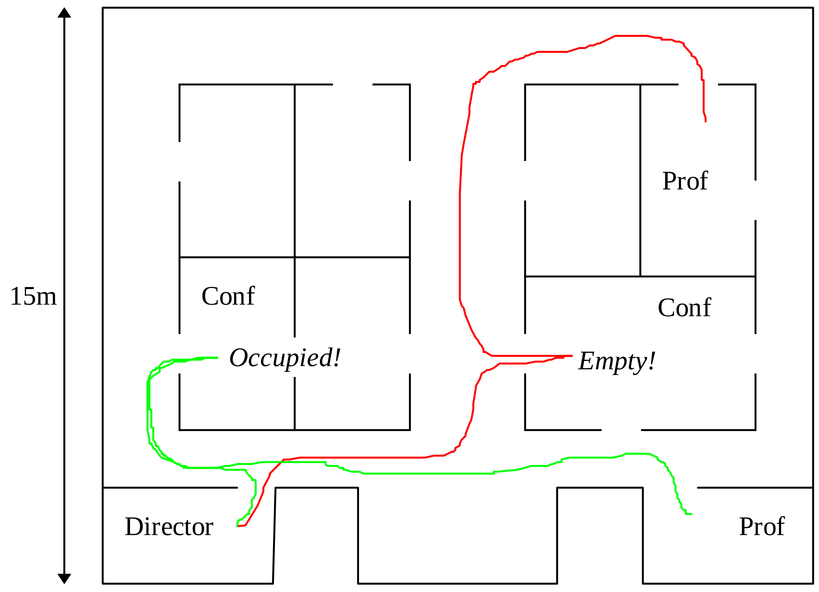


Figure 3 Paths of the two Pioneer robots.

The green line robot arrived at the professor's office just a little ahead of the red line robot. As the robots entered the professors' offices, they announced the time of the meeting and the conference room. At this point, the Koala robot announced the meeting to the Director, and the task was completed.

The next day, the finals, was almost an anticlimax, except for the large audience and the presence of Alan Alda and the camera crew from Scientific American Frontiers. We started the Pioneers out from the Director's office, and they went happily on their way. There was too much going on to keep track of both of them — they were both announcing what they were doing, the camera folks were dancing around, and we had to tell the audience what was happening. Before we knew it, both robots were in the professors' offices, announcing the meeting: only 4 minutes and 30 seconds to completion! From watching the other teams, we knew the nearest time would be close to 10 minutes. Parallelism does work sometimes...

In the rest of this article, we'll briefly describe Saphira (the robot control program), the Open Agent Architecture, and then discuss our implementation of multirobot planning and control using these tools.

2. Saphira and OAA

2.1. The Saphira Robot Controller

The Saphira architecture [Saffiotti 1995; Konolige 1996] is an integrated sensing and control system for robotics applications. At the center is the LPS (see Figure 4), a geometric representation of space around the robot. Because different tasks demand different representations, the LPS is designed to accommodate various levels of interpretation of sensor information, as well as {\it a priori} information from sources such as maps. Currently, the major representational technologies are:

- A grid-based representation similar to Moravec and Elfes' occupancy grids [Moravec 1985] built from the fusion of sensor readings.
- More analytic representations of surface features such as linear surfaces, which interpret sensor data relative to models of the environment.
- Semantic descriptions of the world, using structures such as corridors or doorways (*artifacts*). Artifacts are the product of bottom-up interpretation of sensor readings, or top-down refinement of map information.

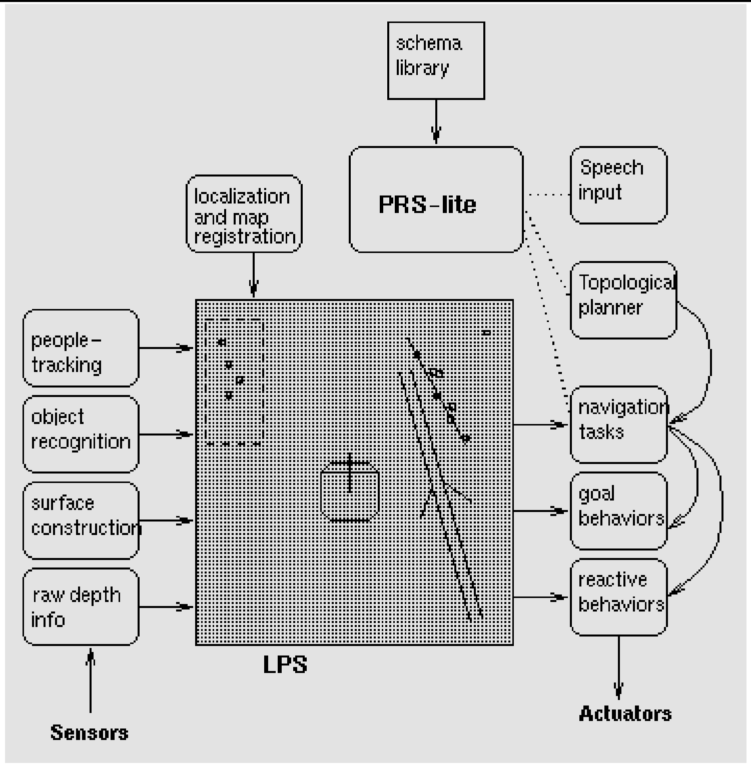


Figure 4 Saphira system architecture. Perceptual routines are on the left, action routines on the right. The vertical dimension gives an indication of the cognitive level of processing, with high-level behaviors and perceptual routines at the top. Control is coordinated by the Procedural Reasoning System (PRS-lite), which instantiates routines for task sequencing and monitoring, and perceptual coordination.

The LPS gives the robot an awareness of its immediate environment, and is critical in the tasks of fusing sensor information, planning local movement, and integrating map information. The perceptual and control architecture make constant reference to the local perceptual space. One can think of the internal artifacts as Saphira's *beliefs* about the world, and most actions are planned and executed with respect to these beliefs.

In Brooks' terms [Brooks 1986] the organization is partly vertical and partly horizontal. The vertical organization occurs in both perception (left side) and action (right side). Various perceptual routines are responsible for both adding sensor information to the LPS and processing it to produce surface information that can be used by object recognition and navigation routines. On the action side, the lowest level behaviors look mostly at occupancy information to do obstacle avoidance. The basic building blocks of behaviors are fuzzy rules, which give the robot the ability to react gracefully to the environment by grading the strength of the reaction (e.g., turn left) according to the strength of the stimulus (e.g., distance of an obstacle on the right). Navigation routines make use of map information to guide the robot towards goal locations, e.g., to a corridor junction. At the same time, registration routines keep track of sensed objects, constantly relating them to internal map objects to keep the robot accurately positioned with respect to the map. Thus, Saphira is able to accept a plan, a sequence of waypoints to a final goal, and execute it while keeping the robot localized within the global map.

2.2. The Open Agent Architecture

When planning our strategy for how to approach this year's robot contest, we decided to take advantage of our recent integration of Saphira as an agent within the Open Agent Architecture (OAA) [Cohen 1994]. The OAA is a framework for constructing multiagent systems that has been used by SRI and clients to construct more than fifteen applications in various domains [Moran 1997, Kameyama 1995]. Applying the OAA to the Office Navigation task in the robot competition could provide the following advantages:

- Distributed: agents can run on different platforms and operating systems, and can cooperate in parallel to achieve a task. Some agents could be placed locally on each robot's laptop controller, while other services could be stored on a more powerful workstation.
- Plug and play: agent communities can be formed by dynamically adding new agents at runtime. It is as easy to have multiple robots executing tasks as it is to have just one.
- Agent Services: many services and technologies encapsulated by preexisting agents can easily be added as resources provided by our agent community. Useful agents for the robot domain would include database agents, mapping agents, agents for text-to-speech, speech recognition, natural language, all which are directly reusable from other agent-based applications.
- Multimodal: the agent architecture has been designed with the human user in mind [Cheyer 1995, Moran 1995]. Agents have been developed to allow people to naturally combine, drawing, speaking, writing with more standard graphical user interface approaches when addressing the set of distributed agents. In a robot domain, we can monitor progress of robots on a map, and if required, give them instructions by speaking "You are here facing this direction" (while drawing an arrow), or "Pick up this object" while indicating a target using circles, pointing or arrow gestures.
- Mobile: the agent libraries are lightweight enough to allow multiple agents to run on small, wireless PDAs or laptops, and communication among agents is fast enough to provide realtime response for the robot domain.

The OAA uses a distributed architecture in which a Facilitator agent is responsible for scheduling and maintaining the flow of communication among a number of client agents. Agents interact with each other through an Interagent Communication Language (ICL), a logic-based declarative language based on an extension of Prolog. The primary job of the Facilitator is to decompose ICL expressions and route them to agents who have indicated a capability of resolving them. As communication occurs in an undirected fashion, with agents specifying what information they need, not how this information is to be obtained, agents can be replaced or added in a "plug-and-play" fashion.

Each agent in the OAA consists of a wrapper encapsulating a knowledge layer written in Prolog, C, Lisp, Java, Visual Basic or Borland's Delphi. The knowledge layer, in turn, may lie on top of existing standalone applications, and serves to map the functionality of the underlying application into the ICL. In the case of the physical robots, we installed an agent interface on top of Saphira, so that information about the robot’s location, and commands to navigate the robot, were made available to all agents.

The OAA agent library provides common functionality across all agents. Each agent can respond to or produce requests for information or service, and can install triggers to monitor real-world conditions. Triggers may make reference to temporal events, to changes in local or remote data values, to specific agent communication messages or to domain-specific test conditions provided by some agent (e.g., a trigger request "When mail arrives from Bob..." will automatically be installed by the Facilitator on the mail agent, who can perform this verification).

3. Robots as Physical Agents

3.1. System Design

The system we developed is made of a set of independent agents (including robots), able to communicate in order to perform cooperative tasks. An operator can graphically monitor the whole scene and interactively control the robots. A top level program, the *strategy agent*, was designed to synchronize and control the robots and software agents.

Figure 5 is a diagram of the complete system, including the physical location of all agents. The facilitator, database agent, map manager agent, strategy agent and speech recognition agent were running on a UNIX workstation (Sparc20). On the robots, each Saphira agent was running (under Windows 95) on a laptop computer, each equipped with sound devices and text-to-speech converters. The link between the

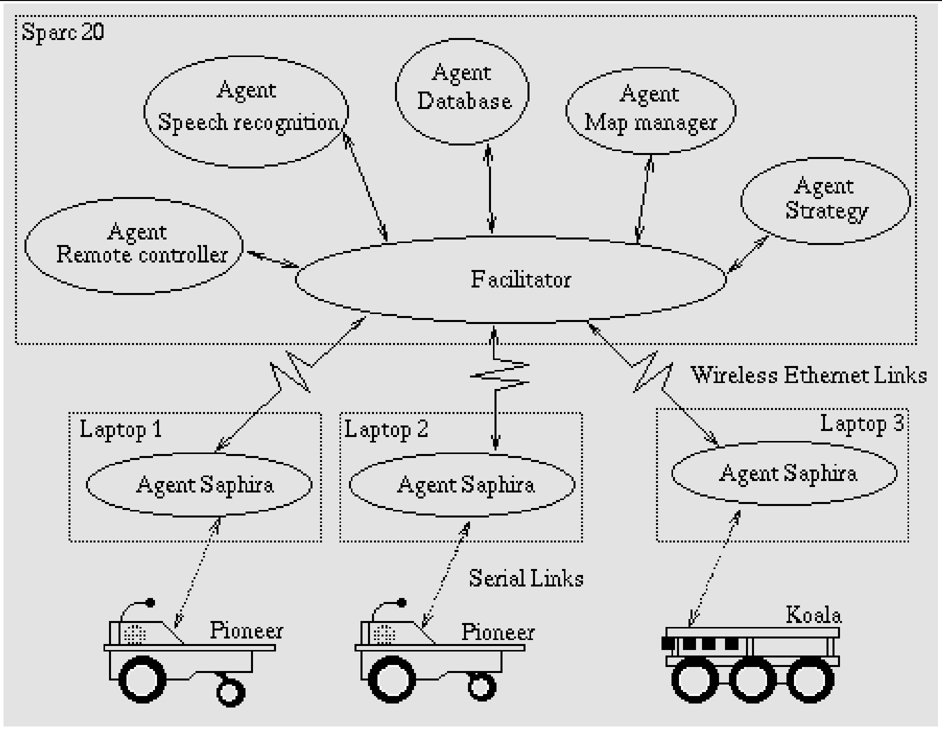


Figure 5 Organization of physical and software agents for the AAAI contest.

robots and the Sparc20 was through wireless Ethernet links.

All agents start running and connect to the facilitator, registering their capabilities so that other agents can send them requests. This is the essential part of the agent architecture: that agents are able to access each others' capabilities in a uniform manner. Many of the interface agents already exist at SRI: the speech and pen gesture recognition agents, for example. To access these capabilities for the robots, we have only to describe how the output of the interface functions should invoke robot commands. In addition, since agents are able to communicate information by asking and responding to queries, it is easy to set up software agents, like the mapping and strategy agents, to keep track of and control multiple robots. We'll briefly describe the capabilities of the agents.

3.1.1. Robot Information and the Database Agent

Each robot agent provides information about the robot state, and accepts commands to control the robot. The information includes

- Position with respect to the robot's internal coordinate system
- Robot movement status: stopped, moving forward, turning
- Currently executing behaviors on the robot

An interesting problem is how two agents maintain a consistent coordinate system. Commands that are robot-relative, e.g., "move forward", are interpreted with respect to the robot's internal coordinate system. Other commands, such as "Go to office EK288," must be interpreted with respect to a common global framework. The Database Agent is responsible for maintaining a global map, and distributing this information to other agents when appropriate. Each physical robot has its own copy of the global map, but these copies need not be exactly alike. For example, an individual map may be missing information about an area the robot has no need to visit.

During movement, each robot keeps track of its global position through a combination of dead-reckoning (how far its wheels have moved) and registration with respect to objects that it senses. It communicates with the database agent to update its position about once a second, and to report any new objects that it finds, so they can be incorporated into the global database and made available to other agents. In this way, the database agent has available information about all of the robot agents that are currently operating.

3.1.2. The Mapper Agent and Multimodal Input

If a robot becomes lost, it can query the facilitator to help relocalize. Currently, this means human intervention: the facilitator signals that a particular robot is lost, and asks for a new position for the robot. The state of each robot is displayed by the *map manager agent*, or *mapper*. All currently known objects in the database, as well as the position of all robots, are constantly updated in a 2-dimensional window managed by this agent. Figure 6 shows the mapper's view of the database contents. Corridors, doors, junctions, and rooms are objects known to the mapper. A robot's position is marked as a circle with an arrow in it, showing the robot's orientation.

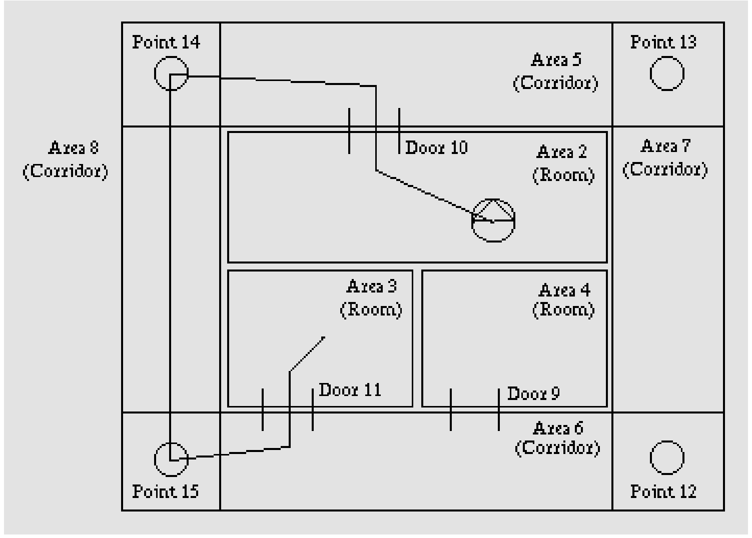


Figure 6 The mapping agent's view of the database.

To correct the position of a lost robot, the user can point to a position on the map where the robot is currently located, or simply describe the robot's position using speech input. This is one of the most useful features of the OAA architecture, the integration of multimodal capabilities.

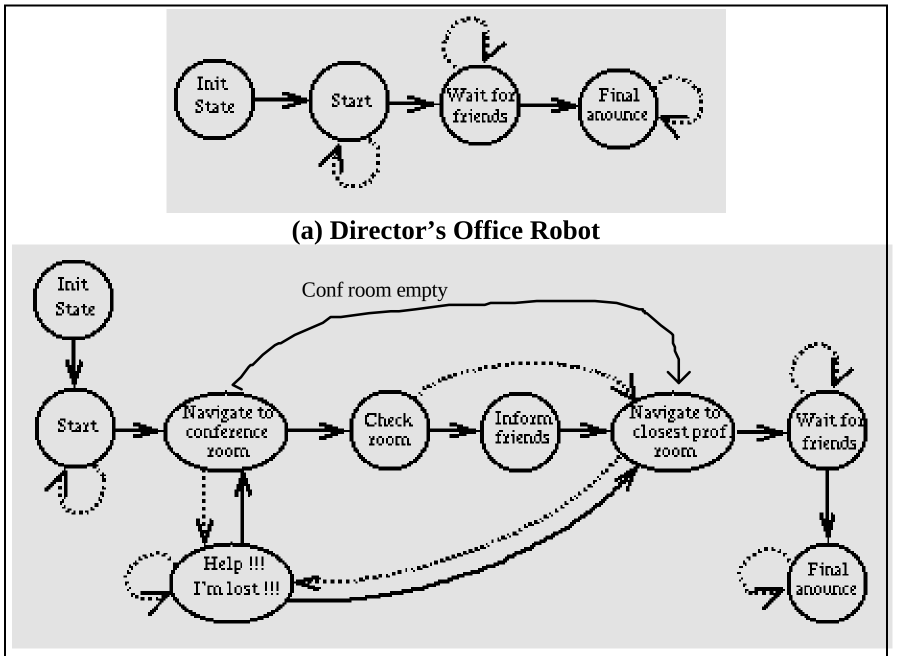
Currently, the system accepts either voice input or pen gestures. The interpretation of the gestures depends on context. For instance, when the robot is lost, the user can tell it where it is by drawing a cross (for the location) and an arrow (to tell the robot where it faces) on the map. Using 2D gestures in the human-computer interaction holds promise for recreating the paper-pen situation where the user is able to quickly express visual ideas while she or he is using another modality such as speech. However, to successfully attain a high level of human-computer cooperation, the interpretation of on-line data must be accurate and fast enough to give rapid and correct feedback to the user. The gesture recognition engine used in our application is fully described in [Julia 1995]. There is no constraint on the number of strokes. The latest evaluations gave better than 96% accuracy, and the recognition was performed in less than half a second on a PC 486/50, satisfying what we judge is required in terms of quality and speed [Moran 1996]

Given that our map manager program is an agent, the speech recognition agent can also be used in the system. Therefore, the user can talk to the system in order to control the robots or the display. For instance, it is possible to say : ``Show me the director's room" to put the focus on this specific room, or ``robot one, stop", ``robot one, start", to control a given robot.

Using the global knowledge stored in the database, this application can also generate plans for the robots to execute. The program can be asked (by either a user or a distant agent) to compute the shortest path between two locations, to built the corresponding plan and send it to the robot agent. Plans are locally executed through Saphira in the robots themselves. Saphira returns a success or failure message when it finishes executing the plan, so the database agent can keep track of the state of all robots. In the figure, the plan is indicated by a line drawn from the robot to the goal point, marked by an “X”.

3.1.3. The Strategy Agent

The *strategy agent* controls the coordinated movements of the robots, by keeping track of the total world stated and deciding what tasks each robot should perform at any given moment. While it would be nice to automatically derive multiagent strategies from a description of the task, environment, and robots, we have not yet built an adequate theory for generating efficient plans. Instead, we built a strategy for the event by hand, taking into account the various contingencies that could arise. The strategy was written as a set of coupled finite-state machines, one for each robot agent. Because the two Pioneer robots had similar tasks, their FS machines were equivalent. Figure 7 shows the strategies for these agents.



(a) Director’s Office Robot

(b) Traveling Robot

Figure 7 Finite State Strategy Machines for the two types of robots.

Note that the FS strategies are executed by the strategy agent, not the robots. Each node in the FS graph represents a task that the strategy agent dispatches to a robot, e.g., navigating to a particular location.

The robot in the Director’s room has a simple task: just wait until all the other robots have completed their task, then announce the time and place of the meeting. Transitions between states are triggered by events that come into the database: a robot successfully completing a task, or some condition becoming known, e.g., whether a conference room is empty or full. The dark arrows indicate successful completion of a task, while the dotted arrows indicate failure.

Both traveling robots have the same strategy. After initialization, they go to a conference room (the strategy agent makes sure they pick different rooms). At any point during navigation, if they get lost, they signal the strategy agent that the navigation was unsuccessful, and the strategy agent asks the mapping agent to return a new location for the robot. This happened several times during preliminary runs, when one of the robots attempted to navigate through a conference room and got lost. We were able to tell the robot where it was, and keep going from that point.

Arriving at a conference room, the robot checks if the room is empty. If so, it informs the database, continues on to the nearest professor's room. If, while one robot is navigating to its conference room, the strategy agent learns that the other robot has found an empty one, it immediately starts the first robot navigating towards the professor's office. Once at the professor's office, the robots announce the expected time of the meeting, based on estimates of how long it will take the last robot to reach its professor's office.

4. Conclusion

The AAAI robotics competition has come a long way since its inception in 1992. At this point, robot navigation in office environments is becoming increasingly reliable, so that we are able to concentrate on interesting problems of high-level strategy, including efficient management of teams of robots. The integration of software agent tools has proven to be a great benefit, making human communication with the robot much more natural and easier to develop. But there still is a need to develop efficient multiagent planning tools, that is, planners that develop efficient strategies for teams of robots.

5. The Team

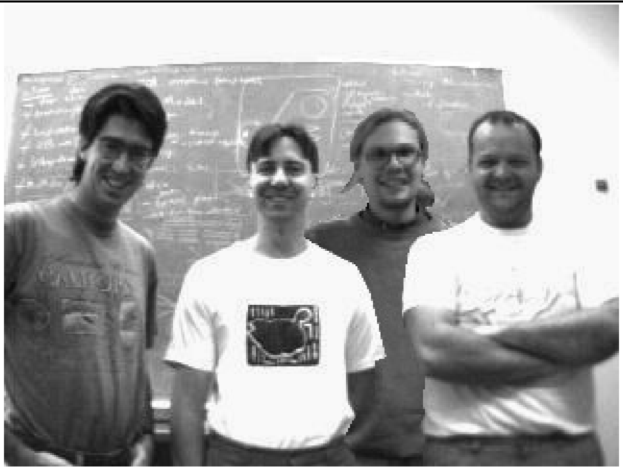


Figure 8 The SRI International Team: Adam Cheyer, Kurt Konolige, Didier Guzzoni (Swiss Federal Institute of Technology), Luc Julia

6. Bibliography

Rodney A. Brooks. *A layered intelligent control system for a mobile robot.* Proceedings of the IEED Conference on Robotics and Automation (1986).

Adam J. Cheyer and Luc Julia. *Multimodal Maps: An Agent-based Approach*. International Conference on Cooperative Multimodal Communication (CMC/95), 24-26 May 1995, Eindhoven, The Netherlands.

P. R. Cohen, A. J. Cheyer, M. Wang, and S. C. Baeg, "An open agent architecture," in AAAI Spring Symposium, pp. 1--8, March 1994.

Claire Congdon et al. *CARMEL vs. FLAKEY: A comparison of two winners.* AI Magazine, **14**(1) (1993).

Luc Julia and Claudia Faure. Pattern recognition and beautification for a pen based interface. In ICDAR'95, pages 58-63, Montreal, Canada, 1995.

M. Kameyama, G. Kawai, and I. Arima, "A real-time system for summarizing human-human spontaneous spoken dialogues," in the Proceedings of the Fourth International Conference on Spoken Language Processing (ICSLP-96), October 1996.

Kurt Konolige et al. *The SAPHIRA Architecture: A Design for Autonomy.* Journal of Experimental and Theoretical AI, to appear.

D. B. Moran and A. J. Cheyer, "Intelligent agent-based user interfaces," in Proceedings of International Workshop on Human Interface Technology 95 (IWHIT'95), (Aizu-Wakamatsu, Fukushima, Japan), pp. 7--10, The University of Aizu, 12-13 October 1995.

Douglas B. Moran, Adam J. Cheyer, Luc E. Julia, David L. Martin, Sangkyu Park. *The Open Agent Architecture and Its Multimodal User Interface.* SRI Tech Note, 1996.

D. B. Moran, A. Cheyer, L. Julia, D. Martin, SK Park, "Multimodal User Interfaces in the Open Agent Architecture," to appear in IUI97 Conference Proceedings (Orlando), January 1997.

Hans P. Moravec and Alberto E. Elfes. *High resolution maps from wide angle sonar.* Proceedings of the 1985 IEEE International Conference on Robotics and Automation (1985).

Alessandro Saffiotti, Kurt Konolige, and Enrique Ruspini. *A Multivalued Logic Approach to Integrating Planning and Control.* Artificial Intelligence **76** (1-2) (1995).

Kurt Konolige. *Operation Manual for the Pioneer Mobile Robot.* SRI, 1995

Philip R. Cohen and Adam Cheyer. *An Open Agent Architecture.* AAAI Spring Symposium, 1994

Adam J. Cheyer and Douglas B. Moran. *Software Agents.* SRI, 1995