Physical Applications of an Agent Architecture
Examples in Mobile Robotics and Surgery Training Systems.

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

February 25, 1997

1 Abstract

In this paper we explain how an agent based approach can solve some of our crucial problems in the two main fields of research we are working in. The first one (where we started to use an agent based philosophy one year ago) is mobile robotics, and the second one, shortly described here, is a simulator allowing surgeons to train laparoscopic interventions. In mobile robotics applications, an agent based method allowed us to reach two goals : allowing several users (through high level interfaces) to control and monitor distant robots, the second one was to make robots communicate each other in order to perform cooperative tasks. In our second field of work, agent based applications allow surgeons to share environments (tele teaching) through virtual reality based interfaces. Our agent based activities are developed by using and developing new tools over the Open Agent Architecture ^{TM} (designed at SRI International / AIC, Menlo Park, California).

2 Mobile robotics activities

The goal of our group is to provide users of mobile robots with high level interfaces, allowing them (assuming that they do not know anything or very little about mobile robots) to control several robots in a natural way. To describe our tools, we will use as frame work the system developed to compete in the AAAI robot contest[6] which took place last summer in Portland (Oregon). Thanks to an agent architecture, we allowed three robots to cooperate each other, under the supervision of a user (using a multimodal interface). This flexible approach turned out to be very robust and efficient : we won the competition.

2.1 The framework

As introduced, we present our system by showing what we have developed for the AAAI 96 mobile robots contest. Here are a few words about the event we competed in. 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.

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 [1] as an agent within the Open Agent Architecture (OAA)[2]. The OAA is

a framework for constructing multi agent systems that has been used by SRI and clients to construct more than fifteen applications in various domains [3]. Applying the OAA to the Office Navigation task in the robot competition was done to 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 [2][3]. 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 real time response (about 1 Hz) 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 (see figure 1). Agents interact with each other through an Inter agent 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 stand alone 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[1], 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).

2.3 Robots as physical agents

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.

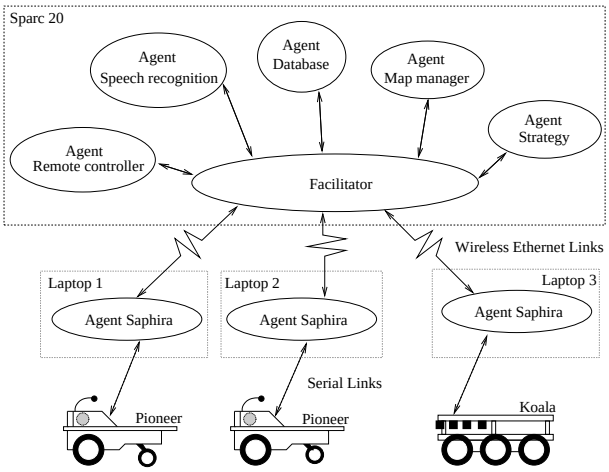

Figure 1: Organization of physical and software agents for the AAAI contest.

Figure 1 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[1] agent was running (under Windows 95) on a laptop computer, each equipped with sound devices and text-to-speech converters. The link between the 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.

Saphira

Saphira[1] is a powerful system which allows a robot to be independent by giving it high level behaviors (movement constraints, obstacle avoidance, map building, location to reach). This control program extracts pieces of information by reading and interpreting sensors values, such as sonars readings, camera images or wheels positions. It uses its knowledge, if any, (a priori known maps or spatial information) and compares it with the incoming information (from sensors) in order to adapt the robot's behavior as well as the information related to the environement (closed door, obstructed corridor). Therefore, distant agents can query Saphira to be get information about a specific robot or to send it commands such as a sequence of goals to execute.

Robot Information and the Database Agent

Each Saphira agent provides information about the robot state, and accepts commands to control it. 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

(recalibration, repositioning). 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.

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.

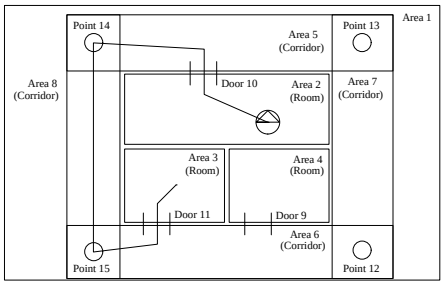

Figure 2: The mapping agent's view of the database.

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 2 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. 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 [4]. 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 PC0 486/50, satisfying what we judge is required in terms of quality and speed [5]. 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".

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 multi agent 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 (FS), one for each robot agent. Because the two Pioneer[1] robots had similar tasks, their FS machines were equivalent. Figure 3 shows the strategies for these agents.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, a Koala[10], 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.

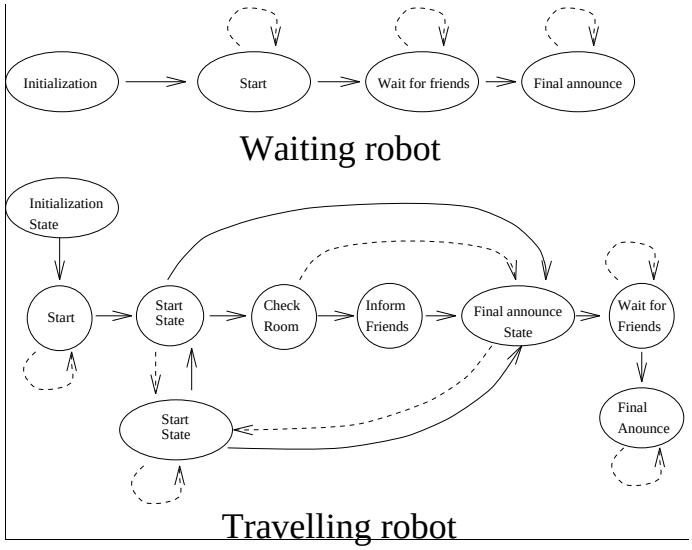

Figure 3: Finite state strategy machines.

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 (by using computer vision based analysis). 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.

3 Virtual surgery activities

3.1 Introduction

The Institute of Micro-Engineering at EPFL is developing, in collaboration with industrial partners, a training system for laparoscopic surgery. In this kind of surgery, also called minimally invasive therapy, surgeons insert a camera (endoscope) and tools inside the patient through a set of small holes. Thus, when a surgeon performs, he looks at a display to the images provided by the endoscope and does not stares at his hands anymore. This technique has a lot of advantages comparing to classic surgery. The recovery time is significantly shortened, so the period the patient has to spend in the hospital is reduced. Ugly scars can also be avoided and the trouble caused by the intervention is less painful. The system we intend to develop provides the user with both real time visual and tactile feedback information. This section explains how agents could help us to enhance our simulator in order to provide surgeons with high level multimodal interfaces, using concepts tested within our mobile robotics activities.

3.2 Agent compatible

The interesting point here, is that the whole system is designed as an agent compatible with the OAA^{TM}[2]. It means that every object in the scene (including the point of view) can be locally controlled to any 3D pointing device described above, or remotely linked to a distant agent. This expands the possibilities of such a simulator, the main advantages are :

- Sharing
  Each object of a given scene can be shared among different users located in distant places. For instance, during a training session a senior surgeon can reach the junior's station to remotely monitor the scene and tele control the student's work in order to teach her the right gesture.

- Reuse
  This feature allows us to reuse all existing agents such as the speech recognition agent, the database agent, robots agents or mapper with our virtual reality based system. For instance, the speech recognition system is seen by the simulator as a 3D pointing device. By saying commands like "move in", "move left", the user can move any object or the point of view in a given scene. It is important to underline that allowing the voice to be an input device can help to simplify the gesture and the cooperation between surgeons. It therefore allows them to think about new approaches in endoscopic surgery.

- Multimodality
  By reusing existing techniques, we could make our surgery interface work as a multimodal interface. The involved modalities are speech recognition and 3D pointing (mouse 3D, force feedback system)

3.3 The tools we developed

The heart of the system is a library called libptk[7]. This set of functions allows a programmer to quickly and easily create virtual reality based simulators. The main features of this library are :

- Real time rendering
  The library is built to provide real time simulations. It means that the frame rate should be at least about 25 Hz.

- Ability to deal with three types of objects
  What we call an object is a volume, rendered in a 3D scene. Libptk deals with three types of objects; the first one, *solids* , represent static volumes needed for the background of the scene. The second one, *organs*, are more dynamic than *solids*, they can deformed in real time when they are touched by elements of the third type of objects, *tools*.

- Texturation facilities
  Each object loaded in a scene can be textured. A texture is placed into the surface of an object through a texture laboratory[8], allowing the user to choose how the texture will stick to the object. In addition to that, textures can be classic 2D images or a real time video flow, coming from a video database or any conventional camera device[9]. This technique dramatically increases the level of realism of the scene, especially for dynamic effects such as bleedings of smoke diffusion.

- 3D pointing devices
  In a given scene, each object can be attached to a 3D pointing device (space mouse, 3D mouse, force feedback system or a syntaxer)

- Ability to write a file to describe the scene
  When the user is willing to create a scene, she only has to write a file which tells the simulator what kind of solids have to be loaded and how they will behave with respect to the events.

4 Conclusion

4.1 Advantages

As introduced, this paper presents the advantages we obtained by using an agent architecture within our two fields of research. In mobile robotics activities, we won the AAAI mobile robots competition by allowing

robots to behave as agents in order to cooperate. In our virtual surgery project, we developed a tool which allows people to share worlds through high level interfaces. Here, high level interface means multimodal tools (gesture and speech recognition) including real time 3D images and force feedback systems. For instance, during a real operation, the main surgeon asks his co-worker, who holds the endoscope, to move it in order to focus in the right area. In our simulator, by linking the point of view to the speech recognition agent, we could quite easily simulate this important feature. In addition to that, a senior surgeon can connect his or her interface to a distant junior's station, to directly monitor and interact with the student.

4.2 Synergies

Our two domains of research seem to be very different, but a lot of concepts developed in one can be (thanks to the reusability feature offered by an agent based approach) directly integrated into the second one. For instance, we are currently developing virtual reality based interfaces to control robots by using the tools designed for our surgery simulator. So, users can use multimodal 2D (figure 2) and 3D (figure 4) interfaces to control shared robots.

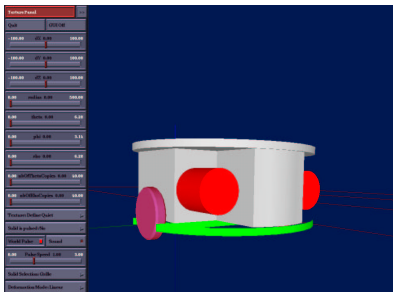

Figure 4: The mapping 3D interface agent's view of the database.

References

[1] K. Konolige. *Operation Manual for the Pioneer Mobile Robot.* SRI, 1995

[2] Philip R. Cohen & Adam Cheyer. *An Open Agent Architecture.* AAAI Spring Symposium, 1994

[3] Adam J. Cheyer & Douglas B. Moran. *Software Agents.* SRI, 1995

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

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

[6] Didier R. Guzzoni, Kurt Konolige, Adam J. Cheyer, Luc Julia, *Many robots make short work.* AAAI Journal, 1997

[7] P. Nguyen, *libptk A summary.* EPFL, 1996

[8] Y. Andenmatten, *Représentation réaliste d'organes.* EPFL, 1997

[9] F. Quinet, *Utilisation de techniques infographiques pour la génération de textures dynamiques.* EPFL, 1997

[10] K-Team *Koala USER MANUAL.* EPFL, Lausanne, September 1994