REAL: A Virtual Laboratory for Mobile Robot Experiments

Eliane Guimarães, Antonio Maffeis, James Pereira, Bruno Russo, Eleri Cardozo, Marcel Bergerman, *Member, IEEE*, and Mauricio F. Magalhães, *Member, IEEE*

***Abstract*—This paper presents REAL (Remotely Accessible Laboratory), a virtual laboratory accessible through the Internet. The objective of REAL is to provide remote access to a mobile robots infrastructure. REAL has been implemented as a new generation telecommunication service, not as a commonplace World Wide Web (WWW) application. As such, it employs a sophisticated access framework, a communication infrastructure able to support multimedia flows, and a component-based software construction. The architecture of REAL relies on open standards, such as WWW and its related technologies (HTTP, HTML, XML, Java, etc.) and a common object request broker architecture (CORBA).**

***Index Terms*—Robotics, scientific and engineering applications, software architectures, telecommunication applications, telematics.**

I. Introduction

SINCE THE widespread use of the World Wide Web (WWW) during the early 1990s, the Internet has been turning into a truly multiservice telecommunication network. Telecommunication services that in the past demanded specialized networks are being offered through the Internet (now termed *telematic services*). Such new services employ multimedia communication and more sophisticated service logic in order to provide the user with some degree of management and configuration capabilities. Examples of such services include telerobotics, virtual laboratories, distance learning and training, and live audio/video (A/V) multicasting.

Telematic services consist of three separated sessions: access, service, and communication [1]. The access session allows the user to register, unregister, manage, customize, and gain access to the service. The service session provides the mechanisms for supporting the execution, control, and management of the service. Finally, the communication session allows the parties involved in a service to communicate. REAL (Remotely Accessible Laboratory), a virtual laboratory accessible through the Internet, was implemented as a telematic service.

The paper is organized as follows. Section II describes virtual laboratories as telematic services offered through public and private networks. Section III describes a model for telematic services and the architecture of REAL. Section IV presents implementation details of this architecture. Finally, Section V closes the paper with some brief concluding remarks and future research directions.

Manuscript received February 20, 2002; revised March 26, 2002.

E. Guimarães is with the Renato Archer Research Center (CenPRA), 13082–190 Campinas, Brazil.

A. Maffeis, J. Pereira, B. Russo, E. Cardozo, and M. F. Magalhães are with the Faculty of Electrical and Computer Engineering, State University of Campinas (UNICAMP), 13083-970 Campinas, Brazil.

M. Bergerman is with the Genius Institute of Technology, 69075-904 Manaus, Brazil.

Digital Object Identifier 10.1109/TE.2002.804404

II. Virtual Laboratories

Practical experience is an important component of education. However, the time and money required for the planning and construction of scientific laboratories is outside the scope of many institutions. In this context, education and engineering scholars have proposed the creation of research and development virtual laboratories to allow users to perform experiments from a remote location. Users can plan and conduct experiments, collect experimental data, and analyze the results as if they were physically present in the laboratory.

Virtual laboratories are an important educational tool that bring together geographically distant research groups, allowing them to share data, documents, video, and voice, while integrating their computational and laboratory resources. Among the many benefits of virtual laboratories, the following are particularly important.

1) Resource sharing becomes a reality, improving the utilization of costly equipment.
2) Easier access to educational and research material is provided to students and professionals.
3) Scientific investigation standards can be established in areas where practical experimentation is a required part of research.
4) Reduction in travel time leads to productivity enhancement.

Two areas that benefit greatly from the concept of virtual laboratories are robotics and computer vision, since these are areas where experimental results are a fundamental part of the scientific investigation. The main objective of REAL [2] as a virtual laboratory is to provide researchers and students located anywhere access to an XR4000 autonomous mobile robot so that they can test and validate robot control and navigation methods without spending large amounts of money on an actual robot. To obtain a low-cost and widely available virtual laboratory, REAL was implemented through the existing Internet. Some literature [3]–[9] describe projects in the field of virtual laboratories and Internet robots.

REAL shares many common features with these projects, including WWW interactivity and video support for mission supervision. However, REAL considers virtual laboratories as complex telematic services, not as commonplace WWW ap-

plications. The following are some of the similarities between telematic services and virtual laboratories that lead to this approach.

1) The roles of service provider and service user are well identified.
2) Strict control of subscription, access, and use of the service must be enforced.
3) A need for effective service management strategies exists.
4) A need for real-time communication (for supporting telepresence in the case of virtual laboratories) exists.
5) Security and privacy issues must be well addressed.

While many projects on Internet robots simply allow users to manipulate the robot, REAL allows users to perform complex robotics experiments, such as controlling the robot with a user-supplied navigation algorithm. Section III presents a model for telematic services offered through the Internet. The REAL virtual laboratory is entirely based on this model.

With respect to education, REAL can enhance undergraduate and graduate courses in the field of robotics by supplying or complementing a laboratory infrastructure available to the students. Experiments in the field of autonomous navigation, environmental mapping, sensor fusion, mission planning, and robot control can be performed on the REAL virtual laboratory.

A typical use of REAL is as follows. In a course covering mobile robot control, the students are asked to implement a navigation algorithm presented in the classroom. The students employ simplified robot models and synthetic environments in order to test and tune their algorithms. The next step is to assess how these algorithms perform on a real robot. A navigation library supplied by the robot's manufacturer allows data acquisition and robot control functions to be performed in the C programming language. After a navigation algorithm is compiled and linked with this library, it is ready to execute on the robot's control processor, where it has full access to the robot's capabilities. As it executes, the algorithm can generate a trace file, storing such information as sensor readings, calculations performed, and actions taken. This information, retrieved a posteriori, is employed for tuning the navigation algorithm in order to contemplate peculiarities of the robot and its environment. The process repeats until the algorithm is able to drive the robot safely around the obstacles.

III. Telematic Services and REAL

The proposed model for telematic services is based on software components and frameworks [10]. The model allows the creation and deployment of services by tying together a set of reusable components. Methods, properties, events, and streams are the "glue" that links components together. Methods implement the component's functionalities (as in object-oriented approaches). Properties define a set of "state variables" normally related to the component's configuration and current status. Events are asynchronous notification messages generated by one component and received by zero or more components. Streams are continuous media flows (e.g., audio or video) generated by one component and presented by zero or more components.

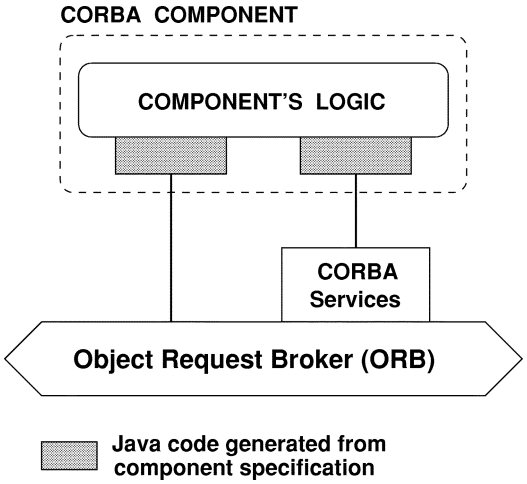

Fig. 1. A CORBA-compliant software component.

Components are deployed on a set of network nodes and can be aggregated into frameworks. A framework can also aggregate other inner frameworks. Frameworks provide a skeleton of an application that can be customized by the developer. A framework also defines a control structure to which all the components within it are subjected. Examples of frameworks are the common object request broker architecture (CORBA) [11] and Java media framework (JMF) [12]. In the first case, portable object adapters provide a standardized control for server objects attached to the object request broker (ORB). In the second case, JMF provides Java applications with standardized and configurable components for continuous media processing.

In the proposed service model, components and frameworks are realized in such a way that:

1) services can be built from component composition;
2) services can be deployed into heterogeneous platforms;
3) services can be easily configured and managed;
4) no software installation at the user's computer is necessary.

In order to fulfill these requirements, the components of our service model interact via CORBA. Component methods are called through the ORB, allowing location transparency between the interacting components. Properties, events, and streams are based on the corresponding CORBA specifications for these services, that is, the property service [13], the event service [14], and the A/V streaming service—A/V streams, respectively [15].

Fig. 1 illustrates a model of a CORBA-compliant component. Components can be deployed into the user's computer by implementing them as signed Java applets. Components are specified using extensible markup language (XML). From the component specification, an XML parser and an interface definition language (IDL) compiler generate a set of Java classes. These classes allow the component's interfaces to access the ORB and the needed CORBA services.

The architecture of REAL relies on frameworks and components as described previously. As such, three frameworks were implemented in order to provide the service parts: access framework, basic navigation framework, and advanced navigation framework. In addition, a framework for distance

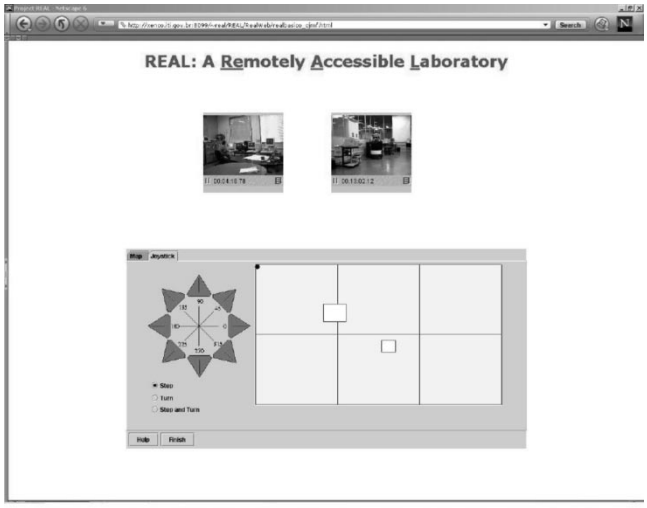

Fig. 2. Interface of the basic navigation framework with two video panels and one tabbed panel.

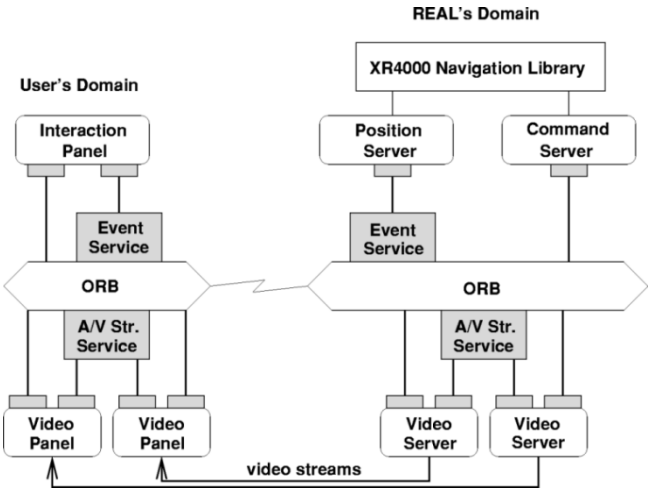

Fig. 3. Main components of the basic navigation framework.

learning in the field of robotics is under implementation. These frameworks are described in sequence.

A. Access Framework

The access framework implements, in part, the service architecture as specified by the Telecommunication Information Network Architecture Consortium [1]. From the user's viewpoint, this framework allows users to subscribe and unsubscribe to the services and to start a subscribed service. REAL offers four services, consisting of three modes of interaction (basic, advanced, and learning modes) plus a laboratory reservation feature in which the user can schedule a time slot for interacting with the robot.

From the service's point of view, the access framework offers a set of features concerning usage statistics, access control (e.g., usage upon reservation), and service control (e.g., service termination due to time expiration).

B. Basic Navigation Framework

The basic navigation framework offers a simplified mode of interaction in which the robot is viewed as a vehicle able to move from a starting point to a final point or to move in a sequence of discrete steps. This framework is composed of three components deployed at the user's domain: two for presenting real-time video and one for interaction with the robot. At the REAL's domain, four components are deployed—two for supporting the interaction with the robot and two for performing video capture.

The interaction component has a panel in which the user can select between two tabs: *map* and *joystick* (Fig. 2). The *map* tab presents a simplified position map to the user showing the robot's environment, the robot's initial position, and the obstacles constraining the robot's movements. The user can point the mouse to set a target for the robot's final position. When the target is set, a button labeled *move to target* is enabled. By clicking this button, the robot starts to move toward the target guided by a predefined navigation algorithm based on potential fields [16]. While the robot moves, the user can follow the trajectory, which is continuously updated on the position map. Video panels showing images from panoramic and on-board cameras complement the interaction between the user and the virtual laboratory.

The trajectory update is based on events exchanged between two components: the interaction panel and the position server (Fig. 3). The position server component computes the robot coordinates at intervals of 0.5 s. For each coordinate, the server generates an event position and pushes it to the interaction component that extracts its contents (the robot's coordinates) and updates the position map.

By selecting the *joystick* tab, the user is presented with the same map showing the robot's initial position and obstacles, plus an interface that mimics a joystick. In this mode of interaction, the user drives the robot (at steps of 20 cm or turns of 45°) by clicking on a set of navigation arrows. The user command is transmitted to the command server component that promptly translates it into robot movements. Fig. 3 shows the main components of the basic navigation framework.

C. Advanced Navigation Framework

The advanced navigation framework allows users to plan and execute complex robotics experiments that exploit the full capabilities of the robot. Experiments consist in executing a user-supplied code on the robot's control processor. This code can perform any operation on the robot, for instance, to perform a movement, to read a sensor, and/or to store data for further processing. The user-supplied code must be written in the C programming language. In order to assist the user in developing code, the framework supports code compilation, storage, and execution through its interface. A certain amount of disk space on the REAL file server is allocated to each subscriber. Source code, compiled code, and mission-acquired data are kept in this space for a period of time.

The advanced navigation framework relies on two components for file management—a file manager and its user interface. The file manager component, deployed at the REAL's domain, exports a CORBA interface with a set of methods for file management functions. These functions include file trans-

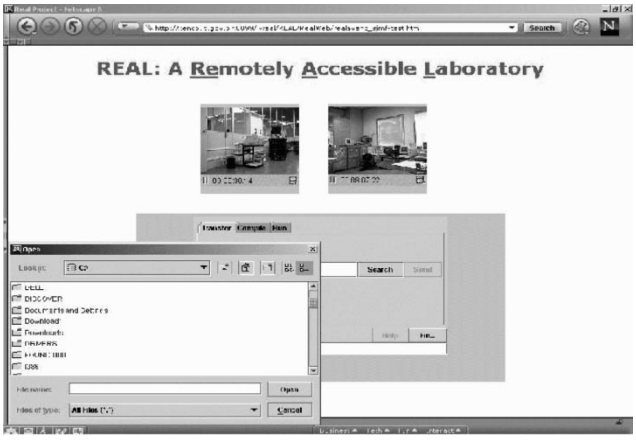

Fig. 4. File management interface of the advanced navigation framework.

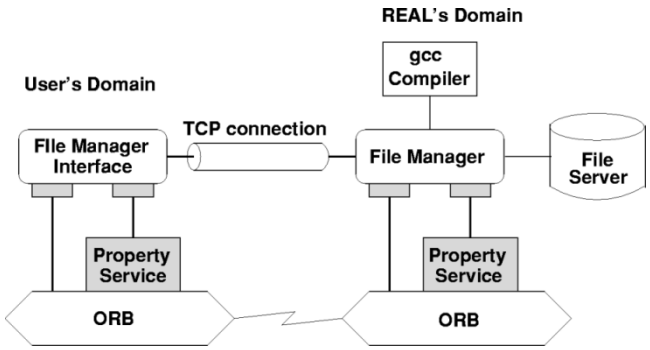

Fig. 5. Components performing file management in the advanced navigation framework.

ferring, compilation, and execution on the robot's control processor. The file manager also defines a set of properties that the interacting component may inspect and modify. These properties store information about the current user, the file being manipulated, the session's beginning time, and directives for compiling a file (parameters to the compiler).

The interaction process starts with the user submitting a source code for compilation through the file manager interface. The file can be chosen by browsing the local file system or by typing the file name (Fig. 4). After the selection, the file is transferred through a transfer control protocol (TCP) connection and stored on the REAL file server. Next, the user can compile the just-transferred file or any other file stored on the file server under the user's area. The results of the compilation are sent back to the user and presented on a separate navigator window. If the compilation succeeds, the executable code is also stored on the file server. From this point, the user can execute the code on the robot's control processor.

Fig. 5 shows the two components performing file management. As in the basic mode of interaction, video feedback is also available and provided by the same components of Fig. 3. However, the most valuable information provided by the advanced mode of interaction are sensor data acquired by the user-supplied code during its execution. These data are stored on the REAL's file server for further processing by the user.

Both navigation frameworks demand a set of preventive measures against a faulty or malicious navigation algorithm or sequence of moving steps. The approach was to design a watchdog daemon that periodically stops the navigation program, scans the sensors, and, based on the robot's direction and speed, determines whether the robot is keeping a safe distance from the obstacles. If this is the case, the navigation program resumes. Otherwise, the daemon stops the robot and kills the navigation program. This action generates a "navigation abort" notification to the user.

D. Distance Learning Framework

The distance learning framework supports a mode of interaction in which students follow the interactions with the robot conducted by an instructor. This framework has an interface similar to the basic navigation framework in which two video panels and a position map are employed in order to follow the robot movements. The interface has no interaction capabilities but receives and presents the same video streams and position events as generated by the video and position servers (Fig. 3).

This mode of interaction demands group communication in order to propagate video segments and events from a single producer to multiple consumers. Fortunately, both CORBA services employed for media streaming and event propagation support group communication.

In addition to the video panels and position map, three extra features are being implemented. The first feature is an audio stream that allows students to listen to the instructor in real time. Like video streams, audio streams are directly supported by the A/V streams service. The second feature is a presentation facility that allows training materials (pictures, graphics, diagrams, etc.) to be presented to the student's Web browser. A component running on the student's Web browser receives events encoding the Web links to be presented. The contents indicated by the Web links are accessed in a regular way through hypertext transfer protocol (HTTP). The source of uniform resource locator (URL) events is a component running at the instructor's Web browser with an interface allowing the instructor to select a Web link for presentation. Finally, the last feature is a chat facility that supports text-based communication between the instructor and students.

IV. Implementation Issues

Fig. 6 shows the infrastructure of the REAL virtual laboratory. On the REAL's domain, a local area network (LAN) integrates several servers, a router, and the access points for two wireless LANs (WLAN). The WLANs connect the robot's on-board computers to the wired LAN. The first WLAN, operating at 1.6 Mb/s, connects the processors that control the robot. The second WLAN, operating at 11 Mb/s, connects a portable computer that performs video capture from the on-board camera.

The servers perform robot control, service access control, user interaction, and video capture (from a panoramic camera). In addition, an HTTP server stores hypertext markup language (HTML) pages and Java archive (JAR) files. These servers are based on microcomputers and SUN Sparc machines running Linux and Solaris operating systems, respectively. On the remote user's side, a Java 2-enabled Web browser suffices for accessing the virtual laboratory. All client-side components were

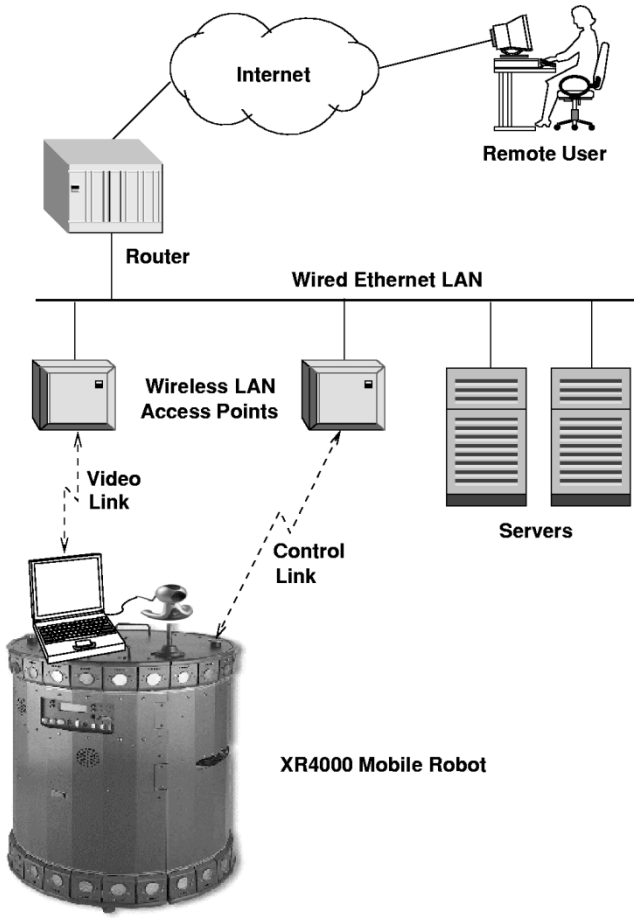

Fig. 6. Infrastructure of the REAL virtual laboratory.

implemented through signed Java applets downloaded on demand from the REAL's HTTP server.

In terms of software development, service components follow the model presented in Section III. The position and command server components are written in C++, while the remaining components are written in Java. Two CORBA implementations are employed—MICO [17] for C++ components and Java IDL [18] for Java components. MICO is a reliable public-domain CORBA implementation, while Java IDL is part of the Java 2 programming environment [19]. Interoperability among these CORBA implementations is assured by the Internet inter-ORB protocol (IIOP). The CORBA event, property, and A/V streams services were implemented in Java.

V. Conclusion

As the Internet is turning into a truly multiservice network with a steady increase in bandwidth and decrease in response time, the environment becomes more suitable for implementations such as REAL, a robotics virtual laboratory designed as a telematic service. REAL implements a very rich service model based on software components, open standards, and new WWW technologies, such as XML.

Today, REAL is being evaluated by the institutions participating in its development. The plan is to open the access to Internet users when the validation phase is concluded.

As the implementation described in this paper becomes fully operational, some extensions are being considered. These extensions rely on the network's ability to provide quality of service (QoS). One extension will incorporate a QoS framework that enables the delivery of high-quality video. The second extension is to allow control algorithms to execute at the user's domain. In this case, low levels of network delay and jitter must be assured in order to guarantee a stable control of the robot. Finally, a framework for cooperative robotics experiments is being considered as another possible extension.

References

[1] *The Tina Book*, Y. Inoue, M. Lapierre, and C. Mossotto, Eds., Prentice-Hall Europe, Hertfordshire, U.K., 1999.

[2] E. Guimarães, A. Maffeis, J. Pereira, B. Russo, M. Bergerman, E. Cardozo, and M. Magalhães, "REAL: A virtual laboratory for mobile robot experiments," presented at the 1st IFAC Conf. Telematics Applications in Automation and Robotics, Weingarten, Germany, July 2001.

[3] Proc. 1st IFAC Conf. Telematics Applications in Automation and Robotics, Weingarten, Germany, July 2001.

[4] B. Dalton and K. Taylor, "Distributed robotics over the internet," *IEEE Robot. Automat. Mag.*, vol. 7, pp. 22–27, June 2000.

[5] K. Golberg, S. Gentner, C. Sutter, and J. Wiegley, "The mercury project: A feasibility study for internet robots," *IEEE Robot. Automat. Mag.*, vol. 7, pp. 35–40, Mar. 2000.

[6] H. Hirukawa and I. Hara, "Web-top robotics," *IEEE Robot. Automat. Mag.*, vol. 7, pp. 40–45, June 2000.

[7] P. Saucy and F. Mondada, "Open access to a mobile robot on the internet," *IEEE Robot. Automat. Mag.*, vol. 7, pp. 41–47, Mar. 2000.

[8] R. Simmons, J. Fernandez, R. Goodwin, S. Koenig, and J. O'Sullivan, "Lessons learned from Xavier," *IEEE Robot. Automat. Mag.*, vol. 7, pp. 33–39, June 2000.

[9] D. Schulz, W. Burgard, D. Fox, S. Thrun, and A. Cremers, "Web interfaces for mobile robots in public places," *IEEE Robot. Automat. Mag.*, vol. 7, pp. 48–56, Mar. 2000.

[10] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*. Reading, MA: Addison-Wesley, 1997.

[11] R. Orfali and D. Harkley, *Client/Server Programming with Java and CORBA*, 2nd ed. New York: Wiley, 1998.

[12] Sun Microsystems Inc. Java Media Framework API (2002, Feb.). [Online]. Available: http://java.sun.com/jmf/

[13] Object Management Group (2002, Feb.) Property Service, v. 1.0, Tech. Rep. formal/2000-06-22. [Online]. Available: http://www.omg.org

[14] Object Management Group (2002, Feb.) Event Service, v. 1.1, Tech. Rep. formal/2001-03-01. [Online]. Available: http://www.omg.org

[15] Object Management Group (2002, Feb.) Audio/Video Streams, v. 1.0, Tech. Rep. formal/2000-01-03. [Online]. Available: http://www.omg.org

[16] J. C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.

[17] A. Puder and K. Roemer, *MICO: An Open Source CORBA Implementation*. San Mateo, CA: Morgan Kaufmann, 2000.

[18] Sun Microsystems Inc. Java IDL (2002, Feb.). [Online]. Available: http://java.sun.com/products/jdk/idl/

[19] Sun Microsystems Inc. Java 2 Platform Standard Ed. (2002, Feb.). [Online]. Available: http://java.sun.com/j2se

Eliane Guimarães received the B.S. and M.S. degrees in 1977 and 1990, respectively, from the State University of Campinas (UNICAMP), Campinas, Brazil, where she is currently pursuing the Ph.D. degree in the Faculty of Electrical and Computer Engineering.

She is a Computer Scientist at the Renato Archer Research Center, a federal research center located in Campinas, Brazil. Her research interests include component-based software engineering, distributed systems, and Internet-based services.

Antonio Maffeis received the B.S. degree from the Faculty of Technology of São Paulo (FATEC), São Paulo, Brazil, in 1991. He is currently pursuing the M.S. degree in the Faculty of Electrical and Computer Engineering, State University of Campinas, Campinas, Brazil.

He is a Faculty Member of FATEC. His research interests include distributed systems, Internet-based services, and software engineering.

James Pereira received the B.S. degree in computer engineering from the State University of Campinas (UNICAMP), Campinas, Brazil, in December 2001.

As an undergraduate student, he joined the REAL project team for 18 months. He is currently with AsGa, Paulinia-SP, Brazil, a company in the field of optical communications.

Bruno Russo received the B.S. degree in computer engineering from the State University of Campinas (UNICAMP), Campinas, Brazil, in December 2001.

As an undergraduate student, he joined the REAL project team for 18 months. Currently, he is a Consultant in the Belo Horizonte area, Brazil.

Eleri Cardozo received the B.S. degree from the University of São Paulo, São Paulo, Brazil, the M.S. degree from the Technological Institute of Aeronautics, São Paulo, Brazil, and the Ph.D. degree from Carnegie Mellon University, Pittsburgh, PA, in 1978, 1981, and 1987, respectively.

From 1979 to 1992, he was an Assistant Professor at the Technological Institute of Aeronautics. Currently, he is an Associate Professor in the Faculty of Electrical and Computer Engineering, State University of Campinas (UNICAMP), Campinas, Brazil. His research interests include distributed systems and computer networks.

Marcel Bergerman (S'94–M'96) received the B.S. and M.Sc. degrees from the University of São Paulo, São Paulo, Brazil, and the Ph.D. degree from Carnegie Mellon University, Pittsburgh, PA, in 1990, 1992, and 1996, respectively.

From 1997 to 2000, he was the Coordinator of the Robotics and Computer Vision Laboratory at the Information Technology Institute in Campinas, Brazil, where he managed or comanaged projects involving Internet-accessible laboratories, autonomous robotic airships for aerial inspection, and control of underactuated manipulators. Currently, he is with the Genius Institute of Technology, Manaus, Brazil, as a Project Leader in the technology group.

Mauricio F. Magalhães (M'97) received the B.S. degree from the University of Brasilia, Brasilia, Brazil, the M.S. degree from the State University of Campinas (UNICAMP), Campinas, Brazil, and the Ph.D. degree from the National Polithecnic Institute of Grenoble, Grenoble, France, in 1975, 1979, and 1983, respectively.

Currently, he is a Professor in the Faculty of Electrical and Computer Engineering, UNICAMP. His research interests include computer networks and distributed systems.