REAL—A Virtual Laboratory Built from Software Components

ELIANE G. GUIMÃRES, ANTONIO T. MAFFEIS, ROSSANO P. PINTO, CARLOS A. MIGLINSKI, ELERI CARDOZO, MARCEL BERGERMAN, MEMBER, IEEE, AND MAURICIO F. MAGALHÃES, MEMBER, IEEE

Invited Paper

This paper presents the Remotely Accessible Laboratory (REAL), a virtual laboratory accessible through the Internet. REAL allows a remote user to manipulate a mobile robot in a mode of interaction suitable to his or her level of expertise. A basic mode of interaction, dedicated to users with limited knowledge of robotics, supports interaction via teleoperation. In a more advanced level of interaction, expert users can plan and execute complex robotics experiments that exploit the full capabilities of the robot. In this mode of interaction, experiments in the field of autonomous navigation, environmental mapping, sensor fusion, mission planning, and robot control can be performed. Finally, a third mode of interaction allows a set of trainees to follow the interactions conducted by an instructor. The architecture of REAL departs from the commonplace World Wide Web applications, since it employs a sophisticated software architecture based on software components. This architecture presents a high degree of reusability that future developments in the field of Internet robots and virtual laboratories can take advantage of.

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

I. Introduction

Being traditionally employed for information sharing and exchanging, the Internet has become an important telecommunication infrastructure. The reason is the ubiquity of the Internet added to a permanent increasing of capacity in both core and access networks. New telecommunication-centric (or telematic) services offered through the Internet emphasize interpersonal interaction as well as interaction with remote environments (real or virtual). Such rich interaction schemes need the manipulation of continuous flows of information carrying audio and video. Of course, these services are enriched with information already manipulated by today's Web applications, including static and animated imaging, recorded audio and video, and text. In short, new Internet applications and telecommunication services are converging. As such, Internet applications must incorporate solutions to issues as real-time communication, quality of service (QoS), multiplicity of users, usage control and management, security and privacy, and, in some cases, billing.

Among these new applications, virtual laboratories (virtual labs) are of special interest. References [1]–[8] describe projects in the field of virtual laboratories and Internet robots. A virtual lab allows 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. Depending on the type of experiments supported, virtual labs must provide some teleimmersion mechanism in order to ”transport” the user to the lab. For instance, in the field of mobile robots, a virtual lab must allow the user to follow the robot during a user's submitted mission. This can be accomplished by live video (and audio) channels, recorded video (downloaded and presented *a posteriori*), animated models, bidimensional maps, or a combination of these.

Building applications as virtual labs is still cumbersome due to complexities involved in continuous media processing, strict usage control (subscription, usage upon reservation, security checks, etc.), asynchronous notifications of many kinds, among others. The development of REAL (Remotely Accessible Laboratory) [1], a virtual lab in the field of mobile robots, has shown that a component-based

Manuscript received February 12, 2002; revised December 2, 2002. This work was supported by the State of São Paulo Research Foundation (Fapesp), Brazil, and the National Council for Scientific and Technological Development (CNPq), Brazil.

E. G. Guimarães and M. Bergerman are with the Renato Archer Research Center (CenPRA), Campinas 13082-120, Brazil.

A. T. Maffeis, R. P. Pinto, C. A. Miglinski, E. Cardozo and M. F. Magalhães are with the Faculty of Electrical and Computer Engineering, Unicamp, Campinas 12081-970, Brazil.

Digital Object Identifier 10.1109/JPROC.2003.809212

software development can lower the development costs, improve reliability, and favor permanent improvements [9], [10]. Although well-established component models can be employed for coding many service functions, they lack of support for functions peculiar to telematic services such as media streaming and reliable notification mechanisms. Moreover, current component models are designed for homogeneous environments, a severe limitation for telematic services and other complex applications.

By the beginning of 2002, the Object Management Group (OMG) standardized a component model for the Common Object Request Broker Architecture (CORBA). The CORBA Component Model (CCM) [11] is not tightened to a particular environment, meaning that components developed according to this model can be deployed across multiple platforms. Due to its recency, products and tools supporting CCM in its full extent are not available yet.

To build a reliable architecture for REAL, we designed and implemented a component model suitable for telematic services: CCM-tel. CCM-tel is neutral, based on CORBA, and shares many common features with CCM, such as the ability of components to send asynchronous notification messages through a reliable notification channel (both unicast and multicast notification schemes are supported). However, two key differences are worthy of mention. First, CCM-tel supports stream interfaces able to produce, transfer, and consume media flows such as live audio and video. Stream interfaces can establish unicast (single producer/consumer) and multicast (single producer/multiple consumers) connections. Such interfaces are absent in the component models currently available. Second, CCM-tel components are specified in Extensible Markup Language (XML). As such, code generation can be accomplished by transforming the XML document that specifies the component into text documents containing Java and Interface Description Language (IDL) pieces of code that implement the component. XML document transformations are specified through Extensible Stylesheet Language Transformation (XSLT) [12], a standard from the World Wide Web Consortium. XSLT processors such as Xalan-Java [13] perform XML document transformations.

The paper is organized as follows. Section II presents the architecture of REAL, providing a brief overview of the CCM-tel component model. The next three sections present how users can interact with the virtual lab. Section III describes a mode of interaction suitable for nonexpert users. Section IV presents the advanced mode of interaction within which expert users can plan and execute complex robotics experiments. Section V presents a mode of interaction where trainees can follow the actions conducted by an instructor. Section VI details the implementation of the architecture of REAL. Finally, Section VII presents concluding remarks.

II. The Architecture of REAL

The functionalities implemented by REAL such as teleoperation and ability to execute user-supplied code can be found on many similar projects such as those reported in [2], [4], [8], and [14]. However, REAL differs from those projects in its software architecture. Instead of implementing a software architecture based on distributed objects, REAL employs a component-based software architecture. Component-based software architectures have most of their code generated from specification. In addition, most nonfunctional aspects such as QoS, security, and persistence are provided by the run-time environment supporting the components. As a result, component-based software developments lead to more reliable, efficient, and cost-effective software systems [10].

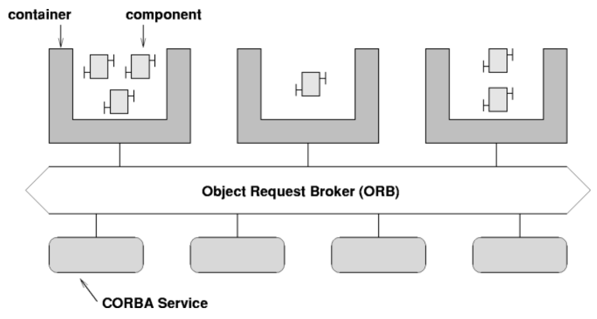

Fig. 1. Architectural pattern adopted by REAL.

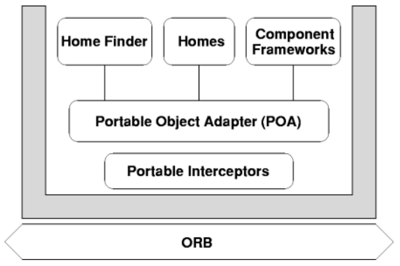

Fig. 2. Architecture of a CCM-tel container.

The architecture of REAL follows an architectural pattern shown in Fig. 1. The object request broker (ORB) is an interconnection infrastructure compliant to CORBA. The ORB interconnects a set of CORBA services and containers. Examples of CORBA services include the property service [15], event service [16], and media streaming service [17]. Containers are places where components are instantiated for execution. Containers provide the necessary resources needed by the components (e.g., resources for execution, communication, and storage).

The architecture of a container is presented in Fig. 2 and consists of five major elements.

1) Component factory (home): an object exposing an interface for component creation, location, and destruction.
2) Home finder: an object exposing an interface for finding homes within the container.
3) Component framework: classes providing the infrastructure needed by a component to interact with the container and with other components.
4) Portable object adapter (POA): ORB functions responsible for managing servant objects.
5) Portable interceptor: object responsible for intercepting the interaction among the components and

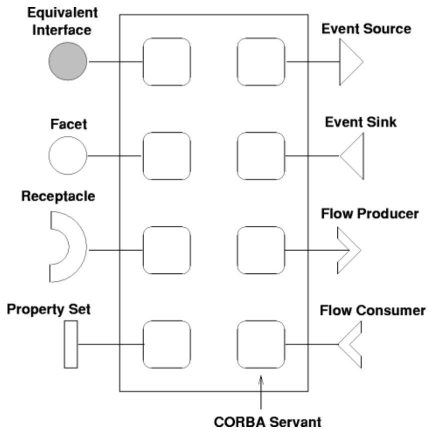

Fig. 3. Structure of a CCM-tel component.

taking appropriate actions such as security check and resource reservation.

A CCM-tel component's is structure presented in Fig. 3. The component consists of a set of servant objects, each one exposing an interface with methods accessible through the ORB. Three categories of interfaces are supported.

1) Equivalent interfaces: these provide basic access operations for retrieving the component's remaining interfaces, its home, and its unique identifier.
2) Ports: interfaces performing complementary roles such as client/server and producer/consumer (complementary ports are connected during the configuration phase of an application).
3) Property sets: interfaces providing access to the component's properties (attribute-value pairs holding information about state or configuration).

A component has a single equivalent interface and an arbitrary number of ports and property sets.

Six types of ports are supported by CCM-tel.

1) Event sources: interfaces able to produce asynchronous notification messages (events).
2) Event sinks: interfaces able to consume events when connected to event sources.
3) Flow producers: interfaces able to capture and transmit audio or video.
4) Flow consumers: interfaces able to receive and present audio or video when connected to flow producers.
5) Facets: general purpose interfaces supporting synchronous, remote procedure call-style operations.
6) Receptacles: these hold references to facets exposed by other components.

Properties, notification, and media streaming are transparently supported by the respective OMG service. The component frameworks in the container hide the manipulation of these services from the component developer.

A set of CCM-tel components were developed for REAL and grouped into three categories [18]: access, service, and communication components. Each category supports a corresponding *session*. An access session allows the user to subscribe/unsubscribe to the services, to authenticate subscribed users, and to start a subscribed service. A service session is established when the user starts a service and allows the user to perform the permitted actions (e.g., to manipulate the robot). Finally, the communication session is responsible for establishing media streaming connections necessary to the service. The service components for each mode of interaction will be described in the next sections. Access and communication components are described in the sequence.

A. Access Components

Access components implement part of the service architecture as specified by the Telecommunication Information Network Architecture Consortium [18]. These components support the establishment of access sessions between the service user and the service provider.

At the user's domain, two access components of the service architecture are installed. The provider agent (PA) component represents the service provider at the user's domain. It holds information about the access session the user has established, such as who the user is, his or her privileges, service sessions in course, and so on. The asUAP (access session—User Application) provides an interface that allows the user to establish a access session, to start services, and to receive notifications from the service provider.

From the service provider's point of view, the service architecture offers a set of facilities for usage statistics, access control (usage upon reservation, for instance), and service control (service aborting due to time expiration, for instance). Four main components at the service provider's side are the following.

1) Initial agent (IA): the component responsible for the establishment of access sessions between users and the service provider.
2) User agent (UA): this represents the user at the service provider's domain, being instantiated by the IA when a service session is established.
3) Service factory (SF): this instantiates the service and communication components.
4) Service session manager (SSM): this manages a service session and provides statistics about the utilization of the service.

The service architecture is highly modular and can provide access control to a wide range of telematic services. Fig. 4 shows how the components previously described are interconnected (dotted lines) and deployed.

B. Communication Components

Communication components support the establishment of communication sessions. In REAL, the communication session consists of two identical multicast video channels. A video channel is assembled by interconnecting one or more components exposing a video consumer port to a component exposing a video producer port. One channel transports live video captured from the robot's onboard camera, while the second channel transports live video captured from a panoramic camera pointed to the robot's environment. Fig. 5 shows the components employed in the communication session. Although a single video consumer component is shown, multiple consumers are supported as demanded

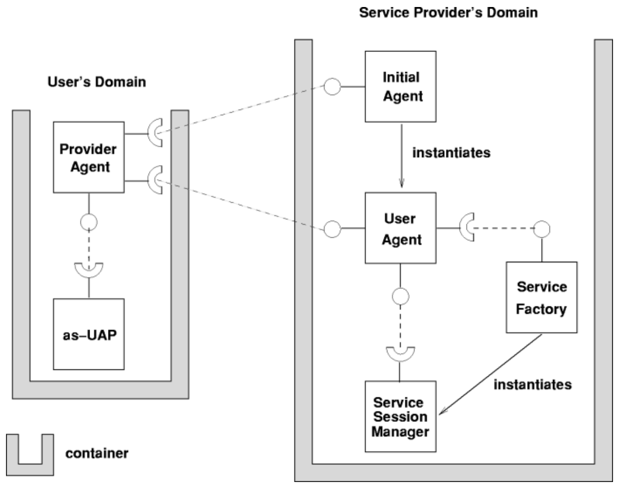

Fig. 4. Components supporting the access session in REAL.

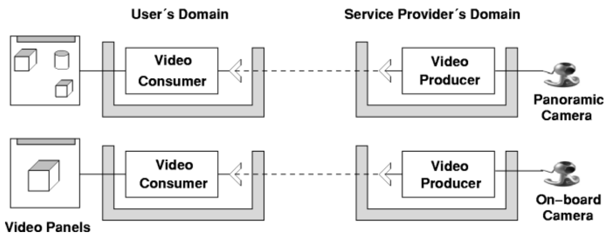

Fig. 5. Components supporting the communication session in REAL.

by the learning mode of interaction. Since components with stream ports perform CPU-intensive tasks (e.g., media capturing, encoding/decoding, and presentation), they are normally deployed on separate containers that provide dedicated resources for them.

III. The Basic Mode of Interaction

The basic mode of interaction offers two simplified forms of interaction with the robot. One form of interaction offers an interface that mimics a joystick. This interface allows the user to turn the robot in steps of 45°, to move straightforward in a step from 0.20 m to 2.0 m, or to perform both movements (turn, then move). In a second form of interaction, the user supplies the robot with a target and issues a command to move the robot to the target. The robot tries to reach the target by employing a navigation algorithm based on potential fields [19].

The user can follow the robot through a simplified position map. This map shows the obstacles constraining the robot movements, the robot itself, and the target (if set). Fig. 6 shows the user interface to the basic mode of interaction. The video panels maintained by the communication session are placed on the upper part of the interface while the position map and joystick interfaces are placed on the bottom part.

At the service provider's domain, two components are deployed. The Telecommand component exposes a facet port with three main methods: move, turn, and move to target. Once a method is called, the component translates its arguments to the corresponding robot movements. The Position Emitter component is responsible for updating the position map. This component reads the robot position in intervals of 1 s and generates asynchronous notification messages through an event source port. Notification messages encode the robot's current coordinates obtained from telemetry. Both Telecommand and Position Emitter components interact with a navigation software running on the processor controlling the robot.

At the user's side, two components are deployed: the Joystick Interface and the Position Map components. Both components have a receptacle port connected to the Telecommand's facet port. The Position Map component also exposes an event sink port that consumes notification messages generated by the Position Emitter component. Fig. 7 shows the components supporting the basic mode of interaction. In addition to these components, the components supporting the communication session (see Fig. 5) are deployed as well.

IV. The Advanced Mode of Interaction

The advanced mode of interaction allows expert users to test their own algorithms in a real-world environment. Such algorithms can exploit many areas of robotics, such as autonomous navigation, environmental mapping, sensor fusion, mission planning, and robot control. User-developed algorithms are coded in the C programming language; they run on the robot's control processor, and can perform any operation supported by the robot's application programming interface (e.g., to perform a movement, to read a sensor, and to store data for off-line processing). An amount of disk space on the REAL file server is allocated to each subscribed user for storing their source code, compiled code, and mission-acquired data.

On the user's domain, a component implementing a user interface for code submission is deployed. This interface presents the user with a file chooser (see Fig. 8) for browsing the local file system or his/her area on the REAL file server. Once a file is selected, it can be transferred, compiled, or executed on the processor that controls the robot. Files are transferred through a Transmission Control Protocol/Internet Protocol (TCP/IP) connection, since CORBA has no standard way to transfer large amounts of data through the ORB.[1]

On the service provider's side, a File Manager component implements the file server. This component exposes a facet port, a property set, and an event source port. The facet port defines methods for file transferring, compilation, and execution. The property set stores information about the file being manipulated (file name, file owner, directives for compilation, etc.). Finally, the event source port emits notification messages to the user's interface such as end of file transferring, URL of a page containing compilation results (presented on a separate user's navigator window), and mission status (e.g., abort messages). Fig. 9 presents the components supporting the advanced mode of interaction. As in the basic

[1]File contents can be transferred as sequences of octets, but ORBs may impose some maximum size for such sequences.

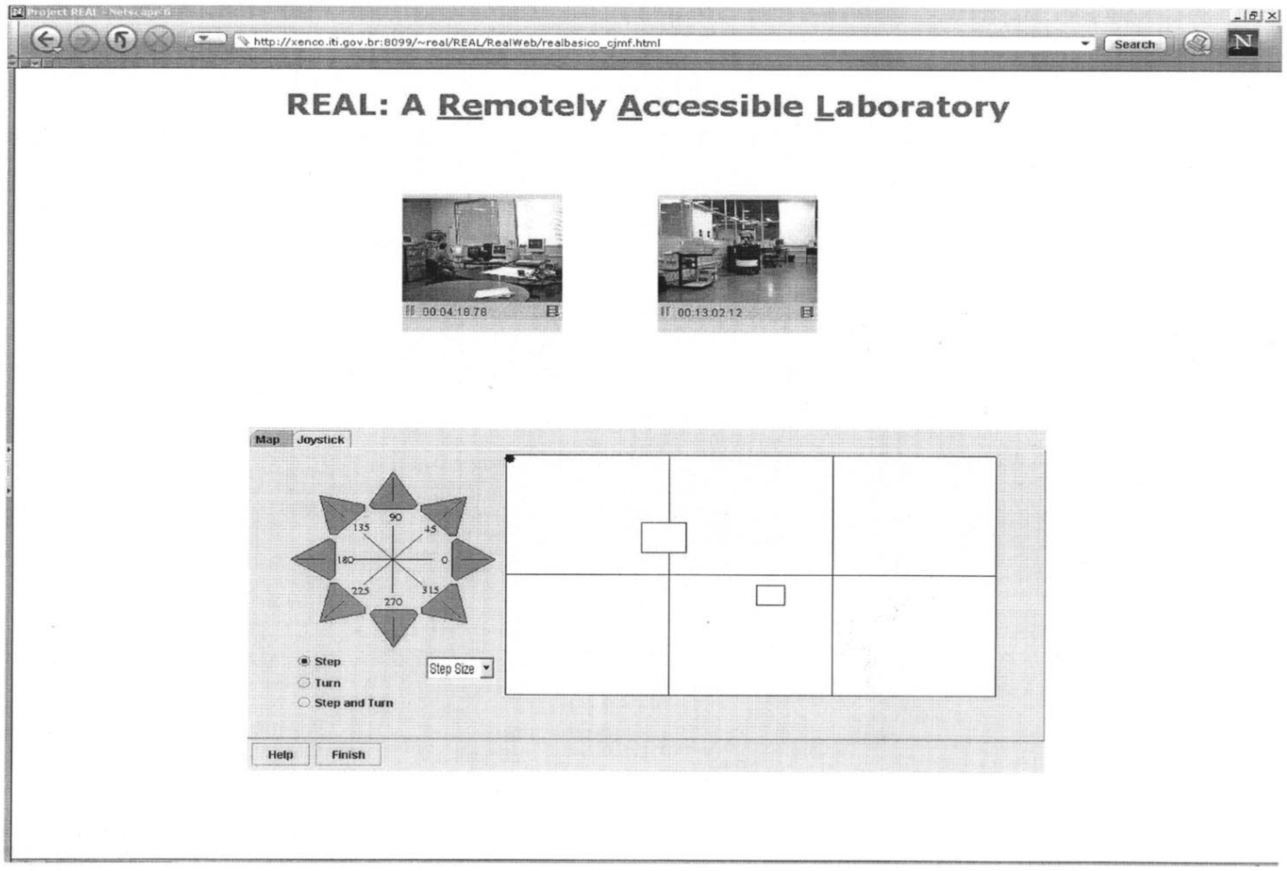
http://xenco.iti.gov.br:8099/~real/REAL/RealWeb/realbasico_cjmf.html
Search
REAL: A Remotely Accessible Laboratory
00:04:18.78
00:13:02.12
Map
Joystick
90
45
0
315
270
225
180
135
Step
Turn
Step and Turn
Step Size
Help
Finish

Fig. 6. The interface of the basic mode of interaction.

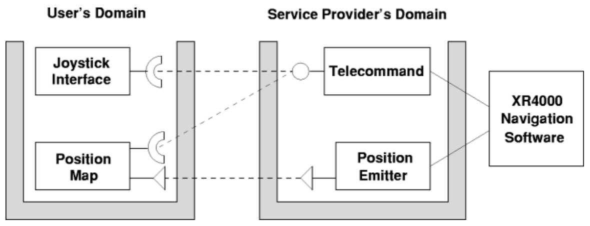

Fig. 7. The components supporting the basic mode of interaction.

mode of interaction, a communication session is established when the service session starts.

Both modes of interaction require a set of preventive measures against a faulty or malicious navigation algorithm or sequence of movements. These measures are implemented as a high-priority security daemon that periodically stops the navigation program, scans the sonar ring, and, based on the robot's direction and speed, determines if the robot is keeping a safe distance from obstacles. If this is the case, the navigation program resumes. Otherwise, the daemon stops the robot and kills the navigation program. In this case, the File Manager component is notified and issues a "navigation abort" notification event to the user.

V. The Learning Mode of Interaction

The learning mode of interaction offers a collaborative environment that allows multiple users to access the virtual lab at the same time. At the present, only a principal user (instructor) can interact with the robot, being the remaining users (trainees) able only to follow the robot with the aid of video panels and position map, as in the basic mode of interaction. Except for the instructor, the same interface of the basic mode of interaction is presented, but with the interaction capabilities disabled. The instructor can access the robot employing the basic or advanced modes of interaction.

The learning mode of interaction has the communication session enhanced with a multicast audio channel carrying voice from the instructor to the trainees. The audio channel employs two complementary components with audio ports. In addition, a facility for presenting didactic material was incorporated. This presentation facility also employs two complementary components. A Slide Chooser component is deployed at the instructor's computer and presents an interface containing a list of Web links (a series of slides in GIF format, for instance). When the user clicks on one element of the list, an event port emits an asynchronous notification encoding the URL assigned to the selected link. A complementary component, the Slide Presenter, deployed on each trainee's computer, receives the notification messages, extracts the URL, accesses its contents via HTTP, and presents the contents on a navigator's frame.

The learning mode of interaction demands multicast communication for audio, video, and notification messages. This form of communication is supported by the CCM-tel component model. Fig. 10 shows the additional components supporting the learning mode of interaction. Each component

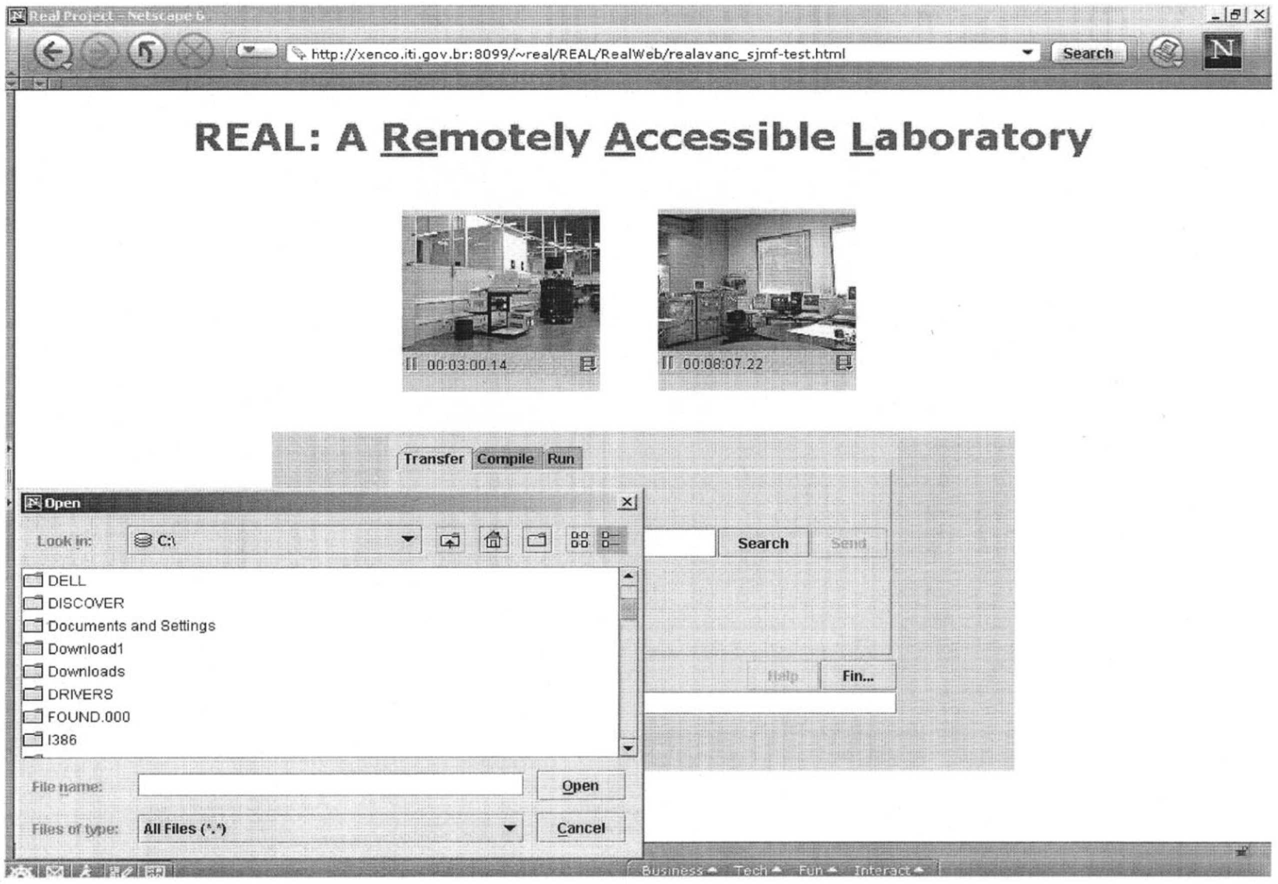

Fig. 8. The interface of the advanced mode of interaction.

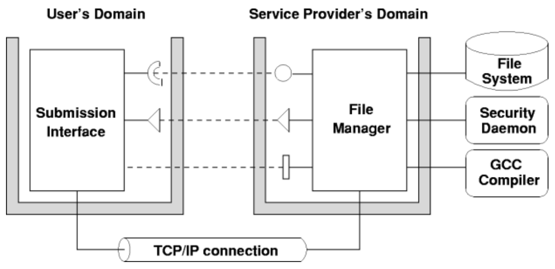

Fig. 9. The components supporting the advanced mode of interaction.

shown is deployed on separate container. A text-based chat capability was added to this mode of interaction in order to provide a low-bandwidth feedback mechanism from the trainees. The components supporting the chat service are not shown in Fig. 10.

VI. Implementation Issues

The software modules that interact directly with the robot must be written in the C programming language as demanded by the XR4000 application programming interface. User-supplied software and a set of modules supporting the three modes of interaction belong to this category. C-based software runs an a processor dedicated to robot control.

The code implementing the access, service logic, and communication consists of a set of software components as described in the previous sections. These components were

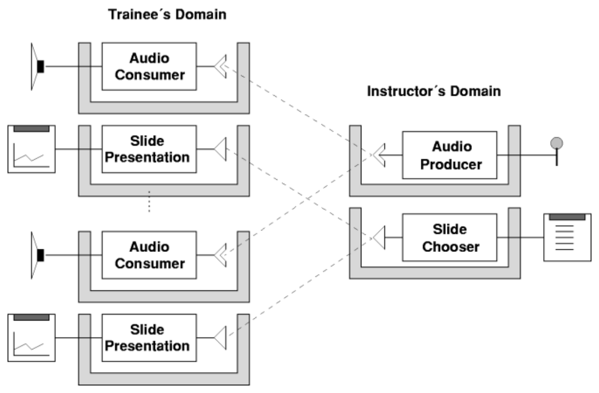

Fig. 10. Additional components supporting the learning mode of interaction.

coded in the Java programming language [20] with most of their code generated directly from their corresponding XML specification. CCM-Builder is a tool that generates component homes and component frameworks for each specified component. For instance, the Video Consumer component shown in Fig. 5 is generated by CCM-Builder from the following XML specification:

```
<component
 type = "VideoConsumer"/>
 <ports>
  <!-- video player port -->
  <flow-consumer name = "REAL-Camera"
   mediatype = "video">
 </ports>
</component>
```

Like components, containers are specified in XML and have their code generated by CCM-Builder. For instance, the component specified previously has its container specified in XML as the following.

```
<container
 type = "VideoPanel"
 deployment-form = "applet">
 <!-- components in this container -->
 <components>
  <componenttype type =
 "VideoConsumer"/>
 </components>
 <!-- POA policies -->
 <poapolicies
  thread = "ORB_CTRL_MODEL"
  lifespan = "TRANSIENT"/>
 </poapolicies>
</container>
```

Components deployed at the service provider's domain are implemented as Java processes, while those deployed at the user's domain are implemented as signed Java applets. A Web browser enhanced with the Java 2 plug-in is the only execution environment required at the user's side. A XML file named *deployment descriptor* specifies deployment attributes for each container. Examples of deployment attributes include media-related parameters (e.g., video frame size and rate), ORB parameters (name server location, interceptors, etc.), and application-specific parameters. From the deployment descriptor, CCM-Builder generates shell scripts for deploying process-based containers or HTML files for deploying applet-based containers.

Java IDL [21], the CORBA implementation of the Java 2 platform, was employed in both client and service provider sides. The CORBA services (property, event and media streaming) were entirely coded in the Java programming language. Media streaming relies on the Java Media Framework API [22] for audio and video capturing and presentation.

A local area network (LAN) connects a set of server processors, a router reaching the Internet, and two wireless LAN (WLAN) access points (see Fig. 11). The containers at the service provider's side are deployed on two separate processors. A third processor runs an HTTP server and stores the HTML and JAR files to be deployed at the user's domain.

The XR4000 mobile robot has two onboard processors dedicated to robot control. These processors employ a WLAN link of 1.6 Mb/s to communicate with the robot's control processor connected to the wired LAN. This processor runs the software that interacts directly with the robot (user-supplied code, security daemon, etc.). A laptop computer, fixed on the robot's top, performs video capturing

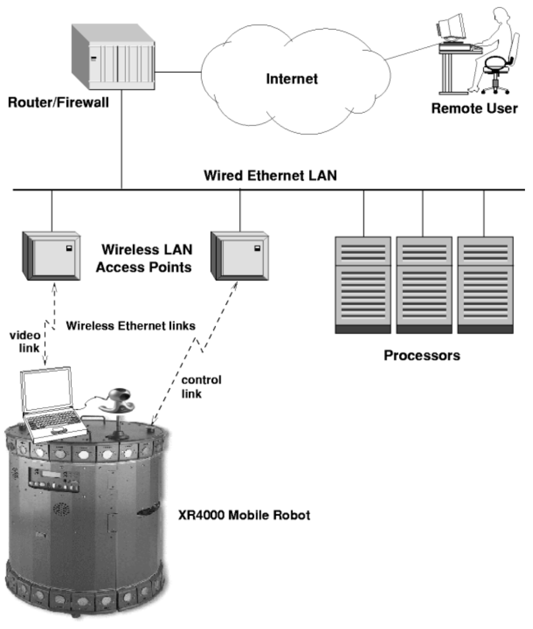

Fig. 11. The hardware infrastructure of REAL.

from the onboard camera and is connected to WLAN access point through a link of 11 Mb/s.

The laptop computer runs the Windows 2000 operating system. All the remaining processors, including the robot's onboard ones, run the Linux operating system.

VII. Conclusion

New Internet applications such as virtual laboratories and Internet robots are closer to telecommunication services than to commonplace Web applications. The reasons are clear: 1) the roles of service provider and service user are well identified; 2) strict control and management of subscription, access, and use of the service must be enforced; and 3) telecom-related issues such as real-time communication, QoS, security, and privacy must be well addressed.

Virtual laboratories and Internet robot services demand a high volume of software that must operate efficiently and reliably. A component-based software development process is able to fulfill such demands. As today's component models require a homogeneous distributed environment, we focused on the CCM, a neutral component model standardized by the OMG.

We developed CCM-tel, a neutral component model based on CORBA and suitable for building telematic services accessed through the Internet. REAL has demonstrated that software components are valuable, since they decrease the development costs and improve the quality of the code. All the components developed for REAL are based on the CCM-tel model.

Today, REAL is being evaluated by the institutions participating in its development. We plan to have a version of REAL supporting undergraduate projects in the field of robotics and artificial intelligence. This version will employ less sophisticated robotic equipment with a Java-based programming interface more suitable for educational projects.

Acknowledgment

The authors would like to thank former students B. G. Russo and J. L. Pereira for their valuable contributions to REAL.

References

[1] E. Guimarães, A. Maffeis, J. Pereira, B. Russo, M. Bergerman, E. Cardozo, and M. Magalhães, "REAL: A virtual laboratory for mobile robot experiments," in *Proc. 1st IFAC Conf. Telematics Applications in Automation and Robotics*, 2001, pp. 209–214.
[2] G. McKee, "The development of Internet-based laboratory environments for teaching robotics and artificial intelligence," *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 3, pp. 2695–2700, 2002.
[3] S. Jia, Y. Hada, G. Ye, and K. Takase, "Distributed telecare robotic systems using CORBA as a communication architecture," *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 2, pp. 2202–2207, 2002.
[4] K. Han, Y. Kim, J. Kim, and S. Hsia, "Internet control of personal robot between KAIST and UC Davis," *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 2, pp. 2184–2189, 2002.
[5] B. Dalton and K. Taylor, "Distributed robotics over the Internet," *IEEE Robot. Automat. Mag.*, vol. 7, pp. 22–27, June 2000.
[6] K. Goldberg, 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.
[7] H. Hirukawa and I. Hara, "Web-top robotics," *IEEE Robot. Automat. Mag.*, vol. 7, pp. 40–45, June 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] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*. Reading, MA: Addison-Wesley, 1997.
[10] G. Heineman and W. Council, *Component-Based Software Engineering—Putting the Pieces Together*. Reading, MA: Addison-Wesley, 2001.
[11] "CORBA Component Model v3.0," Object Management Group, 2002-06-65, 2002.
[12] (2002) XSL Transformations (XSLT) v1.0. World Wide Web Consortium (W3C). [Online]. Available: http://www.w3c.org/xslt
[13] (2002) Xalan-Java v2.4.1. Apache Software Foundation. [Online]. Available: http://xml.apache.org/xalan-j/
[14] F. Rodriguez, A. Khamis, and M. Salichs, "A remote laboratory for teaching mobile robots," presented at the 1st IFAC Conference on Telematics Applications in Automation and Robotics, Weingarten, Germany, 2001.
[15] "Property Service v1.0," Object Management Group, 2000–06-22, 2002.
[16] "Event Service v1.1," Object Management Group, 2001–03-01, 2002.
[17] "Audio/Video Streams v1.0," Object Management Group, 2000–01-03, 2002.
[18] Y. Inoue, M. Lapierre, and C. Mossotto, Eds., *The Tina Book*. Hertfordshire, U.K.: Prentice-Hall Europe, 1999.
[19] J. C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.
[20] (2002) Java 2 Platform Standard Edition. Sun Microsystems Inc.. [Online]. Available: http://java.sun.com/j2se
[21] (2002) Java IDL. Sun Microsystems Inc.. [Online]. Available: http://java.sun.com/products/jdk/idl/
[22] (2002) Java Media Framework API. Sun Microsystems Inc.. [Online]. Available: http://java.sun.com/jmf/
[23] R. Orfali and D. Harkley, *Client/Server Programming with Java and CORBA*, 2nd ed. New York: Wiley, 1998.

Eliane G. Guimarães received the B.S. and M.S. degrees from the State University of Campinas, Campinas, Brazil, in 1977 and 1990, respectively. She is working toward the Ph.D. degree with the Faculty of Electrical and Computer Engineering, State University of Campinas.

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

Antonio T. Maffeis received the B.S. degree in informatics from the Faculty of Technology, São Paulo (Fatec), São Paulo, Brazil, in 1991. He is working toward the M.S. degree with the Faculty of Electrical and Computer Engineering, State University of Campinas, Campinas, Brazil.

His research interests include distributed systems, Internet-based services, and software engineering.

Rossano P. Pinto received the M.S. degree in computer engineering from the Faculty of Electrical and Computer Engineering, State University of Campinas, Campinas, Brazil, in 2001. He is working toward the Ph.D. degree with the Faculty of Electrical and Computer Engineering, State University of Campinas.

His research interests include distributed systems, mobile computing, and computer networks.

Carlos A. Miglinski received the B.S. degree in informatics from the Regional University of Pinhal, Pinhal, Brazil, in 1994. He is working toward the M.S. degree with the Faculty of Electrical and Computer Engineering, State University of Campinas, Campinas, Brazil.

His research interests include component-based software engineering, distributed systems, and Internet-based services.

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

From 1979 to 1992 he was an assistant professor at the Technological Institute of Aeronautics. He is currently an associate professor at the Faculty of Electrical and Computer Engineering of the State University of Campinas. His research interests include distributed systems and computer networks.

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

From 1997 to 2000 he was the Coordinator of the Robotics and Computer Vision Laboratory at the Information Technology Institute, Campinas, Brazil, where he managed or co-managed projects involving Internet-accessible laboratories, autonomous robotic airships for aerial inspection, and control of underactuated manipulators. He is currently a Project Leader in the technology group at the Genius Institute of Technology, Manaus, Brazil, where he is responsible for technology prospection and assessment through product and technology feasibility studies, for establishing technological partnerships with universities and other research institutes, and for securing the investment needed for the realization of technology innovation projects.

Mauricio F. Magalhães received the B.S. degree from the University of Brasilia, Brasilia, Brazil, in 1975, the M.S. degree from the State University of Campinas, Campinas, Brazil, in 1979, and the Ph.D. degree from the National Polytechnic Institute, Grenoble, France, in 1983.

He is a professor at the Faculty of Electrical and Computer Engineering, State University of Campinas. His research interests include computer networks and distributed systems.