Int. J. Engng Ed. Vol. 19, No. 3, pp. 371–378, 2003
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Resource Sharing Software for Distance Learning in Engineering Education*

L. PETROPOULAKIS and B. STEPHEN
Dept. of Electronic and Electrical Engineering, University of Strathclyde, 50 George Street, Glasgow G1 1QE, Scotland, UK. E-mail: L. Petropoulakis@eee.strath.ac.uk

This paper presents the work being carried out in Strathclyde University to utilize generic methods for making computer-aided design (CAD) and simulation packages available on the Internet for distance learning and distance collaboration purposes. The system was developed over a 3-year period and has now evolved into a generic resource sharing structure with several features designed for use predominantly in education but also in other environments. This paper provides an overview of the development, illustrates the functionality and use of the system in sharing different applications and for various user-modes, analyses the significance of such systems in educational establishments, indicates the difficulties experienced thus far in implementation and gives a brief overview of the latest version of this development.

INTRODUCTION

THE ORIGINAL AIM was to develop a system where CAD and simulation packages such as MATLAB/LabVIEW could be used and shared over the Internet for distance learning by our engineering students. Early attempts concentrated on MATLAB software since this package is most commonly used in an Electrical Engineering Department. A conscious decision was also taken to support Octave a free-to-use (useful for students) alternative to early MATLAB versions. (Originally developed by the GNU consortium (www.gnu.org), Octave is available for Unix, Linux, NT and Windows. Its original design has a poor graphic user interface (GUI) and does not support a Simulink-type environment. Both these problems have now been resolved through our development.)

Providing similar functionality for both these packages and enabling easy construction of simulations using a Simulink-type environment was one of the challenges of the work. Our aims therefore were:

1. To develop an environment where users could share the same copies of MATLAB/Octave for teaching and co-operation purposes over Intranet/Internet connections.
2. To provide a Simulink-type environment which was compatible with both Octave and MATLAB and which could be used over intranet/Internet with full user sharing functionality (**Note**: MATLAB/Simulink's Web Server system only permits single user Internet access).
3. To ensure that a familiar and easy-to-use interface (through the use of simple browsers) was in place and that the minimum of additional software was required.
4. To ensure that the operation was not impeded by problems arising from bandwidth considerations through sending graphics files across Internet connections.
5. To ensure platform-independence since our MATLAB/Octave software was available on both Unix/Linux and Windows operating systems. Browser independence was also desirable.
6. To extend the system to other applications of normally used in education and elsewhere.

The objectives of platform and browser independence were easily achieved through a Java development. It was also clear from the outset that to minimise use of bandwidth a client-based graphic system would need to be used. This also implied that a new Simulink-type (also Java-based) interface needed to be developed to provide a graphical simulation environment with full sharing capabilities. The overall system design is totally modular and agent-based [1–5, 9]. Using agents also requires the use of mediators-type agents [6]. (Definitions of agents and mediators as applied in our context can be found in [11].) Through continued development, the system has now evolved into a general-purpose software platform for sharing applications across Internet/intranet connections.

This paper concentrates mostly on the earlier developments of the system and the experience in using it, but it also provides brief overviews of more recent versions and functions. Very recently, additional features have permitted the sharing of any software application (such as LabVIEW or AutoCAD, Word for Windows, Excel, Power Point, etc.) in a robust and consistent manner.

* Accepted 12 February 2003.

THE BASIC SYSTEM (SIMPLE INTRANET/INTERNET USE)

The basic system, written in Java 1.2 was ready by July 2000. This system had no sharing capabilities but it enabled single users to access an Octave/MATLAB application server remotely over Internet/intranet connections, through the use of simple browsers (e.g. Netscape 3.0 or above, IE 4.0 or above) in a Windows NT/98/95 or a Unix/Linux or similar system. (A version of this early system is available from www.webeng.org, by emailing us your request: we must draw your attention to the licensing conditions of MATLAB if you intend to use our software with this application.) Later versions of the software (permitting full sharing capability) are available to Scottish Educational Institutions only. (Full commercial versions of the software will be made available in future.)

The Octave/MATLAB application server must run on a Unix/Linux or a Windows NT-based operating system. (New versions, which allow full sharing capabilities, have several modes which are explained next.)

This system operates on a simple server-client basis. The application (Octave/ MATLAB) runs on the server and the user connects remotely to the server through a browser on a client machine. Commands typed in the browser are executed in the server and the results are returned to the browser for viewing. The system is capable of handling console commands as well as MATLAB m-files. Any data to generate graphics issued by MATLAB/Octave is passed on to the client and plotted locally. Since only textual information is passed over Internet connections, the system is very fast with practically no latency even with slow connections. Tests between Strathclyde (UK), Purdue (US), Dundee and Edinburgh universities indicated that the system is robust and well structured.

The sharing concept—passive and active users

The technical details of the sharing arrangement are illustrated in Fig. 1. The figure shows both the sharing ability and the cross-platform capabilities of our development (Note that the names used in the figure and in the following description are the actual names of our networked computers. For clarity only two client users are shown in the figure.)

On 'Poseidon'—the name of our Sun server, the MATLAB server Java application runs continuously 'listening' for connections. On PC machine 'McGuinn', User1 points the web browser to the 'Poseidon' web page containing the client Java applet. The ensuing execution of the applet produces a page—displayed on 'McGuinn'—and a login dialog requiring user name, password and type, e.g. student/tutor/supervisor. (A tutor is assumed to be a lecturer/teacher/trainer in charge of a whole class of students/trainees. Supervisors are assistants to tutors in charge of only a fraction of supervised students/trainees.)

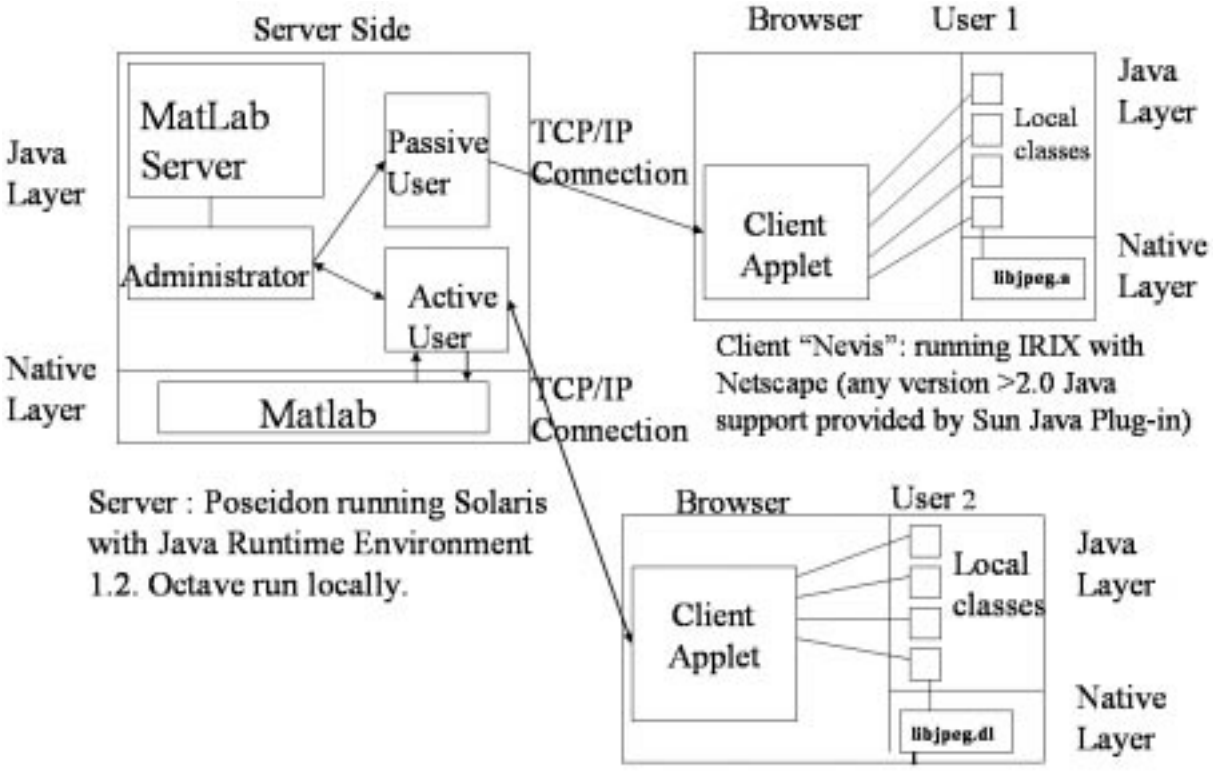

Fig. 1. The sharing concept.

On submission of correct information a connection with the server is established. If User1 is the person who activates the session, an ActiveUser and an Administrator objects are created. The Administrator controls a dynamic lookup table of all users 'looking on' to User1's session. User names enable the server to propagate the commands issued by User1 and also the responses of the system as a result of these commands to other users. Hence, on PC machine 'Nevis', User2 session is initialised in a similar way to User1 session. The only difference is that now a PassiveUser object is created to handle the connection with the User2 client.

Unlike an ActiveUser object, a PassiveUser does not 'own' the copy of MATLAB. PassiveUsers only receive the output from an ActiveUser's copy and cannot interact with the process except to view graphs, save plots to files, or terminate their session.

The above describes a typical 'tutor/multiple-student' teaching-by-showing session, where commands entered into the browser console on the tutor client machine are sent to the server machine where they are executed and the results are then passed on to all users connected to the system (i.e. a multicasting operation ensues). In cases where information other than commands to MATLAB is transmitted (e.g. send plot data to clients or audio or image files to the server), appropriate headers are used to denote this to the system. The system can also handle the submission of files for batch execution of multiple commands just as though they were supplied to the system from a keyboard.

The connections between the machines are standard TCP/IP connections common in Internet-based communications. The number of users on the system at any one time is limited only by software licences or server capacity for the maximum number of simultaneous users. The system also permits a dual server/client functionality on the same computer at the same time. Hence, any computer user can share applications on his/her machine with other users over Internet connections.

Sharing of applications

The features described in 2.1 allow a number of additional modes. These modes (shared, supervisory and supervised) permit the co-operative use of Octave/ MATLAB for the purpose of, for example, teaching, project development and synergistic design. These modes enable PassiveUsers to become ActiveUsers and vice versa.

This approach has important implications in educational, research and development environments, where two or more users 'share' the same copy of the software and can assume active or passive roles to obtain guidance or actively illustrate their ideas or designs. These modes are fully functional and additional refinements are added frequently to ensure extra robustness and preserve continuity of output. Switching between active and passive modes is effected through messages between any PassiveUser and the current

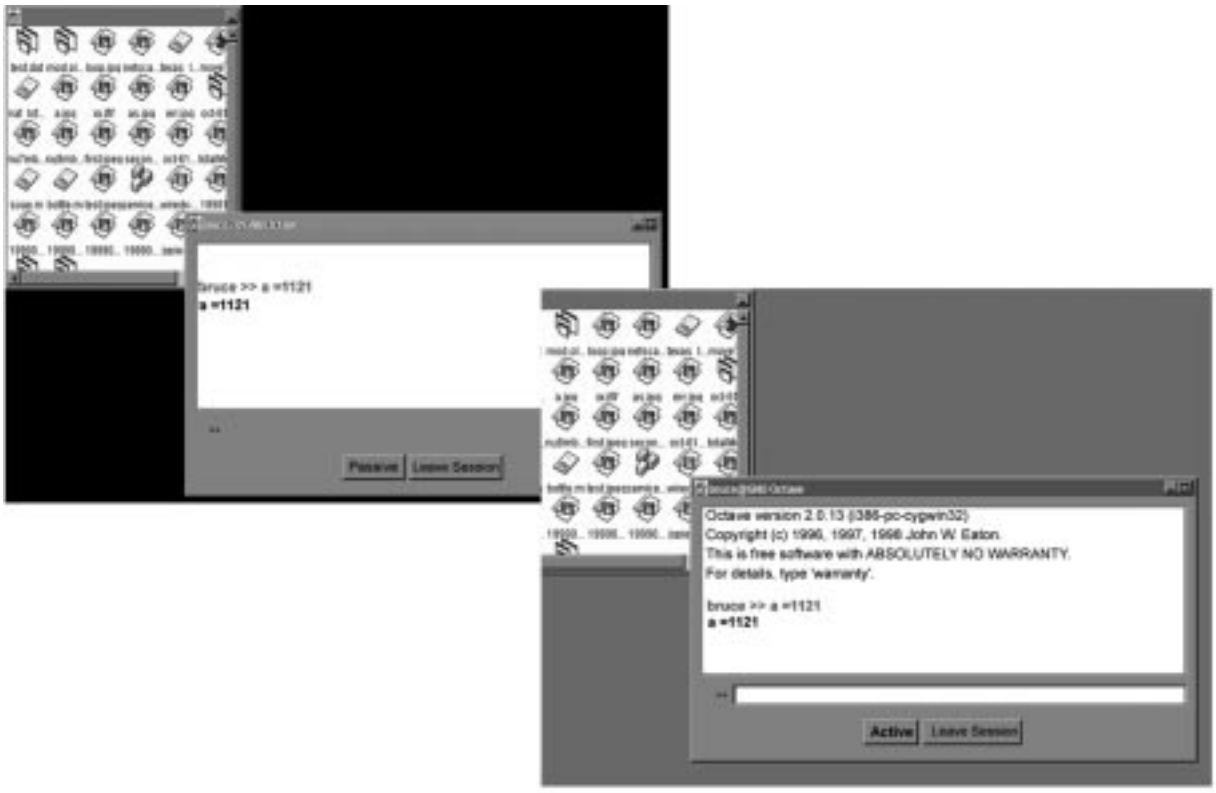

Fig. 2. The sharing mode.

ActiveUser. When an agreement is reached, the system performs the switch in real-time and without loss of data.

Figure 2 illustrates a typical sharing session with an ActiveUser having control of an Octave window (the command line where commands are issued is clearly visible) and the PassiveUser sharing (no command line exists in this user's browser).

It is possible, through these additional modes for students to use their own individual copies of MATLAB/Octave and for tutors/supervisors to view what the students are doing and if required take over control of any student copy to illustrate to the student where (s)he has gone wrong and finally return control back to the student. A history log of all student activities is also available to assist tutors in this task. Tutors have also the capability to supervise a large number of students and attend to each one in turn. Facilities, which alert the tutor to the fact that a student may need attention, also exist in the package. A typical case is presented in Fig. 3 where double-clicking the name (which is highlighted if attention is required) on the list of names on the left-hand side of the window enables tutors to take control of the corresponding student copy.

In case of distant collaboration, the system allows one **ActiveUser** per Octave/MATLAB copy, but an infinite (in theory) number of **PassiveUsers**. An **ActiveUser** can be alerted to the fact that a **PassiveUser** is seeking control of the software and (s)he can surrender this control when agreed. In this way, users can take control of the software in turn. It should be noted that the switching happens without interruptions or loss of data.

The system enables privileged users (e.g. tutors or supervisors) to login into multiple sharing sessions so as to monitor several different students or several classes. Switching between multiple sessions is merely a case of switching between browser copies or by pointing the same browser to a different connection point.

THE GRAPHICS USER INTERFACE (GUI) AND J-SIM

It was known at the outset of this project, that it would be inappropriate to use the existing Octave/MATLAB plotting systems, owing to bandwidth requirements for transmitting graphs and plots over the Internet. For this reason, we have now in place a plotting system that is client-based. Graphs are now plotted at the client end at a rate which the client can handle, whilst the commands and information as to what needs to be plotted is transferred over the Internet in the form of textual data. This is both fast and efficient. Our approach also ensures sufficient bandwidth availability for other multimedia (e.g. sound). The graphic interface is compatible with both MATLAB and Octave and an example of complex plot is shown in Fig. 4. Once generated, graphs may be rotated in 3D, zoomed, saved, etc. The GUI system is currently available for Linux, Unix, and Windows platforms.

Apart from the ability to plot graphics locally,

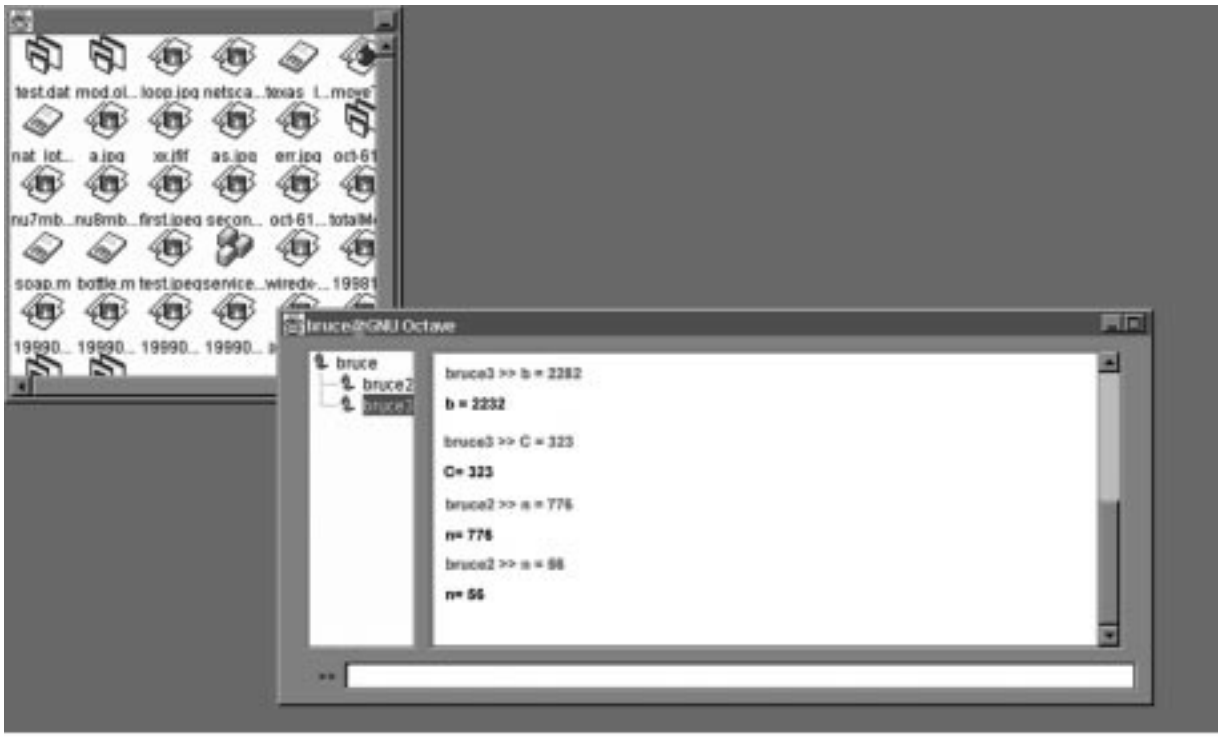

Fig. 3. Controlling several users.

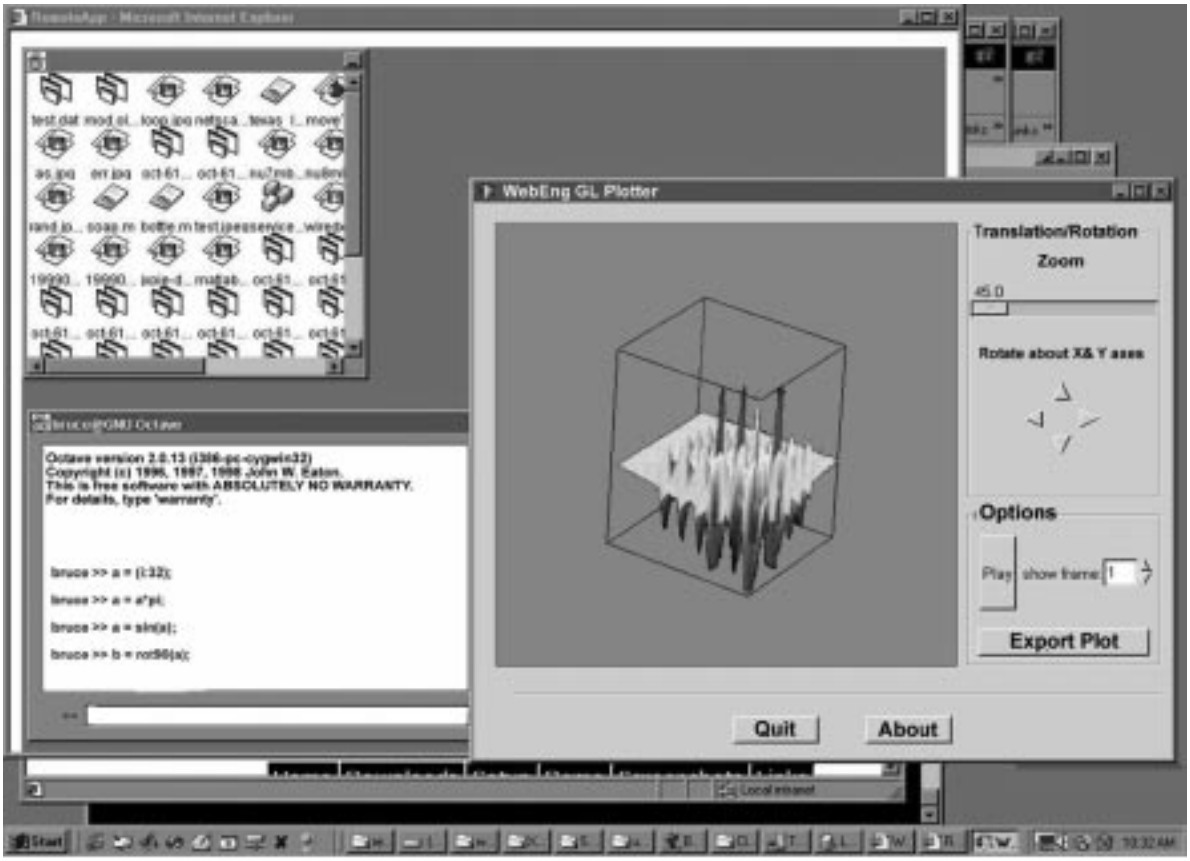

Fig. 4. The plotting facilities.

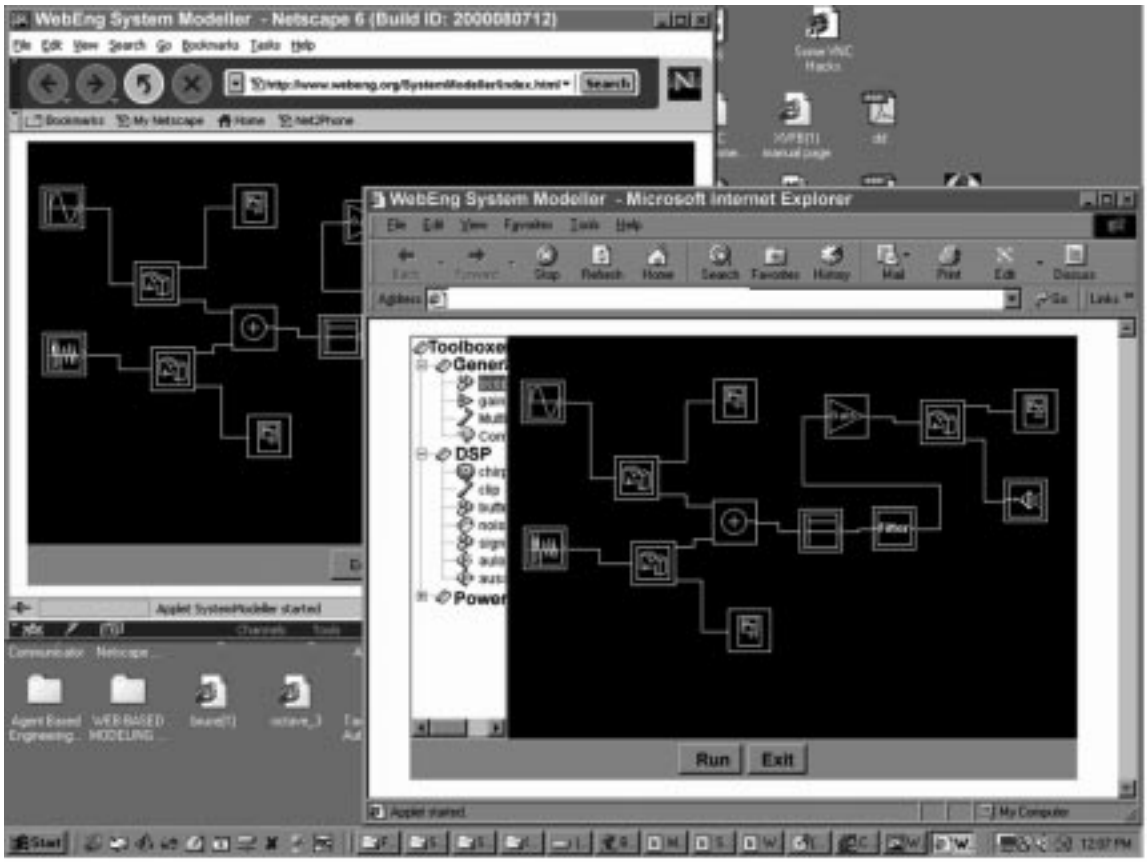

Fig. 5. J-SIM platform and browser independent.

the GUI system also enables us to meet the additional design requirement of providing a graphical simulation environment similar to Simulink. Using Simulink over Internet connections, implies the use of graphics over these connections which require a lot more bandwidth (Mathwork's solution of converting everything to html is another method).

Our solution to this problem is to use our existing client-based GUI in a Java-based Simulink-type workspace environment known as J-SIM. J-SIM operates at the client end and permits the construction of blocks (each containing a mathematical function) through a drag-and-drop action into the workspace. These blocks can then be linked in much the same way as Simulink blocks, to provide complex simulations. As the system is being built, the commands required to construct the simulation are propagated to all the computers of all observing users. Hence, they are able to view on their computer the design progress. The ActiveUser has full control of when to start, pause or stop this propagation process. The development enables users to switch between active and passive roles much in the same way as described in the previous section. A typical session illustrating two users sharing is shown in Fig. 5. Notice that the toolboxes are only available to the ActiveUser. If control of the application is switched, then the toolboxes will also switch accordingly. The browsers shown here are Microsoft Explorer 5.0 and Netscape 6.0.

In its present form J-SIM only has basic toolboxes, which can be individually customised with mathematical functions at simulation build time. However, this is rather slow and ready-made components may be desirable by users. To accommodate this, J-SIM is both user-expandable and easily customisable for particular environments. Users who wish to customise J-SIM for their own particular needs can do so by expanding the number and type of toolboxes available to suit their needs. However, in cases where sharing control of J-SIM is envisaged, the same toolboxes must appear at all client machines that need to control the application.

The system has been tested across various institutions in Scotland and elsewhere. Concerning connectivity and robustness the results have been excellent. In all tested cases, system performance was as designed without major concerns and with little time delay (even with limited bandwidth).

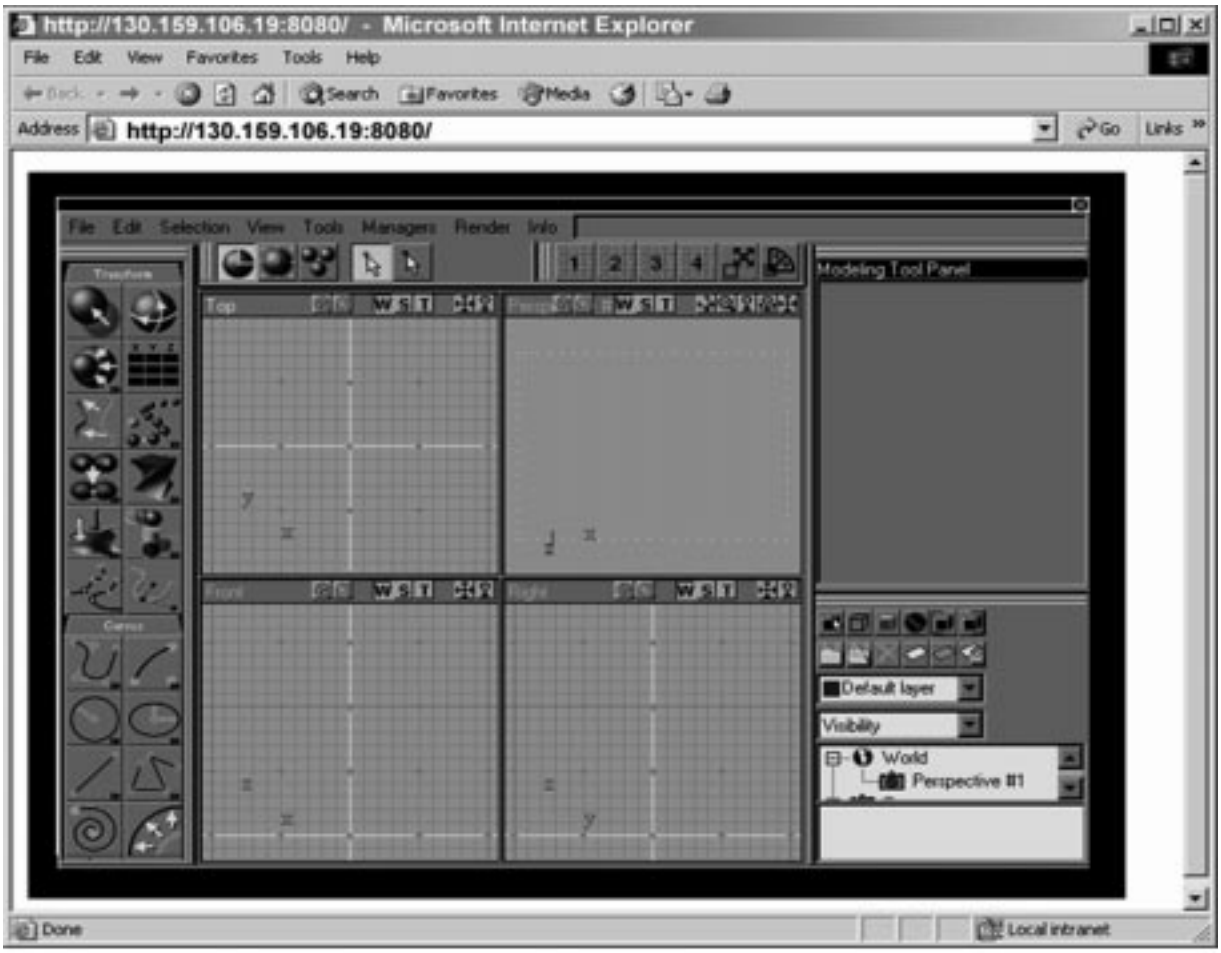

Fig. 6. Desktop sharing system.

DESKTOP SHARING—GENERALISING THE SHARING OF APPLICATION

The generic nature of our development implies that sharing individual applications as described and shown here for MATLAB and Octave, can easily be achieved provided that our method is incorporated as part of the application during the design stage. MATLAB and Octave are unique in that they allow textual information flow between their mathematical engines and plotting facilities. Hence, building the system in the way described around these applications was possible. In general, this will not be possible for other applications used for educational purposes such as LabVIEW or AutoCAD. To overcome this difficulty we have recently extended our development to include a desktop sharing environment, thereby permitting any application to be shared. A typical application shared in this way is shown in Fig. 6. The figure shows a 3D drawing system projected into a user's browser. The sharing operation in this case is much the same as before except that the information passed between server and clients is not purely textual anymore, but it also contains screenshot data. This can be rather slow at times particularly when large screen changes are required.

Although this latest extension to the system is fully functional, it is not yet completely platform independent. The constraint is that applications to be shared must run on a Windows NT/2000/XP operating system. The system operates well in real-time and enables sharing of all applications across mixed platform networks for a reasonable bandwidth availability (e.g. broadband). Further work to improve functionality and achieve full platform independence is being carried out.

FIRST EXPERIENCES IN USING THE SYSTEM

So far our development has been tried on just a few occasions. In most of these cases the purpose of the trials was to debug the software or test its functionality. However, enough trials have been done to evaluate issues such as:

- ease of use and familiarisation;
- ease of information exchange between users (i.e. using the system's message capability to communicate with each other);
- ability to attract tutor's attention;
- ability to respond to demands;
- maximum number of simultaneous users the system can support.

The latter issue is clearly hardware-dependent related to bandwidth availability and server capacity. In our case the most stringent test thus far involved fourteen simultaneous users without any adverse effects being observed. The upper possible limit of simultaneous users who can use this system is not currently known.

The fact that the MATLAB interface was contained within a browser appeared to unsettle a few experienced MATLAB users. However, it did not appear to impose particular difficulties in using the application. Users with no previous MATLAB experience did not find the interface unusual. The ability to share copies and be able to exchange messages on designs appeared not to be used significantly whilst users were within the same room. The tendency here was to attempt to physically attract the attention of the tutor (i.e. by raising their hand) rather than use the system to achieve the same result. We attribute this to habitual use of a traditional classroom approach. When this practice was restricted, students appeared quite happy to use the facilities of the system. Soon, however, it became evident that, in the case of non-experienced users, one tutor could not cope with the number of enquiries reaching his/her terminal. In the case of more experienced users, this particular problem appear to be containable.

One, almost predictable, result of asking various tutors to use the system was their reluctance to depart from known and set ways. It was only the realisation that there was nothing particularly time-consuming about learning the new system that persuaded them to try it out.

The main criticism related to the original system implementation was the fact that, owing to the command line as shown in the browsers (see Figs 2 and 3), users could no longer see the whole of the MATLAB command window and be able to scroll up and down so as to cut and paste commands whenever possible. This has been remedied through the use of a history command-window that the user can see, use for cut and paste, and save commands. This also allows for the subsequent easy creation of m-files (not possible using standard MATLAB facilities).

CONCLUSIONS

We have presented a user interface which, through the use of simple browsers, permits users to access the MATLAB and Octave simulation software through the Internet. This interface has been extended to a system that enables the simultaneous tutoring of a large (theoretically infinite) number of students by a single tutor. The system can be used both in classroom environments and also for distance learning. Additional system structures, which enable co-operation between researchers, have also been developed. This resource sharing system operates across various computer platforms such as (Linux, Unix, Windows, etc.) and using a variety of browsers.

The initial system has now been enhanced to incorporate desktop sharing facilities enabling sharing of all applications. This system is still

under development as it is not as yet fully platform-independent.

A number of international patent applications have been launched for this development, which (to the best of our knowledge) provides unsurpassed sharing capabilities for computer applications through several different modes to facilitate user collaboration.

Acknowledgements—The authors would like to acknowledge the contribution of the Scottish Educational Funding Council (SHEFC) for their continuing support of the WebEng project.

REFERENCES

1. L. Foner, Wh*at is an Agent? Crucial Notions*, MIT Press (1993).
2. S. Franklin, and A. Graesser, Is it an agent, or just a program? A taxonomy for autonomous agents, *Proc. Third Int. Workshop on Agent Theories, Architectures, and Languages*, Institute for Intelligent Systems, University of Memphis, Springer-Verlag, 1996.
3. M. Wooldridge, and N. R. Jennings, Intelligent agents: theory and practice, *Knowledge Engineering Review*, **10**(2), 1995, pp. 1–62.
4. T. Finin, Y. Labrou, and J. Mayfield, KQML as an agent communication language, in J. Bradshaw (ed.), *Software Agents*, MIT Press, Cambridge, 1997.
5. M. R. Genesereth, and S. P. Ketchpel, Software agents, *Communications of the ACM*, **37**(7), 1994, pp. 48–53.
6. G. Wiederhold, Mediators is the architecture of future information systems, *IEEE Computer*, March 1992, pp. 38–49
7. K. Goldeberg *et al*., Desktop tele-operation via the World Wide Web, *Proc. IEEE Int. Conf. on Robotics and Automation*, Nagoya, Aichi, Japan, 1995, pp. 654–659.
8. J. Paaso, Telematics for hich-tech engineering education: from computer-based teaching to telematic learning in software engineering, *Proc. EAEEIE 96 Conference*, June 12–14, 1996, Oulu, Finland, pp. 107–115.
9. N. Jennings, and M. Wooldridge, Software agents, *IEE Review*, January 1996, pp. 17–20.
10. R. L. Shuey, D. L. Spooner, O. Frieder, *The Architecture of Distributed Computer Systems*, Addison Wesley, 1997, ISBN: 0-201-55332-5
11. L. Petropoulakis *et al*., Agent-controlled Internet tools for computer-based distance training in industry and education, *Int. J. Eng. Educ. and Lifelong Learning*, **12**(1–4), 2002, pp. 267–276.

Lykourgos Petropoulakis obtained a 1st Class Degree in Aeronautical Engineering (1982) and a Ph.D. in Robotics and Control (1986) form Salford University UK. He worked briefly for Unimation UK before joining the Department of Artificial Intelligence, University of Edinburgh as a robotics researcher in 1987. In 1991, Dr. L. Petropoulakis joined the University of Strathclyde as a lecturer. Since 1995, his research has been mainly involved in areas relating to Internet-based collaboration and virtual laboratories. Dr. Petropoulakis has over 30 publications in various conferences and journals.

Bruce Stephen holds a B.Sc. in Aeronautical Engineering from the University of Glasgow, UK and MSc in Information Technology Systems from Strathclyde University, UK. He is currently a Research Assistant in the department of Electrical and Electronic Engineering at Strathclyde University where he is working towards his Ph.D. His current research interests include: intelligent agent based systems, distributed/resource sharing systems, neural networks, fuzzy logic, hidden Markov models.