Paradigms in Remote Experimentation

MJ. Callaghan, J. Harkin, TM. McGinnity and LP. Maguire
Intelligent Systems Research Center
University of Ulster, Magee Campus, Derry City, Northern Ireland, UK BT48 7JL

***Abstract*—Experience in teaching engineering related subjects has shown that a complementary approach combining theoretical and practical exercises is vital for effective learning. Increasingly, teaching institutions are offering remote access to distant laboratories as part of an overall e-learning strategy. However, the majority of remote access laboratories developed to date have suffered from a major deficiency, namely the provision of a web based environment that accurately recreates the collaborative group working and tutor driven experiences of traditional on-campus based laboratories. New collaborative remote experimentation environments and architectures are required to enable students in disparate locations to simultaneously and collaboratively complete complex experimental exercises. This paper presents several client-server paradigms that facilitate single user remote access, collaborative working and lecturer led approaches to the provision of remote experimentation facilities.**

***Index Terms*—E-learning, Embedded Systems, Remote experimentation, Collaborative working, Lecturer-led**

I. INTRODUCTION

The proliferation of distance education courses in recent years poses unique challenges for disciplines involving a high level of practical work [1, 2]. For electronic and electrical engineering disciplines, hands-on experience is essential for effective student learning. Traditionally students attended practical sessions in campus based laboratories at fixed times during the academic year [3]. This approach restricts access to laboratory resources to normal working hours which does not meet the needs of students requiring more flexible attendance patterns in line with current lifestyle commitments. The recent growth and widespread availability of high speed broadband Internet access facilitates the inclusion of increased functionality in the development of fully interactive and collaborative e-learning solutions.

Remote experimentation facilities offered as part of a web-based learning approach, affords a number of critical benefits and for engineering distance education courses it is the only realistic method of performing many experiments. This approach allows remotely located students to complete laboratory assignments unconstrained by time or geographical considerations facilitating the development of skills in the use of real systems and instrumentation.

The DIESEL project (Distance Internet-Based Embedded System Experimental Laboratory) was a three year distance learning project funded by the UK Engineering and Physical Sciences Research Council and located at the Intelligent Systems Research Centre on the Magee campus of the University of Ulster, Northern Ireland [4]. The project focused on the development of remote-access laboratories for embedded systems modules on several undergraduate and postgraduate courses. The facilities developed complement, extend and augment existing course provision by enabling students to conduct practical experiments in this area remotely via the Internet [5]. However the DIESEL project and indeed the majority of remote laboratories developed to date have suffered from a major deficiency, namely the provision of a web based environment that accurately recreates the lecturer led and group working experiences of traditional on-campus based laboratories [6, 7]. The architectures and remote experimentation environments described in this paper are extensions to the existing DIESEL project and describe approaches to the provision of collaborative and lecturer-led practical remote experimentation sessions for geographically dispersed students. The collaborative working environment described in this paper allows remote users to synchronously access and share on-campus resources to complete group work on complex engineering based practicals. The lecture-led environment presented allows the lecturer to create, manage and deliver live course material complemented by real-time access to advanced remote experimentation facilities to a widely dispersed audience of students. The approach shown here allows students to undertake real (non-simulated) practical exercises either individually or collaboratively, or alternatively under the direct guidance and supervision of the lecturer. This approach accurately recreates a similar level of lecturer-student and student-student interaction remotely.

Section 2 of this paper summarizes the authors' recent work in this area which focuses on overcoming existing deficiencies in similar and related remote experimentation projects. An overview of the single DIESEL client-server architecture, its components and their functionality are discussed. Section 3 discusses the extension of the existing client-server architecture to facilitate remote collaborative working between groups of students or between a lecturer and students. Section 4 describes the development and implementation of a new client-server architecture to facilitate live lecturer-led collaborative remote experimentation. Section 5 summarizes the authors' research to date and discusses future research opportunities in this area.

II. SINGLE CLIENT-SERVER ARCHITECTURE

The existing DIESEL client-server architecture is a non-collaborative learning environment for remote experimentation which allows individual students to perform real, non-simulated hardware based laboratories for embedded systems remotely, unconstrained by temporal or geographical considerations. At the start of the DIESEL project a comprehensive review and analysis of existing web based remote laboratories was carried out. From this review process a number of key deficiencies in existing remote experimentation laboratories were

identified. It was concluded that in contrast to traditional laboratories, web based remote access facilities were crude in nature with only a fraction of the functionality, accessibility and flexibility of their campus based counterparts, and failed to fully utilize existing hardware and software resources. To address these deficiencies, key features and functionality currently available in campus based laboratories were identified that would need to be replicated in remote access facilities to make the overall experience comparable. These features included facilitating full functional access and remote control of a diverse range of software and hardware resources. The complete comprehensive in-depth review is available in [8].

The DIESEL environment subsequently developed addressed all of the existing deficiencies identified in the review process and offers access to a comprehensive range of modern embedded system technologies and design tools. To facilitate distant access and control of these resources a generic architecture and access-control methodology for remote laboratories was developed which efficiently integrates instrumentation and experimental hardware components (Fig.1). The generic architecture consists of a gateway server (laboratory administrator) connected to the Internet and number of experimental workstations connected to the server via a network-hub.

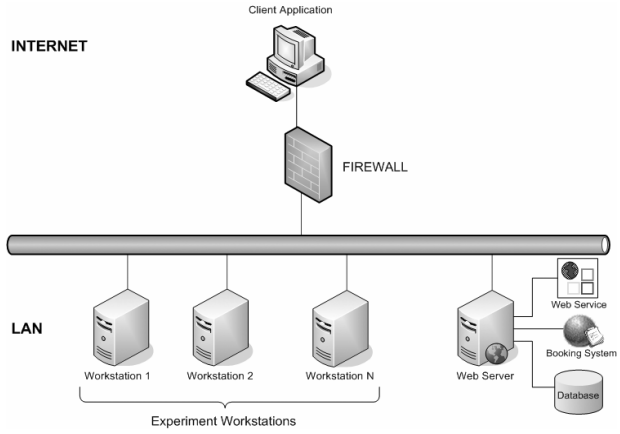

Figure 1. Architecture of remote access laboratory

This architecture is functionally composed of three interacting components; a server based booking system, accessible through the web which allows students to reserve a time slot on any available experimental workstation; a client application which the end user installs on their PC to facilitate access to remote experimentation resources; and a server application which runs on each remote workstation to facilitate the remote access process. The server application is accessed through the client allowing the user full access and control to all the functionality required to complete a laboratory session on a remote workstation.

Students accessing the remote laboratory initially connect to the gateway server which handles administration and authorisation duties and connects validated users to available experimental workstations. Each individual workstation is identical and hosts a range of experimental related hardware and software tools required to carry out practical experiments. Fig.2 illustrates the hardware components of a workstation which includes test instrumentation and a range of experimental boards. The test instruments are configured and controlled from the workstation using the GPIB protocol while the experimental boards are accessed from the workstation using either RS232 or parallel connections as required. A GPIB controlled switching matrix allows the test instrumentation to be connected to a number of test points on the experimental boards as necessary in the course of an experiment.

To enable these hardware and software resources to be remotely accessed and controlled the DIESEL software architecture was developed [8]. The DIESEL client-server approach uses a distributed software architecture developed using Microsoft .NET technology. The architecture for the remote access laboratory consists of a client application, a number of experimental workstations which host individual server applications, a web server which hosts a database, a web service and a web-based booking system.

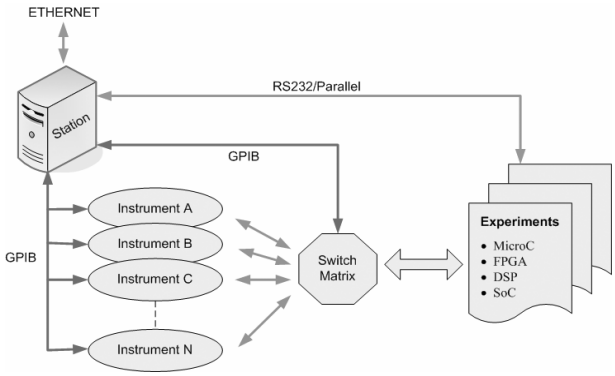

Figure 2. Generic DIESEL architecture

III. CLIENT-SERVER ARCHITECTURE FOR REMOTE EXPERIMENTATION

Fig.3 shows the communication and data flow between the various parts of the remote access lab. Communication between the client application and the server application on the experimental workstation uses peer-to-peer TCP communication and operates over a secure encrypted .NET Remoting channel using 256-Bit encryption. Interaction with the web service occurs over HTTP and messages and data are exchanged using the Simple Object Access Protocol (SOAP). This approach circumvents any problems that could arise with access through firewalls and avoids cross platform issues [9].

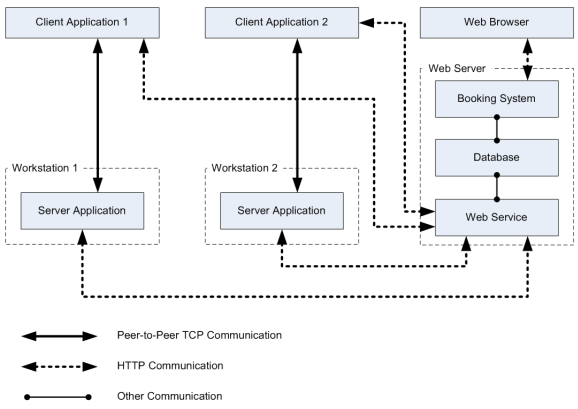

Figure 3. Communications protocol in DIESEL system

The system implements a four-tier communication model: the presentation layer, the data layer, the business layer and the physical layer (Fig. 4). The presentation layer consists of the DIESEL client application and the booking system which is accessed through a web browser. The DIESEL client provides the user interface which allows the user to configure and manipulate embedded systems and instruments remotely. The booking system provides the user interface for making and managing bookings. The data layer provides access to the database through either a web service or the web booking system. The business layer is implemented in the DIESEL server application, and provides access and control to the physical layer. The physical layer consists of all of the hardware resources (e.g. experimental boards). A web service is used as a gateway between the presentation, business and data layers to allow the client application and server application to access the database. This approach was preferred as it allows the separation of the client and server applications from the data storage process.

The server application (business layer) responds to commands from the client application by executing the appropriate control programs on the hardware architecture (physical layer) to configure the embedded circuits, signal routers and instruments while sending commands to the circuit under test. In this approach authenticated individual users complete experiments on remote workstations using a Peer-to-Peer client/server model [9].

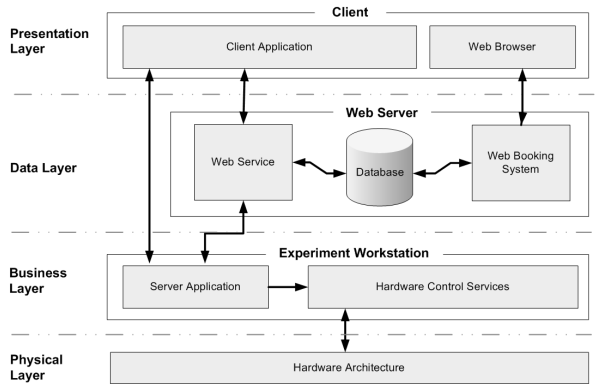

Figure 4. DIESEL four tier communication model

The only operational requirement for environment access and control for the remote student is the installation of a relatively small client application on their PC and a high speed (500kps+) Internet connection.

IV. COLLABORATIVE WORKING ARCHITECTURE

The initial objective of the DIESEL project was to recreate as accurately as possible the on-campus laboratory experience for the remotely based student [4]. The architecture and approach described previously goes some way to achieving this goal. However in an on-campus laboratory setting, students will typically work in groups to complete experiments, mentored and guided in this process by lecturers and support staff [5]. The ability to recreate this aspect of the on-campus laboratory experience was until recently generally unavailable in remote experimentation architectures. To address this deficiency the DIESEL architecture was redesigned and extended to facilitate synchronous remote collaborative working between geographically diverse users [10].

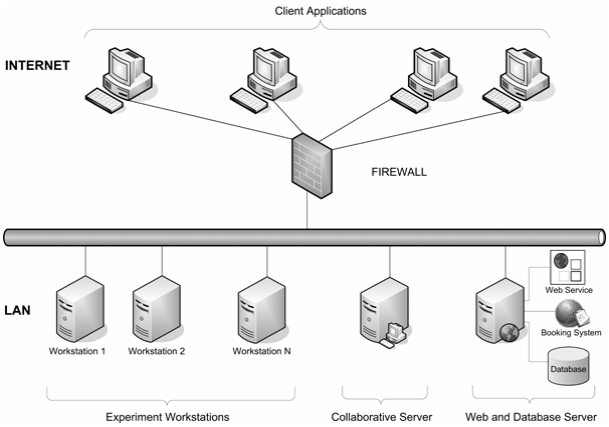

Figure 5. DIESEL collaborative working architecture

A collaborative working server was added and the existing components of the integrated learning environment were enhanced and augmented functionally to facilitate remote working between groups of students or students and lecturers (Fig. 5). In a practical sense, this added collaborative functionality allows remotely based users to work together on the same experiment hosted on the same remote workstation simultaneously, while accessing, viewing and controlling each component of the integrated learning environment e.g. virtual circuits, instrumentation, remote desktop and webcams together. Any changes made by one user to any of the components of the environment are immediately replicated to all users. This enhanced client/server architecture was implemented using the C# platform with .NET Remoting and using Web Services for secure bi-directional communications between the client and server (Fig. 6). To facilitate collaborative working the server application running on the remote workstation listens on a secure .NET Remoting channel for connections from remote users. When an instance of the client software connects to the workstation it attempts to retrieve references to a number of objects located in the workstation's server application.

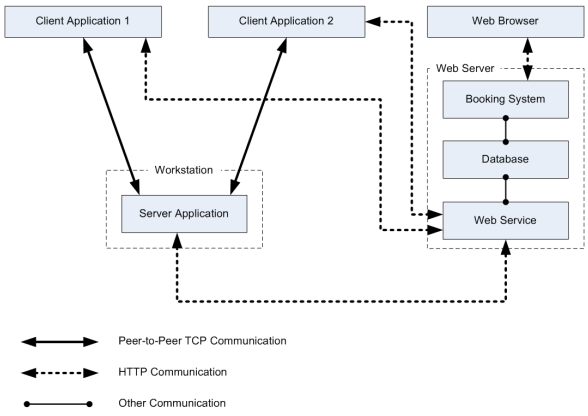

Figure 6. Communications protocol in DIESEL collaborative working architecture

The client application will then subscribe to events from the remote objects and through these events the server application will be able to communicate and synchronise with each instance of the client software. The collaborative version of the integrated learning environment uses a modified remote desktop facility

based on a Virtual Network Computing (VNC) client implemented using C#, and the Remote Framebuffer Protocol (RFB) to facilitate simultaneous multiple users working on a single workstation. An existing version of the VNC Server provides the server-side functionality.

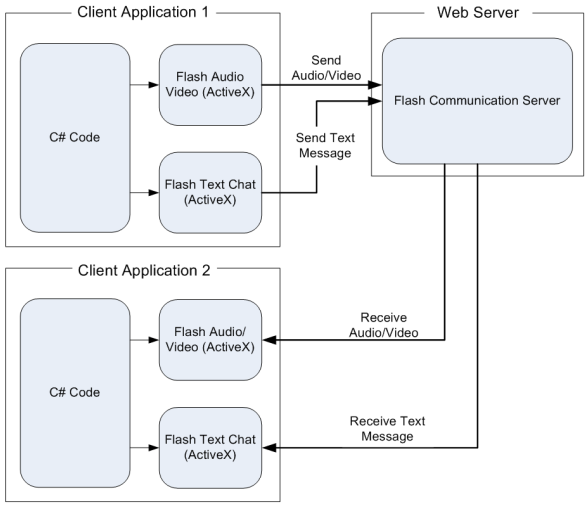

Figure 7. Communications and collaborative working environment

The Macromedia Flash Communication Server was used to provide video conferencing and instant messaging functionality. Visual and audio communication and instant text messaging between users is provided by a Flash ActiveX component embedded in the client application. The Flash component captures the user's webcam and microphone and streams it to a server running Macromedia Flash Communication Server. Flash Communication Server then streams the live audio/video feed to all of the other connected users as illustrated in Fig.7. A remote laboratory session can be reserved through the collaborative booking system. This system will allow a user to book a session and invite users to collaborate in that session. The booking system was developed using ASP.NET with SQL Server as the database. The booking system first requires the user to log in and authenticate. When the user books a session they can choose the date and time and also the student or member of staff they would like to work with (Fig. 8).

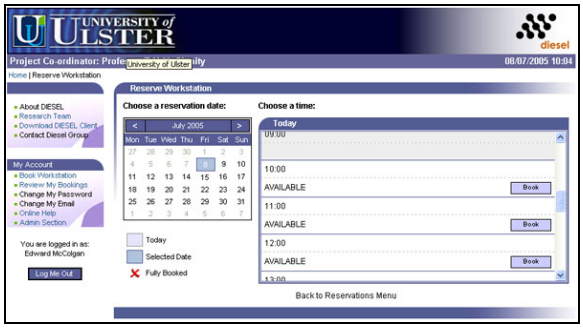

Figure 8. Choosing a booking date and time

After the user has selected their working partners the booking system will send an email to each user notifying them that they have been invited to collaborate on a remote lab session (Fig. 9). The email will provide a web link which the user must click to accept the invitation. When the invitation has been accepted all parties involved will receive notification that the collaborative working session has been booked. The users will be able to connect to the remote lab at the reserved time and collaborate with the other users sharing the session.

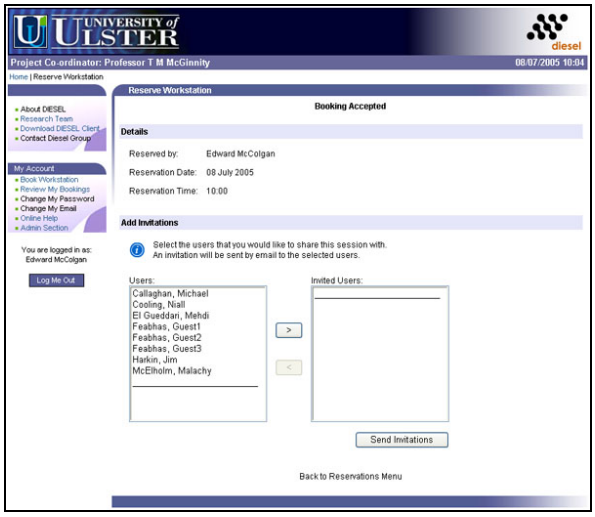

Figure 9. Inviting users to share remote lab sessions

Fig. 10 shows two students working collaboratively using video conferencing tools, remote desktop and a virtual circuit interface to write, compile and download a program to a microprocessor which executes a washing machine simulator program.

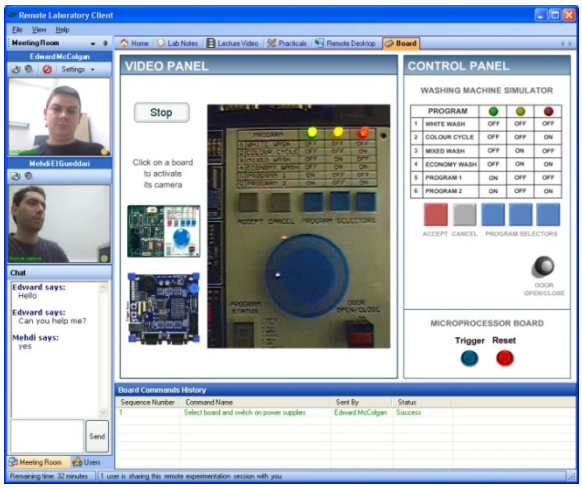

Figure 10. Collaborative working session in progress

V. LECTURER-LED EXPERIMENTATION

The lecturer-led system extends the collaborative working approach to allow educational institutions and training providers to provide remotely located students with access to campus based laboratory resources for remote experimentation augmented by live lectures and tutorials given by tutors [11,12]. The environment developed allows lecturers/trainers to create, manage, and deliver live lectures to a widely dispersed audience of students while allowing students to undertake real (non-simulated) practical exercises using real hardware and instrumentation, either individually or collaboratively. This advanced e-learning environment uses a distributed

architecture (Fig. 11) developed using Microsoft .NET technology and is comprised of four core software components including; the lecturer application, the student application, the experiment workstation server application, and the collaboration server application. In addition to these components there is a web server which hosts an online booking system, a web service and an SQL Server database. The remotely located laboratory hosts a number of experiment workstations similar to those described in the single user model each running the workstation server software and includes the collaborative server which manages intercommunication between clients (lecturers and students), and experimental workstations. These software components interact over a number of different communications protocols including HTTP communication and peer-to-peer TCP communication (Fig. 12). The HTTP communication is used by web browsers to access the online booking system, which allows lecturers and students to schedule access time in the remote laboratory. The HTTP communication is also used by the web service to provide the software components of the e-learning environment with access to the central database. The software components communicate with the web service by exchanging SOAP (Simple Object Access Protocol) messages to access various remote methods served by the service, where typically the web service is used to authenticate users and verify bookings. Peer-to-peer TCP communication is used for intercommunication between the software components i.e. student application, lecturer application, workstation server and collaboration server. The TCP communication operates over a secure, encrypted .NET Remoting channel and uses 256-bit encryption.

.NET Remoting is used to publish a number of server based objects, exposing them to remote processes. These published services can then be accessed by a client application. When a client application subscribes to a remote object it can access the methods, properties and events of that remote object. The collaboration server application manages interaction between experimental workstations and users. By publishing a number of .NET Remoting services the collaboration server allows the other software components to interact (Fig. 13). The collaboration server provides a workstation management service which allows each experimental workstation to register.

Through the workstation management service the collaboration server allows students or lecturers to gain access to multiple workstations simultaneously to either provide or receive assistance and work collaboratively. The user management service maintains a list of connected students and their current activities. This functionality allows lecturers to monitor connected students. The communication service provides a central point for text chat or audio/video communication between users allowing users to communicate. The collaboration server is responsible for broadcasting live lectures from a lecturer application to all students' clients. Through these services the workstation server application allows connected users to control the remote experiment hardware. Each remote workstation also hosts remote desktop server software which allows students and lecturers to access and control the desktop of the experiment workstation.

A. Lecturer Application

The lecturer client application (Fig.14), when installed on any PC allows a lecturer to create, manage and deliver live online courses from any remote location. The installation is a straightforward once off install requiring little end user configuration and a download of less than 5 megabytes dependant on the software configuration of the client PC (system operation requires the .Net framework). Once installed and connected the lecturer can deliver a live presentation to a group of connected students, take part in live discussions and provide technical demonstrations through the shared remote desktop and interactive whiteboard facility. The lecturer can monitor students as they work and provide assistance if required. In addition the lecturer can form/arrange remotely located students into groups for practical work providing individual or group support during these practical exercises facilitating an advanced level of collaborative working between diversely located students.

B. Course Administration

Before a course can commence the lecturer must create it through the administration panel of the lecturer application. Here the lecturer can view and edit current user details adding or removing new or existing users as required. Lecturers can also create new courses, upload new course material and edit existing courses as well as manage course schedules and arrange timetables for live presentations.

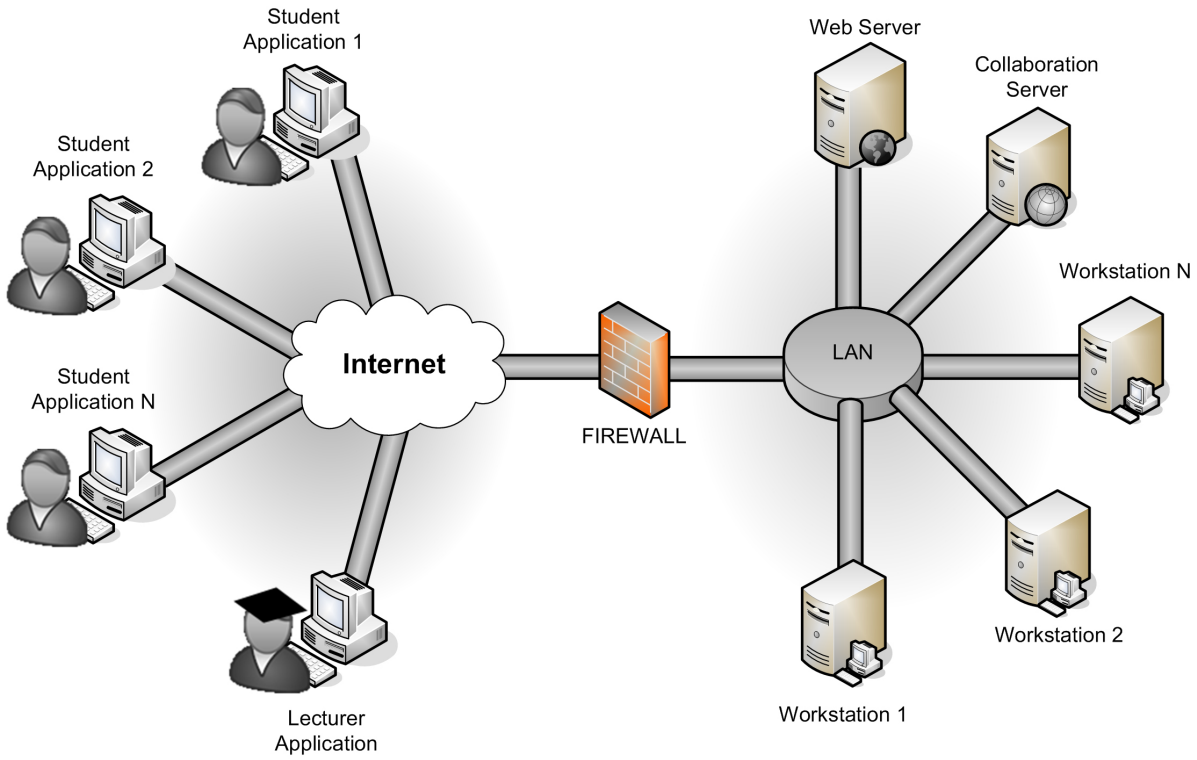

Figure 11. e-learning environment distributed architecture

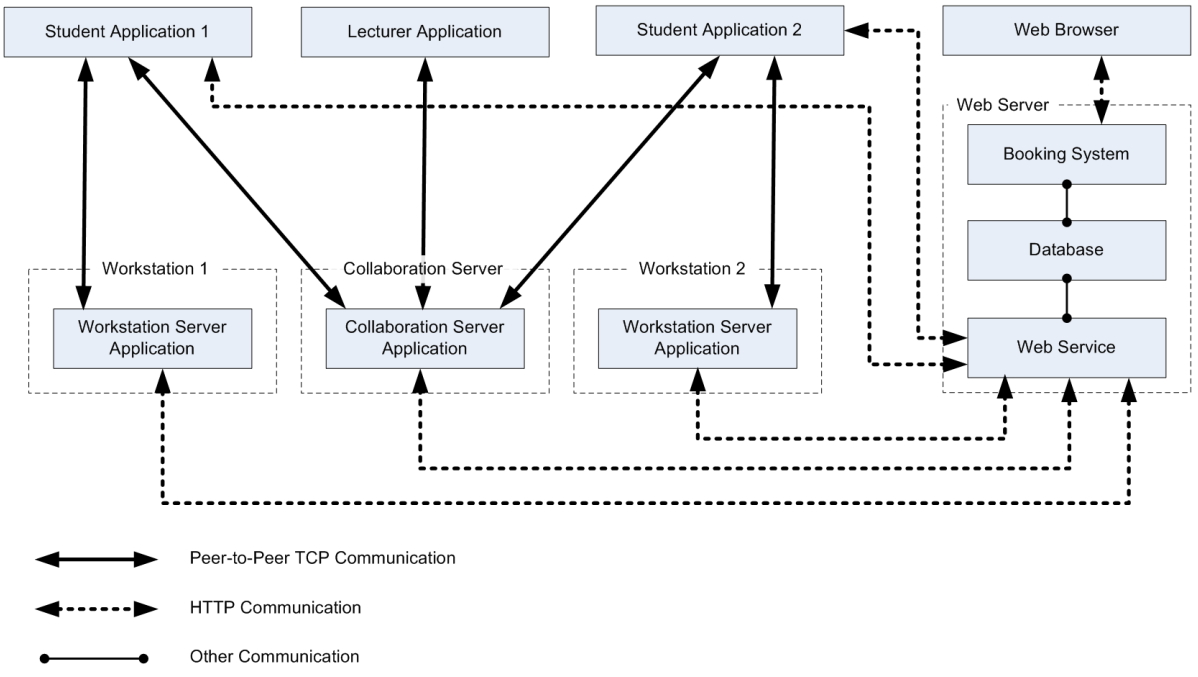

Figure 12. Communications structure

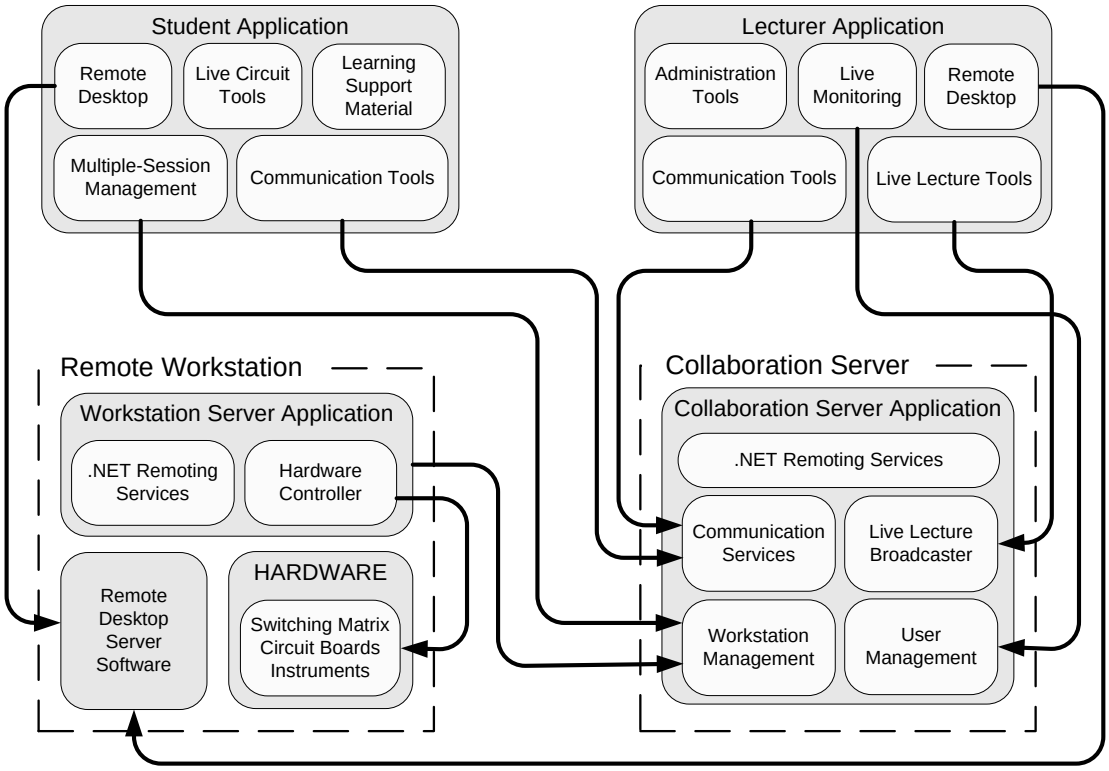

Figure 13. Communication structure: Interaction between applications

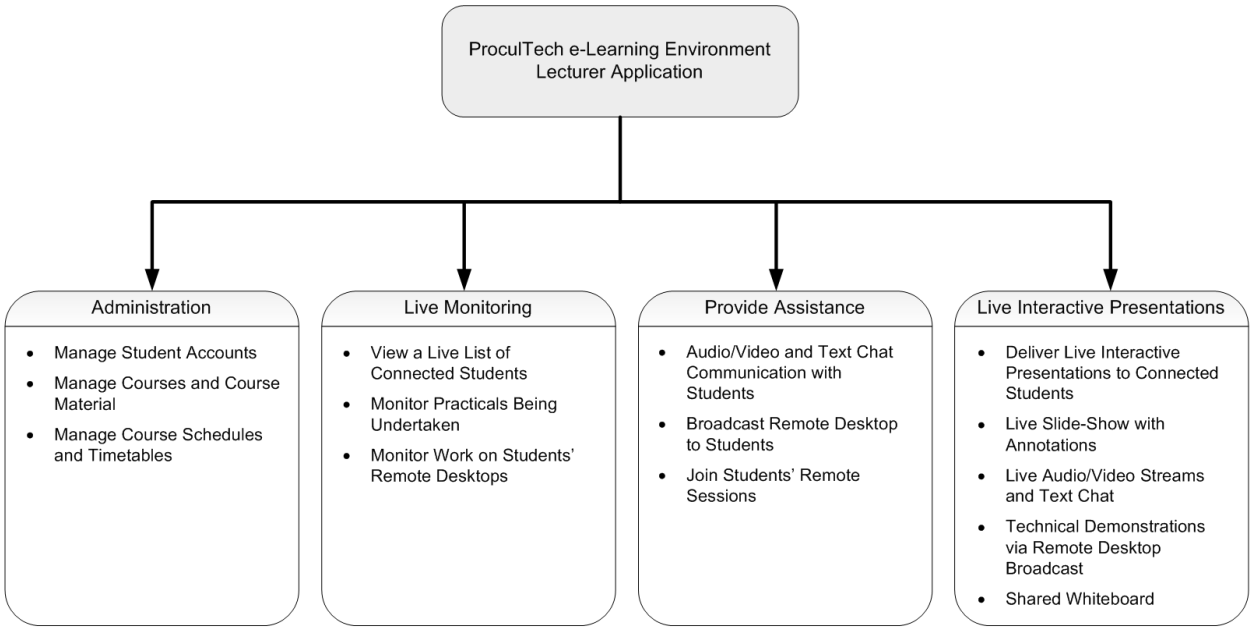

Figure 14. Communications structure of the lecturer application

C. *Live lecture and demonstrating*

At pre-scheduled times the lecturer can conduct a live lecture with accompanying hardware/software demonstrations. The live lecture can consist of a live presentation delivered by the lecturer followed by supported/monitored individual and group practical work. A range of tools are available to the lecturer including webcams, text chat, a presentation/slide-show, a shared remote desktop, and a shared whiteboard. In addition the lecturer can demonstrate and operate a real hardware training board or instrumentation. At scheduled lecture times the students connect to the remote learning environment using the student client. Similarly the lecturer connects to the remote learning environment using the lecturer client.

At initialisation both clients subscribe to the collaborative server which facilitates the communication elements of the environment. When the lecture starts the live presentation it automatically appears on all of the students' client interfaces and includes dynamic content from the lecturer's remote desktop, training board, instrumentation, whiteboard and the lecturer's audio/visual feed. Lecturers can annotate slides which will then be replicated on the student's client interface. The remote desktop element of the environment allows the lecturer to broadcast live demonstrations of software packages/tools and other applications hosted on the remote workstations.

At any time during a session a student may request individual assistance. The lecturer can facilitate this by temporarily suspending the live presentation and connecting directly to the student's experiment workstation to provide individual assistance. When the live element of the session is complete the students can carry out the practical elements working either individually or in groups. At any stage during the practical sessions the student/s can request assistance from the lecturer. The lecturer can take control of the students' client and demonstrate the software or configure hardware as needed.

D. *Student Application*

In addition to watching live lectures, the student client provides the student with all the functionality required to carry out practical remote experiments and to work collaboratively with fellow students. These include videoconferencing and text facilities, virtual instrumentation tools and remote desktop features. The student client has three modes of operation *normal working* mode, *live lecture* mode, and *help request* mode.

During *normal working* mode the client application allows students to schedule and access remotely based hardware and software resources to undertake practical experimentation either individually or collaboratively. This requires the student to book remote experimentation sessions in advance using the central booking system. *Normal working* mode does not require the lecturer to be connected to the learning environment. In *live lecture* mode the student client application will display the content being broadcast by the lecturer while participating in the live session.

In this mode, the lecturer controls what the users see on their screens. The final mode of operation of the student application is the *help request* mode. In this mode the remote students can request assistance from any other connected student or the lecturer. When assistance is granted the lecturer or helping student shares control of the student's client application and can then demonstrate the solution operating all aspects of the system as needed e.g. virtual instrumentation and circuits.

Fig.15 gives an overview of the initialisation process and shows the sequence of events involved in the *help request* process. As each client application (student and lecturer) connects to the e-learning environment they register with the collaborative server, subscribing to remote events and retrieving remote object references [stages 1-4]. The collaborative server maintains a dynamic list of currently connected students and lecturers which is visible to all users of the system.

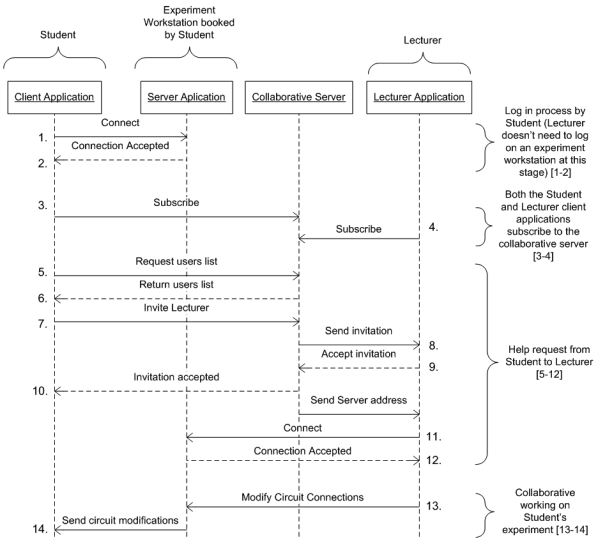

Figure 15. Software interaction during a help request between a student and the lecturer

Any user can initialise the process to start the collaborative assistance session. A student requiring assistance will request a list of active users from the collaborative server [stages 5-6]. When the student selects another fellow student to be the assistant, an invitation is sent to that user's client application and a prompt is displayed informing the requested user that their assistance has being sought [stages 7-8]. If the help request is accepted the user's client application will connect to the experiment workstation of the user who requested help [stages 9-12]. The assistance provider can temporarily access and control the workstation of the student in need for the duration of the help session [stages 13-14].

Fig.16 shows the lecturer client components which include webcams, text chat, a presentation/slide-show, a shared remote desktop, and a shared whiteboard. Fig.17 shows a multi-user lecture-led session in progress with the lecturer broadcasting and presenting live to a number of students.

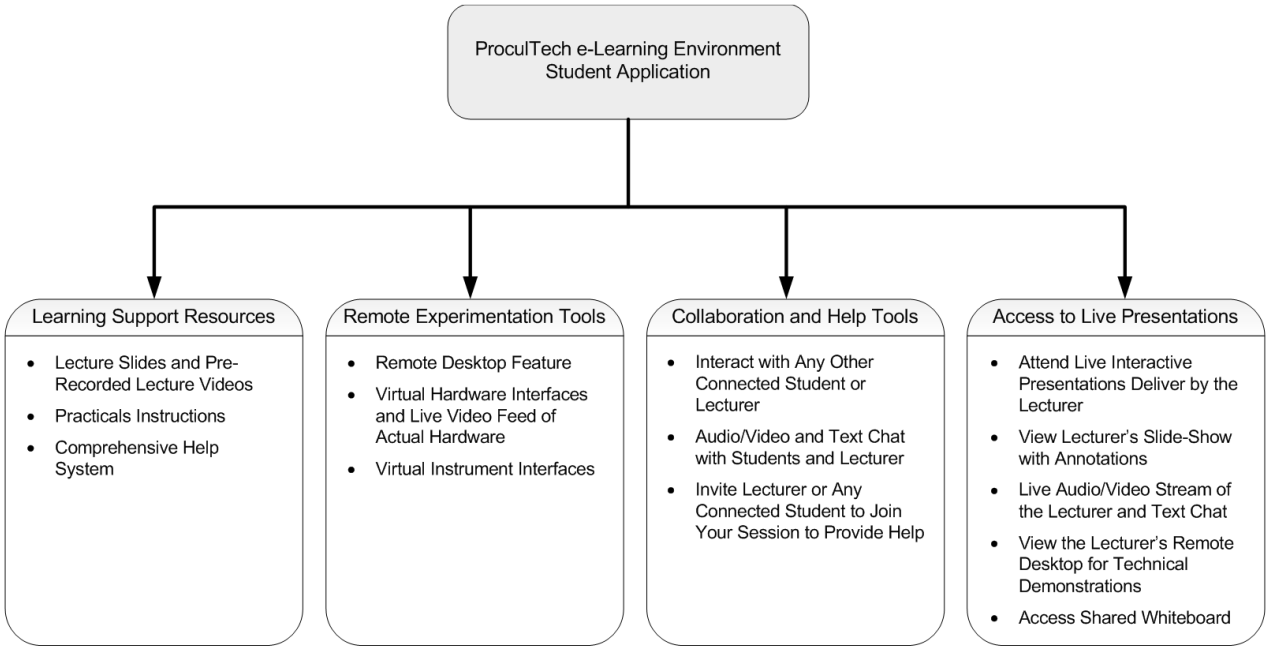

Figure 16. Overview of the student application features

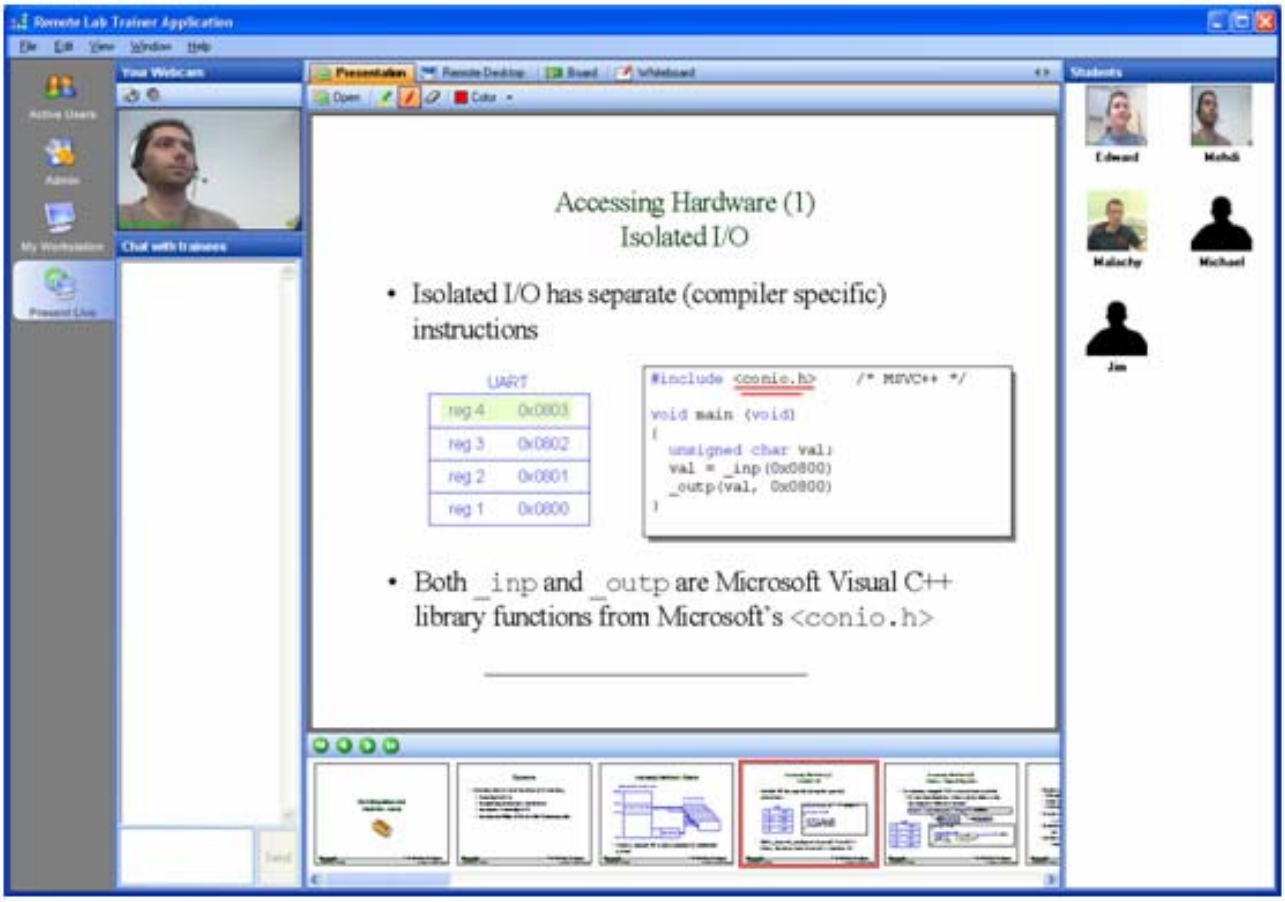

Figure 17. Live presentation using the lecturer client

V. DISCUSSION AND CONCLUSION

This paper presented several integrated learning environments for remote experimentation laboratories which encompassed single, collaborative and lecturer-led approaches. Single user environments are now commonplace and extensively used both in education, training and industry. As high speed internet becomes increasingly commonplace the demand for functionality that includes collaborative working and tutor led systems will increase. The lecturer-led environment presented in this paper allows the presentation of live lectures and demonstrations to diversely located students and offers remote assistance and individual tuition if required. In addition, students can complete complex practical laboratory exercises either individually or collaboratively with the majority of the functionality currently offered by the campus experience. The approach offered here affords a number of critical benefits allowing remotely located students to attend live lecturers and complete laboratory assignments unconstrained by time or geographical considerations. This integrated learning environment, while initially developed for educational use, has great potential for use in the continuing professional development market. Future research in this area will concentrate on extending the range of experiments available in the system and on investigating the issue of implementing an automated help system for user support in complex remote experimentation environments.

In a wider research context, 3D virtual collaborative and immersive environments are becoming mainstream and increasingly educational institutions are investigating the use of persistent virtual worlds for experiential and practical based learning. Recent conferences on Second Life [13] highlight the level of educational interest in this domain and already companies including IBM are investigating the integration of real world data with instrumentation in the context of virtual worlds [14]. Clearly there will be future opportunities in this emerging area for remote education and distance learning.

REFERENCES

[1] Gillet D. et al., (2000), "Advances in Remote Experimentation", 19th American Control Conference, ACC'2000, 20 -25

[2] Shen H. et al., (1999), "Conducting Laboratory Experiments over the Internet", IEEE Trans. on Education, 30, 191-199

[3] Beetner, D., Pottinger, H., Mitchell, K. (2000) 'Laboratories Teaching Concepts in Microcontrollers and Hardware-software Co-design', 30th Annual Frontiers in Education Conference, Vol.2, S1/1-5

[4] Callaghan M.J, Harkin J, McGinnity T.M, Maguire L.P, (2006) "Client-Server Architecture for Remote Experimentation for Embedded Systems", International Journal of Online Engineering (iJOE) , Vol. 2, No. 4, Kassel University Press, ISSN 1861-2121

[5] Callaghan, M.J, Harkin, J, McGinnity, T.M, Maguire, L.P, (2002): "Internet-based Methodology for Remotely Accessed Embedded Systems", IEEE Conf. Systems, Man and Cybernetics, 157-162

[6] Callaghan, M.J, Harkin, J, McGinnity, T.M, Maguire, L.P (2003): "Collaborative Environment For Remote Experimentation", Intl. Conf. Microelectronic systems Education 1st-2nd June 2003 Anaheim, California, 157 -162

[7] Esche, S. A (2002): "Scalable System Architecture for Remote Experimentation. Proc. 32nd ASEE/IEEE Education Conf. Boston, Mass., USA, Nov. 6th - 9th

[8] Callaghan, M.J., Harkin, J., McGinnity, T.M., Maguire, L.P. (2003) 'Integrated Architecture for Remote Experimentation', IEEE International Conference on Systems, Man and Cybernetics, pp. 4822-4827

[9] Harkin, J., Callaghan, M.J., McGinnity, T.M., Maguire, L.P, (2005) 'Intelligent User-Support in Learning Environments for Remote Experimentation', IEEE International Conference on Information Technology and Applications, Sydney, Australia, pp. 203-209

[10] Callaghan MJ, Harkin J, McColgan E, McGinnity TM, Maguire LP, (2007) "Client-server architecture for collaborative remote experimentation", Special Issue of the Journal of Network and Computer Applications, Vol. 30, No. 1, Elsevier, ISSN 1084-8045, Pages 1295-1308

[11] Amaratunga, K., Sudarshan, Raghunathan (2002) "A Virtual Laboratory for Real-Time Monitoring of Civil Engineering Infrastructure," International Conference on Engineering Education, Manchester, UK.

[12] Wolf, W., Madsen J. (2000) 'Embedded Systems Education for the Future', Proceedings of the IEEE, Vol.88, No.1, pp. 23-30

[13] http://www.secondlife.com

[14] Proceedings of the Second Life Education Workshop, Second Life Community Convention, Slccedu2007 San Francisco, California, August 18-20, 2007

AUTHORS

Michael Callaghan is a Lecturer in the School of Computing and Intelligent Systems at the University of Ulster. He is a member of Intelligent Systems Research Center within the University of Ulster. His current research interests relate to the Remote Experimentation and Hybrid Intelligent Systems. (e-mail: mj.callaghan@ulster.ac.uk)

Jim Harkin is a Lecturer in the School of Computing and Intelligent Systems at the University of Ulster. He holds a Bachelors, MSc and PhD in Electronic Engineering from the University of Ulster. He is a member of the Intelligent Systems Research Center within the University of Ulster and his current research interests relate to the design and implementation of intelligent reconfigurable embedded systems. (e-mail: jg.harkin@ulster.ac.uk)

Martin McGinnity is Professor of Intelligent Systems Engineering at the University of Ulster. He holds a first class honours degree in physics, and a doctorate from the University of Durham. He is a Fellow of the IEE, member of the IEEE, and a Chartered Engineer and leads the research activities of the Intelligent Systems Research Center. His research interests relate to the creation of intelligent computational systems and the area of intelligent systems in general. (e-mail: tm.mcginnity@ulster.ac.uk)

Liam Maguire is a Professor of Computational Intelligence and Head of the School of Computing and Intelligent Systems, University of Ulster. He obtained a MEng (distinction) and a PhD in Electrical and Electronic Engineering from the Queen's University of Belfast. His current research interests relate to the creation, design and implementation of intelligent systems. (e-mail: lp.maguire@ulster.ac.uk).

Manuscript received 11 November 2007. Published as submitted by the authors.