Accessibility and usability in complex web based learning applications: lessons from the PEARL project

Martyn Cooper, Chetz Colwell
Accessible Educational Media (AEM) group
Institute of Educational Technology
Open University
United Kingdom
m.cooper@open.ac.uk, c.colwell@open.ac.uk

Telmo Amaral
Departamento de Engenharia Electrotécnica e de Computadores (DEEC)
Faculdade de Engenharia da Universidade do Porto
Portugal
tga@fe.up.pt

Abstract: This paper reports approaches promoting accessibility for disabled students to science and engineering practical work, developed in a major European educational technology project PEARL. It has developed the facilities for conducting a range of teaching experiments in science and engineering subjects at higher education level remotely over the WWW. One of the initial rationales for the project was to facilitate increased access for disabled students to practical work in these disciplines. This paper concentrates on one of these practicals based on the remote control of an optical spectrometer. It briefly describes the overall systems architecture and then the user interface sub-system. The approaches adopted to promote accessibility for disabled students at both systems and interface levels are outlined. It reports the results of interim expert evaluations and user trials with this prototype and how these have informed the subsequent development work. Lessons from this work are generalisable to other Web based applications, particularly with JAVA based client applications or applets.

Introduction

The PEARL project (Practical Experimentation by Accessible Remote Learning) started in March 2000. Cooper, et. al. (June 2000) introduced the project in detail; only a brief introduction is included here to set the paper in context. The PEARL project objectives are:

- To develop a flexible system enabling students to conduct real-world experiments remotely over the WWW.
- To research the pedagogic impact of this approach validating its developments in different educational contexts and subject areas in higher education.

Key results of project are four instantiations of the PEARL approach, one at each of the participating universities, providing teaching experiments in the following subject areas:

- Foundation level science (Open University)
- Cell biology (University of Dundee)
- Manufacturing engineering (Trinity College, Dublin)
- Electronic engineering (Faculty of Engineering, University of Porto)

These are realized through a systems integration of: a remote controlled lab facility, a control and communications infrastructure working over the Internet, an accessible user interface sub-system and synchronous collaboration tools. A major deliverable planned is a PEARL Handbook designed to enable other institutions to adopt the PEARL approach in their teaching.

PEARL, from its conception, was seen to provide significant potential for enabling greater access for disabled students to practical work in science and engineering subjects. This set the project team a difficult

challenge, particularly in the user interface design and the design of infrastructure that enables the user interfaces to communicate with the remote laboratory. It is this aspect of the project's work that is the focus of this paper. The paper describes in particular the instantiation of the PEARL approach developed for the Open University. It is around this prototype that the accessibility work in the project has been concentrated.

PEARL is a major project with a total effort over the 3 years in excess of 30 person years and a budget of about $2 million.

The accessibility rationale

The PEARL project seeks to make experimental work in science and engineering accessible to students with a wide range of disabilities. Disabled students are grossly underrepresented in these disciplines. The reasons for this are complex and include significant factors from primary and secondary level education. However access to experimental work is cited as a key barrier in analysis of this problem both in North America and Europe. Much work has been done over the past 20 years to make computers accessible to people with all kinds of disabilities and a high degree of success has been achieved. Thus for most people with a disability, seeking to participate in higher education, access technology for a computer, where needed, is commercially available. Hence the approach in the PEARL project has been to firstly to make practical work computer controlled. It then had to ensure that the computer software and interface design followed well-established design for accessibility principles, as exemplified by the WAI Content Guidelines [URL 1] or various manufacturers' guidelines [URL 2, URL 3] and that the software was compatible with the available access technologies.

The system developments

This section presents an overview of the PEARL system focusing on the realisation of the OU's Spectrometer Experiment. **Figure 1** shows a schematic of the architecture of a remotely accessible spectrometer. The Server application implements the high-level control operations that can be requested remotely. These high-level operations are translated to calls to low-level control methods exposed by the Hardware Access application. The low-level methods are invoked across a local CORBA bridge. The Server and the Video Transmitters interact, to allow the synchronisation between the execution of control operations and the transmission of video, when necessary. This interaction is also achieved via local CORBA bridges. The Hardware Access application embeds the hardware-access ActiveX control and exposes its low-level control methods to the Server via CORBA. These low-level methods access the control routines that run inside the central processing unit (CPU) of a motion controller board [model DMC-18x2] from Galil Motion Control, Inc. installed in a PC. These routines control the actual jig's hardware, which consists of motors, sensors and valves, via the motor drive circuitry. The Hardware Access can run on a different computer from the Server.

The systems architecture

The Video Transmitter application, based on the JMF package, can be instantiated once for each available Video Camera. One of the Video Cameras captures an overall image of the spectrometer apparatus; a second camera is attached to the spectrometer's telescope and a third camera captures an image of the Vernier scale that allows the position of the telescope to be measured. A video stream is transmitted to each connected Client. The video streaming uses the Real-Time Transport and the Real-Time Control protocols (RTP and RTCP). Each instance of the Video Transmitter can if desired run on a different computer from the Server.

All the CORBA bridges were based on version 4.0.5 of ORBacus for Java and ORBacus for C++ implementations of the CORBA standard, from IONA Technologies, Inc. The Client, Server and Video Transmitter applications are being developed in the Java programming language, in the Java 2 Platform, Standard Edition (J2SE) from Sun Microsystems, Inc. The Hardware access application is being developed in the C++ and Microsoft Visual Basic languages.

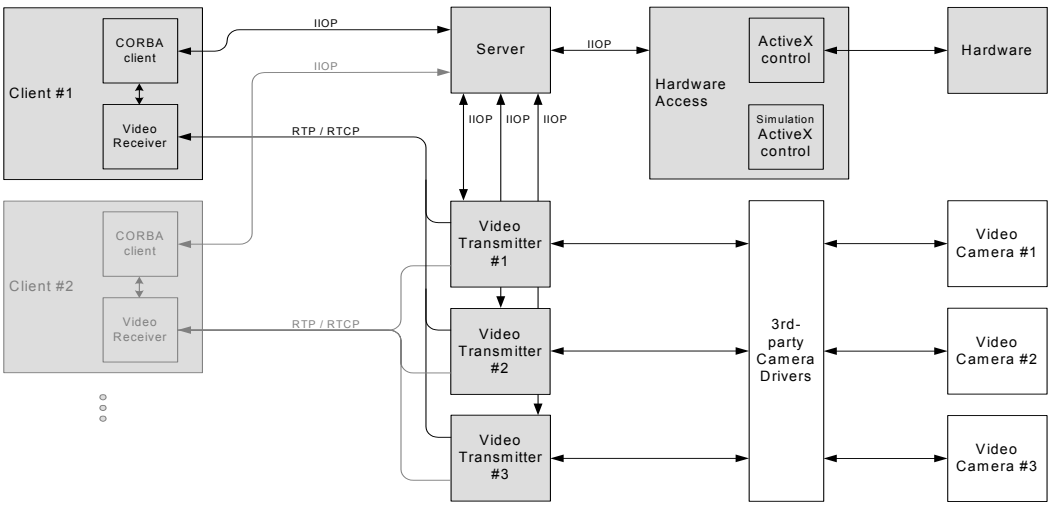

Figure 1: A schematic of the system architecture of remotely accessible spectrometer

The overall approach to the systems architecture has been a modular one so that the different modules can be readily adapted when creating a new experimental facility. The whole CORBA based architecture means that these modules can be mapped onto physical machines in a flexible fashion as dictated by other design constraints.

The Remotely Controllable Jig

The photo in

Figure 2 shows the experiment jig developed for the Open University by project partners Zenon S.A. of Athens. A simple optical spectrometer as used by Foundation Level Science students at the university has been adapted so that it is motor controlled. The spectrometer has been mounted on a moving table so that the students can position it in front of different light sources. As configured in

Figure 2, these are a Sodium (*Na*) Lamp, which is used to give a reference spectrum to calibrate the spectrometer and a Bunsen burner. Air is bubbled through one of up to 6 different metal salt solutions, as selected by the students, and into the gas feed so the Bunsen burner. Therefore the spectra associated with the chemical in each bubbler can be observed. The slit size and focus of the collimator, used to ensure the light is parallel as it passes into the diffraction grating, have been motorised, as has the rotation of the telescope.

The driver circuitry for all the motors, valves and sensors is located together in a dedicated housing. This in turn is interfaced to the Motion Controller Board installed in the same PC as providing the Hardware Access indicated in **Figure 1**.

User Interface Approach

This section presents a detailed description of how the user interfaces of the client application are realised. The objective here is to mediate between the high level commands offered on the user interface to the low level commands required by the jig's control circuitry and to do so across the Internet. The design has been conceived in such a way as to promote usability and accessibility and so that it is readily adapted when creating a new PEARL experimental facility.

When the Client starts, the XML Parser object uses the DTD Schema file to verify that the XML Instance file corresponds to a valid description of the GUI. The XML Parser is based on Xerces v1.4.1 from The Apache Software Foundation. The custom XML Content Handler object then uses the description contained in the XML Instance to actually build the GUI, that is, to create the necessary components and lay them out on the

screen. During the build process, other resource files are accessed, containing the images, accessible names, tool-tips and keyboard shortcuts to be used on the buttons placed on the Controls Tabbed Pane.

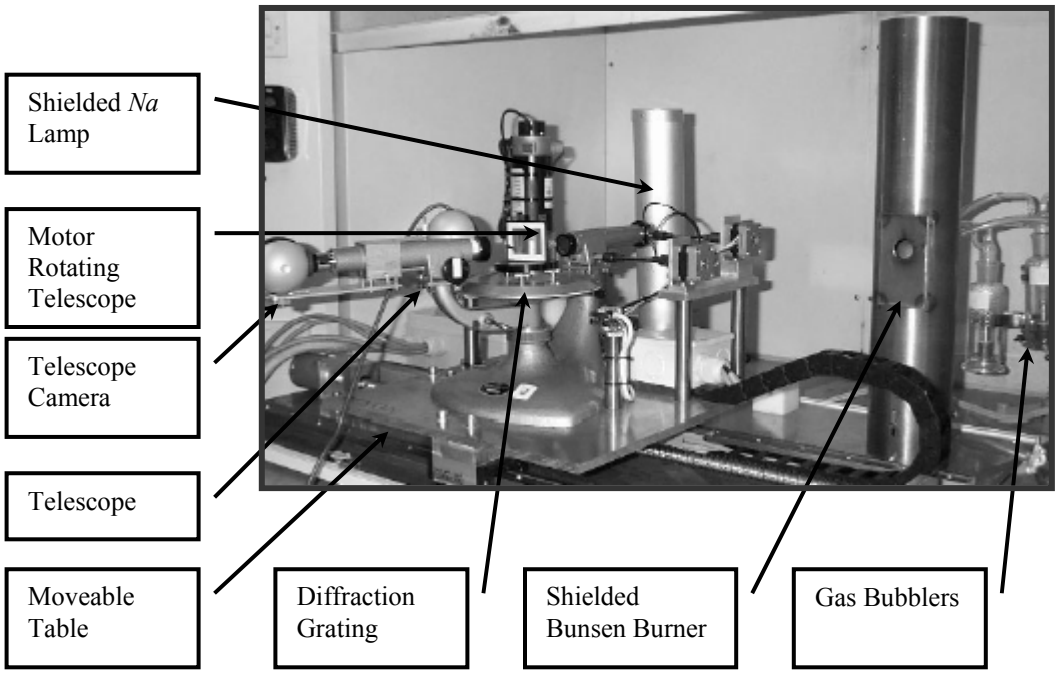

Figure 2: Annotated Photograph of Motorised Optical Spectrometer Jig

In addition, during the build process a CORBA Client object is created, which connects to the CORBA Server across the Internet. The buttons on the GUI are set to trigger remote requests for high-level commands exposed by the CORBA Server. The established CORBA connection is a bi-directional one, so the Server is able to send information back to the Client not only as the result of requested commands but also asynchronously in response to events in the remote lab.

A special method exposed by the CORBA Server retrieves the computer names and port numbers on which the Video Transmitter instances run. The XML Content Handler then creates the Video Receiver object, which connects to the Video Transmitters across the Internet. Once the video streaming connections are established, the live video feed components are placed on the Video Feeds panel and the construction of the GUI is complete. The Client needs to know only the computer name and port number of the CORBA Server in order to establish all the necessary video connections. Thus, the server-side arrangement of computers and port numbers used for the Video Transmitter instances is flexible.

The Client can run either as an application or as an applet within an Internet browser. In fact, to provide support for both formats, only very small portions of the code had to be written differently. **Figure 3** shows the client running as an applet within Microsoft Internet Explorer. Moreover, a mechanism is implemented to allow the client to run on a computer having a private name / IP address within a Local Area Network (LAN), as long as the LAN's firewall policy allow the establishment of client TCP and UDP connections to the Internet.

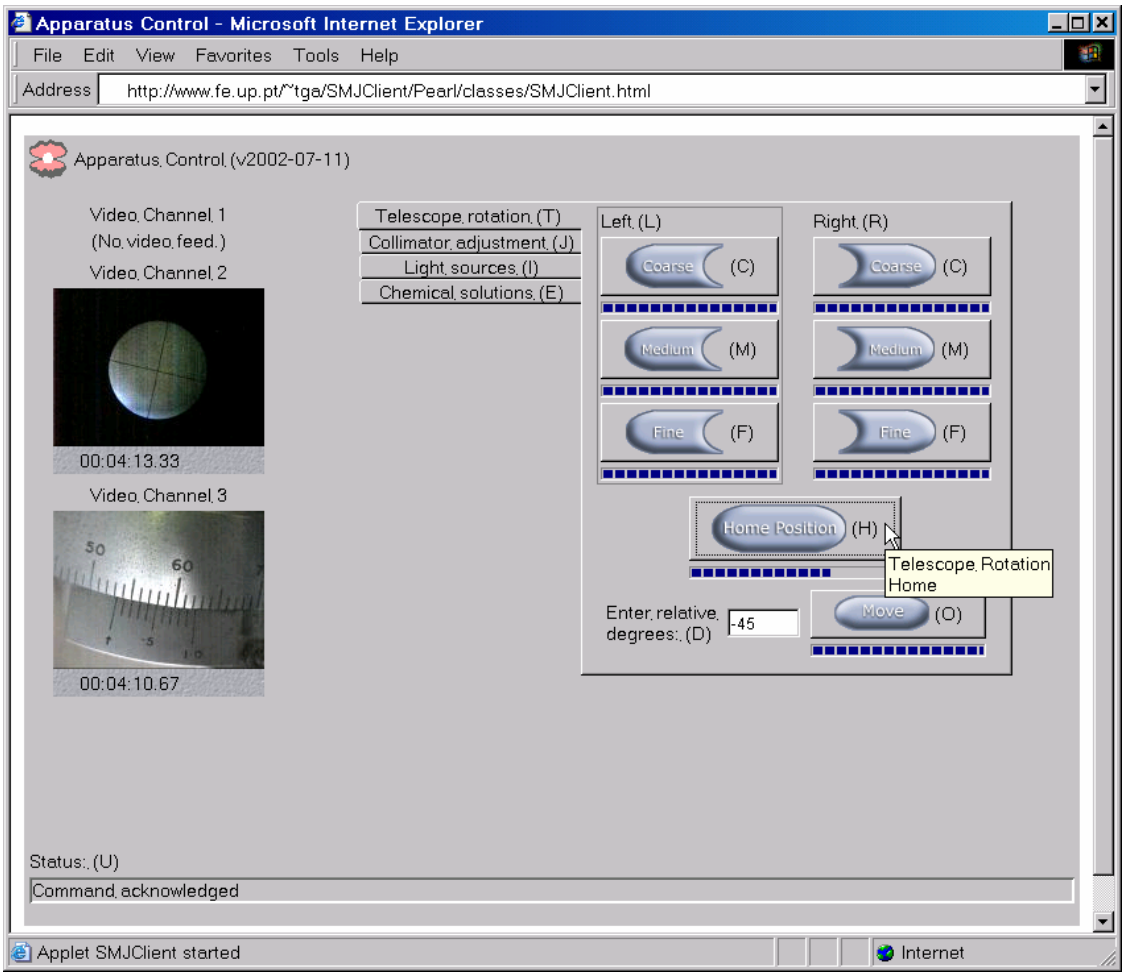

Figure 3 – Screen-capture of the client user interface, running as an applet within Internet Explorer, using Windows Classic display appearance scheme (only 2 video feeds shown)

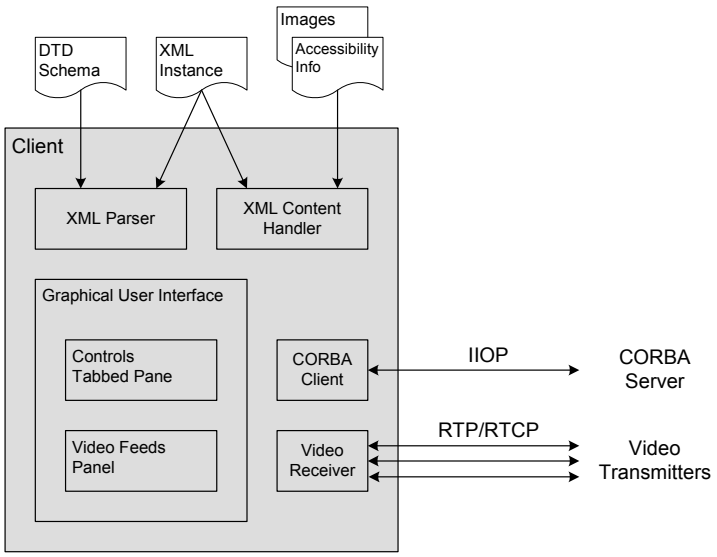

Figure 4 – Internal schematic of the client application / applet

As shown in **Figure 3**, the graphical user interface (GUI) of the client application / applet consists of a panel on the left containing the video feeds and a panel on the right (a tabbed pane) containing all the necessary controls.

The schematic in **Figure 4** shows the internal components of the client, along with the input files that are relevant for the construction of the GUI.

Approaches to Achieving Usability and Accessibility

Usability requirements and design features

There are several complex issues that need to be addressed in the design of a user interface for remote control of apparatus. The main issue is the impact of delays between the user's operation of a control, and the execution of that command in the laboratory, and the feedback, via the video feeds, that the command has been completed. The design therefore includes buttons, rather than knobs or sliders, to support students in making discrete commands. During the task, students may need to operate any one of up to 20 controls at any time, and therefore as many controls as possible need to be easily accessible. The design therefore groups the controls for different aspects of the spectrometer into 'tabs' (similar to Windows property sheets). The users of this software will not have much time to learn to use it, and may only use it once to complete the task. They will be provided with some instructions, but it is essential that the user interface 'affords' the purpose and operation of the controls. Therefore the design includes several graphical images (mostly in the form of arrows and grouping lines) to provide this affordance.

Accessibility requirements and solutions

The design incorporates several requirements in order to make it accessible to students with disabilities. These are presented below in bold text, along with a description of how they are met in the design. The requirements were based on guidelines for the accessibility of Web content [ibid. URL 1] and the expertise of the researchers and their colleagues.

The below table summarises the key accessibility criteria addressed within the projects developments and links this with notes on how these were implemented in the JAVA based client programmes. In most cases these are generalisable to any interactive application implementing its client software in JAVA.

Accessibility Criteria	**Implementation Notes**
Inheritance of Windows properties:	
Some disabled users require or prefer specific settings within the operating system in order to use the computer. For example, people may require certain colour combinations, font style and size, or to use utilities such as StickyKeys[1].	The client's GUI is based on JFC/Swing components from J2SE v1.4.0, which can track user's display preferences in Windows, such as colours, fonts, metrics, etc, and can also respond to dynamic changes made by the user. In addition, The GUI is based on layout managers, so that, when font sizes change, component sizes throughout the entire component tree may be easily updated. **Figure 3** shows the client tracking the Windows Classic display appearance scheme.
Text labels on all controls:	
The screen-reading software used by visually impaired users for speech output requires all user interface elements to have text labels which can be read out.	All controls have their Java accessible names set in the code, either implicitly or explicitly. In addition, the Java Runtime Environment (JRE) used to run the client (either as an application or as an applet) is configured to use version 1.0.2 of the Java Access Bridge for Windows. Thus, JAWS for Windows v4.0.2 is able to read the accessible names of all controls. All the controls have associated tool-tips, which implicitly set the control's accessible description. Tool-tips pop-up when the mouse pointer hangs over the control, as shown in **Figure 3.**
Keyboard operable:	
Many disabled users cannot or prefer not to use a mouse and therefore need to be able to	The order in which the controls can be focused using the Tab key is set by the way in which the controls are laid-out on the

[1] See details of this and other such utilities in the Windows 'Accessibility options' control panel.

operate all user interface elements via the keyboard. The PEARL application provides two methods of keyboard operation. The Tab and Enter keys respectively can be used to navigate between, and operate, the controls. In addition, keyboard shortcuts enable the user to operate all controls quickly by reducing the need to navigate between controls.	GUI. The Enter key is bound to the spacebar, so that either one or the other may be used to activate the control that has the input focus. In addition, for each control, a keyboard shortcut is associated (via the control's input and action maps) to a global action object. When a shortcut is used, the global action object determines which action should be taken and triggers that action (e.g. a button is clicked, or a tab on the tabbed pane is selected). Certain groups of buttons are allowed to share the same shortcut keys, given that the groups have themselves an associated shortcut (e.g. the Left and Right groups of buttons in **Figure 3**). Activation of controls using keyboard shortcuts is accompanied by transference of the input focus, so that controls activated by shortcuts may have their accessible names read by JAWS. Standard mnemonics are not used, because the need for a modifier key is not desired.
Reminders of shortcuts included in labels:	
Users are supported in learning the keyboard shortcuts with the inclusion of the shortcut in the label of the control. For example, one button is labelled "Focus coarse left. C." to indicate that the shortcut for this button is C.	The accessible names of all controls include the corresponding shortcut key character, so JAWS may read it.
Auditory indication of command progress:	
To provide visually impaired users with feedback on the progress of commands, on-speech sounds are provided to acknowledge the system's receipt of a command, that it is being executed, and that it is complete. This information is also provided visually for other users in the form of progress bars located beneath each control.	Short sounds accompany the changes in state of the progress bars located below each button. The simple support for sound of the Java core APIs is used to read and emit the sounds from files in WAV format. The CORBA connection between the client and the server is bi-directional, so that the intermediate state of a command ('command acknowledged') may be triggered by an asynchronous CORBA call-back.
Status bar:	
The status information provided in the progress bars is also presented in a status field. The application focus can be temporarily moved to this field using a shortcut. This enables visually impaired users to query the current status in case they missed the auditory indication.	The status bar is a regular label component which accessible name is dynamically updated, every time the command state changes. Pressing the status bar shortcut key simply deviates the input focus momentarily to the status bar, so JAWS may read it, and then returns the focus to the component that previously owned it.

The implementation of support for the system look and feel (colours and fonts), nested layout managers and accessible names and descriptions was straightforward. In fact, it is good practice to implement such features in any Java GUI. Other accessibility features, more specific to the GUI of the spectrometer client, involved non-standard implementation techniques, namely the direct shortcuts (that is, not requiring a modifier key). It also required the mechanism to support shortcuts for groups of buttons, the indication of the 'command acknowledged' stage and the shortcut mechanism for the status bar.

XML transformation approaches

The interfaces to the laboratory equipment are automatically generated from XML description of the interaction elements for the experiment. These descriptions are transformed automatically on initialisation of the applet or application software. This raises the possibility of transforming differently for different users to realise their particular interface. This has been a deliberate design approach in order to be able to provide tailor-made interfaces. The approach was outlined more fully in Cooper et. al. (October 2000) this is subject to further research and development and evaluation and will be reported in detail at a later date.

Evaluations to date and Lessons Learnt

Initial usability evaluations

Prior to the implementation of the accessibility solutions, an initial user interface design was evaluated with 6 non-disabled students. The students worked in pairs to conduct part of the overall task and were located in different rooms in order to simulate working at a distance. They used Microsoft NetMeeting to communicate with each other to discuss the task, and with a remote tutor. On completion of the task the students were asked their opinions of the user interface via a questionnaire and an interview.

The results of this evaluation show that, overall, the students enjoyed doing the task and found it interesting. They also liked the collaborative aspect of it, and found this useful. The main issue that was identified was that the students required more feedback that their actions at the user interface were being executed by the apparatus, as this was not always obvious from the video feeds. On the basis of this evaluation a number of other improvements have been made to the user interface. These changes mostly relate to the labelling, positioning, and grouping of the controls.

In order to evaluate the developments since the initial evaluation further evaluation is planned towards the end of 2002. This evaluation will involve the same methodology and metrics as were used in the initial usability evaluation.

Initial accessibility evaluation

During the adaptation of the user interface for accessibility it was continually tested for keyboard operability and compatibility with a Java-supporting screen-reader (Jaws for Windows 4.02). Therefore the authors were reasonably sure of compatibility between the application and the screen-reader. Once all the adaptations had been made a blind professional evaluator informally evaluated the interface. This session highlighted some aspects of the interface that required improvement. Some of these issues clearly illustrate the benefits of evaluating with even just one disabled user. For example, the keyboard shortcut on a Move button was V, and the label was "Move V". When presented in synthetic speech there was an elision between these two words and the evaluator thought the shortcut was E.

The accessible user interface is to be formally evaluated by students with a range of disabilities. The aims of these evaluations are to examine a) the compatibility of the application with the assistive technologies and operating system settings used by students with disabilities, b) the usability of the software for students with disabilities, and c) whether the application supports accessible learning. The methodology and metrics used in these evaluations will largely be the same as those used in evaluations involving non-disabled students, but a number of adaptations will be required:

- Blind students will need to collaborate with sighted students in order for the latter to describe the visual aspects of the experiment
- Students with disabilities will use different channels of the collaboration tools depending on their preference or disability
- Metrics used with disabled students will include questions about use of their assistive technology with the application, and about the general approach used by the project to support students with disabilities in conducting experiments remotely

There exist some challenges in working with disabled users in developmental evaluations. There is a general problem of recruiting and retaining suitable subjects and the time commitment required of them over the development cycle. Each user will usually require their own set-up of Assistive Technology. There is an issue of representation; the diversity of disability and coping strategies and educational background that exists means it is never practical to work with a truly representative sample. In the educational context what we are seeking to evaluate is whether the software is accessibly promoting effective learning, metrics for this can be difficult to devise.

The application cannot currently be used effectively with screen magnification software. Although the interface can be magnified by such software, the latter cannot follow the application focus as it moves around

the screen. This function is important for users of this type of assistive technology. It is thought this problem is due to lack of support for Java in the magnification software. This issue is to be further investigated.

Conclusions

The PEARL project has demonstrated the remote control of complex functionality can be achieved across the Internet. Further, evaluations conducted to date confirm that this can be an effective way of extending access to practical work at higher education level. The user interface design for the Open University's Optical Spectrometer is a good exemplar of how widespread accessibility for disabled people can be achieved in JAVA based applets or applications whether these be for interfaces to remote functionality, locally run simulations, or interactive multimedia. Indeed it is now being used as such across the Open University.

Acknowledgements

The PEARL project is funded by the European Commission's Information Society Technologies program (Project reference IST-1999-12550). The partners in the project are the Open University, University of Dundee, Trinity College - Dublin, Faculty of Engineering at the University of Porto and Zenon SA - Athens.

References

Cooper M., Scanlon E. and Freake S. L. (June 2000). Remote Controlled Teaching Experiments, in Science and Engineering Subjects, Accessible over the World-Wide-Web - The PEARL project, *Proc. ED-MEDIA 2000 Conference*, Montreal, Quebec, Canada.

Cooper M., Santacruz L. P., Donnelly D. and Sergeant P. (October 2000) User interface approaches for accessibility in complex World-Wide-Web applications- an example approach from the PEARL project. *Proc. 6th ERCIM Workshop on User Interfaces for All*, Florence, Italy.

Technical notes and URL references that have informed the work described in this paper:

- IONA Technologies, Inc, *ORBacus For C++ and Java - Version 4.0.5*, 2001.
- Sun Microsystems, Inc, *Java Media Framework API guide - JMF 2.0 FCS*, 1999-11-19.
- Freedom Scientific, Inc., JAWS for Windows 4.02 Features and Enhancements, 2002, http://www.freedomscientific.com/fs_products/software_jaws402newfea.asp.
- Jeff Dunn, Developing Accessible JFC Applications, 1 June 2000, http://www.sun.com/access/developers/developing-accessible-apps/.
- Sun Microsystems, Inc., Java Look and Feel Design Guidelines, Second Edition, Chapter 6: Behaviour, Version 2.0, February 2001, http://java.sun.com/products/jlf/ed2/book/HIG.Behavior.html.

URLs referenced in the text:

URL 1: WAI Content Accessibility Guidelines: http://www.w3.org/TR/WAI-WEBCONTENT/
URL 2: Sun's Accessibility Guidelines: http://www.sun.com/access/developers/index.html
URL 3: IBM's JAVA Accessibility Guidelines: http://www-3.ibm.com/able/accessjava.html