Int. J. Engng Ed. Vol. 24, No. 1, pp. 46–55, 2008
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 2008 TEMPUS Publications.

Interactive Student Engagement Using Wireless Handheld Devices*

L. PETROPOULAKIS and F. FLOOD
Department of Electronic and Electrical Engineering, University of Strathclyde, Royal College, 204 George Street, Glasgow G1 1XW, UK. E-mail: akis@eee.strath.ac.uk, fflood@eee.strath.ac.uk

This paper presents an initial design of a pilot wireless Classroom Communication System (CCS) used for continuous and interactive engagement of students aiming at enhancing student critical thinking, extending attention span and enabling better student assessment. The system was designed mostly for engineering students and is intended to be used in lectures, tutorials or laboratories. The design should ultimately enable students to use, amongst other software, standard engineering packages such as MATLAB, PSpice, or Electronic WorkBench to construct designs, perform simulations and obtain answers to design problems using just wireless handheld pocket PCs. The system is based upon a CSCW system originally designed to be used anytime during lectures or tutorials and may involve the guidance and personal intervention of a lecturer or tutor. It is intended to support several modes and allows group or one-to-one personal tutoring. The system may also serve as a means of assessing individual student performance and in assisting lecturing staff with other tasks.

Keywords: wired classroom; handhelds; pocket PC; assessment

INTRODUCTION

THE ADVANTAGES OF INTERACTIVE COMMUNICATIONS in classrooms have successfully been demonstrated over a number of years in several educational institutions. Amongst the innovators in this area are the University of Massachusetts, Amherst with their Classroom Response System [1] and Harvard [2].

Lectures usually have two important aims: to increase student understanding of the presented concepts, and to do so in ways that demand attention by the audience and even provide enjoyment. Past studies have cast doubt on whether traditional lecture delivery accomplishes the first, finding that passive observation of demonstrations does not significantly improve student understanding of the associated concepts.

Lecture presentation is directly linked to the effectiveness of the presented lecture—a good presentation tends to induce further discussion and promote active thinking as well as further exploration into the subject area, prompting students to discover inconsistencies or weaknesses in their own thinking or knowledge. With universities currently being the main vehicle for mass higher education, student numbers are increasing and so are the ranges of skills and motivation these students have. At the same time, secondary education is sometimes being driven by success in examinations (e.g. in UK)—an approach which is at risk of promoting an attitude amongst students that the most important issue in their education is assessment rather than learning. It is for these reasons that there is now, more than ever before, a need for universities to focus on more effective and less laborious means of learning.

In response to this need, a system was developed at the University of Strathclyde in Glasgow known as NATALIE [3]. The philosophy was to re-emphasize the role of critical thinking by moving away from the traditional lecture format. NATALIE allows tutors to present information to large classrooms and obtain responses from the students through a 'voting' process using infrared transmitters and receivers. Useful as this system has shown to be, it has several limitations due to the technology involved:

- Only multiple choice questions may be asked during lectures.
- It can only be used in specially designed classrooms.
- It evaluates student knowledge but it is not thorough enough to provide a broader view of student ability.
- It can be used in most lecturing courses but it provides little assistance in subjects where the use of engineering packages is essential.

Our intention is to build upon the positive aspects of interactive Classroom Communication Systems (CSS). To do so it is, we feel, necessary to attempt to remove the aforementioned limitations and provide a fully Interactive Classroom Learning Environment (ICLE). Our attempt to achieve this is based on a modified version of our Computer Supported Collaborative System (CSCW) which is capable of sharing all types of software across

* Accepted 6 November 2007.

several computer platforms including wireless handheld PCs. Our intention was to develop a support system which would permit the engagement of students at any time, in any environment and for any course modules.

Studies using handheld PCs have been carried out in the Pebbles project [4, 5]. This is an initiative which has been supported over a number of years by organizations including Microsoft, Hewlett Packard, the National Science Foundation and DARPA. The project aims at evaluating the use of handheld PCs in classroom environments, for people with disabilities, as the command post of the future, as a personal universal controller, etc. The experiments carried out in classroom environments amounted to using PDAs as a 'voting' device (i.e. much in the same way NATALIE used custom-made infrared devices). Further details on this development are reported in [6].

A more sophisticated system known as 'Classtalk' is reported in [7] and [8]. This is a classroom communication system which was developed in collaboration with Texas Instruments using graphic calculators. Classtalk was relatively expensive but very sophisticated allowing questions other than multiple choice and with bi-directional feedback. Because of the growing choice in tools for interactive teaching, it was decided at the end of 2000, that 'Classtalk' was no longer a system in demand.

MOTIVATION AND OBJECTIVES

Our motivation for creating the present system stemmed initially from the requirement to provide students with a more enjoyable and rewarding classroom experience, allow tutors additional flexibility in presenting difficult engineering concepts through advanced demonstrations possibly involving visual displays at a reasonable cost. It was clear from previous research that this would need to involve an interactive classroom environment which could also help increase student attention span.

The development presented here was inspired partially by the work presented in [4] and [5], partially by the sophistication of Classtalk, and partially by a small scale survey carried out amongst our student population.

Our intention for this project was to create a comprehensive interactive classroom environment where the use of standard engineering software packages would be possible and where multiple interactions could take place amongst students and tutors on a group or individual basis. Classtalk's capabilities appeared to answer most of the requirements but we were looking for a considerably less expensive system and one that could provide a mobile solution which could readily be used in any lecturing, laboratory or tutorial class. Central to the overall concept is the idea that students would carry and use their own portable device. If successful, this approach can potentially have additional benefits in reducing costs for purchasing and maintaining large numbers of underused laboratory computers which often exist in engineering and other university departments.

Tablet PCs and wireless connections appeared to be the obvious choice for our experiments but there were two main constraints: cost and size. It was obvious that tablet PC prices would be outside the range of most students. More importantly, however, our survey indicated that the size of tablet PCs made them largely unattractive to students. Nearly 75 per cent indicated unwillingness to carry tablet PCs —preferring something less bulky instead. The reported results and experience of the Pebbles project indicated that handheld PCs could potentially provide a solution. Thus a decision was taken that the development would focus on PDAs as the main client device.

The system presented here also has a different approach to that presented in [5] and [6] and hence different objectives. It is primarily aimed at engineering courses and the aim is to permit use of standard engineering software packages whilst, at the same time, providing means for continuous, rigorous and automated assessment of student performance. This experimental system, which is still under development, was set up to evaluate the following:

a) establish best possible routes of delivering material and engaging students using handheld devices;
b) evaluate the technological constraints associated with use of these devices, particularly pertaining to graphic intensive packages used in engineering;
c) examine additional overheads involved in preparing material to be delivered in this way;
d) gauge user response to this experimental system;
e) examine bestpossiblemethods to assess student performance.

To achieve these objectives we decided to build upon our previous work on computer collaboration and extend it so that handheld PCs could be used in much the same way as ordinary PCs but with the added advantages which would enhanced student experience, provide continuous, better and faster evaluation of student ability and performance, and allow flexibility and mobility.

The concept then is that students can log on to the system, using PDAs and wireless connection points, and access applications such as MATLAB or PSpice in designated servers. Tutors can share with students (individually or collectively), guide them or oversee their work, take control of applications if needed, assign questions, exercises or tests and generally interact with students collectively or individually in a wireless intranet environment.

THE ORIGINAL GENERIC ARCHITECTURE

The architecture on which the current development was based is a patented platform-independent server-client collaboration system. The applications which are shared all execute on a server but PC clients only require simple Java-enabled browsers to control and view the applications. For Windows CE clients Java-enabled browsers was not an option and therefore a special interface was developed to enable the same functionality. More details on the overall structure depicted in Fig. 1 can be found in [9]. The information provided here only pertains to how the system handles multiple users in a classroom environment and the various classes of users the system can handle.

The server

The server side is an administration and client controlling system. A daemon process 'listens' for incoming network connections. When a client is identified, a Client Control Agent (CCA) is activated to handle the client requested session. The CCA connects the client and, if this is the first client, an application session is instantiated and the client assumes full control of the application. Subsequently detected clients wishing to join the same session are provided with as much control of the application as their login status allows them. Thus a user with full control over an application has a privileged connection whilst a user who is merely an observer is allowed to receive output from the application and observe the actions of the privileged users only. Other levels of application control also exist in between.

It should be noted that a server need not be thought of as a simple central computer device. It should be viewed as a cluster of machines, of computers (the number can vary) all of which work to provide services to clients.

The client

A successful login by a client initiates a process in the server to add the user to an existing sharing session, a new sharing session, or a single user session. If a login is successful, the agents required to handle communications are created on both sides of the network connection

For Windows CE clients the interface is shown in Fig. 2.

When the connection is made, the current state of the server is passed on to the client. This includes the number and names of sessions available, any applications available for sharing within each session, the usernames of users whose sessions are—at that moment—available to join.

Figure 2 also shows the various user modes that are currently available. These are: Single User, Tutor User, Supervised User, Supervisor User, Sharing User and Passive User.

Each of these modes defines the privileges and amount of control users are allowed to have during sharing sessions. For example, a Passive User is merely an observer whereas a Supervisor User has full control over a select number of applications only. In contrast, a Tutor User has full control of all the applications available during sessions. In cases where several sharing sessions take place simultaneously (as for example in an on-line tutorial) Supervisor Users can take control of applications active in their session only. Tutors can assume control of any application in a session they participate. It is possible for users to login with full control status in one session, but with only observer status in another. A hierarchy control tree is shown in Fig. 3.

MODIFIED ARCHITECTURE

The development presented here is based on a standard Computer Supported Collaborative Work (CSCW) system. In general the architectures of such systems are unfortunately not well conceived for use in large classroom environments, certainly not for use with thin clients. In class-

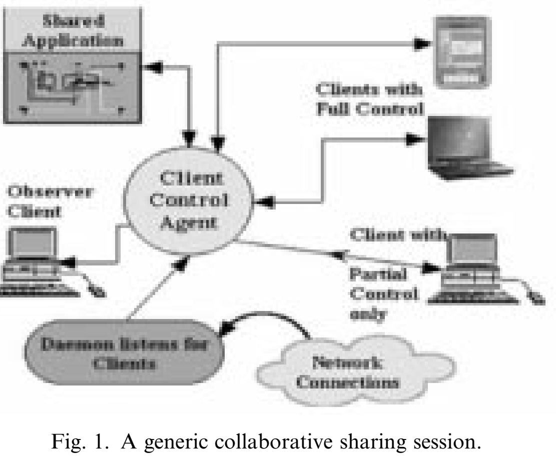
Shared Application
Clients with Full Control
Observer Client
Client Control Agent
Client with Partial Control only
Daemon listens for Clients
Network Connections

Fig. 1. A generic collaborative sharing session.

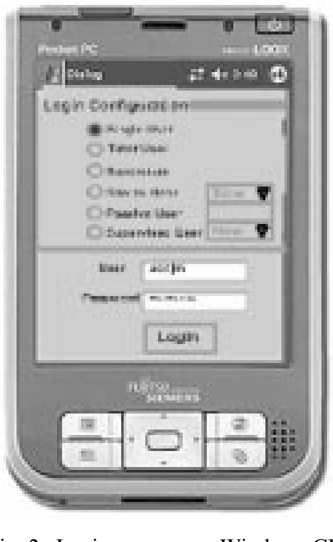

Fig. 2. Login process on Windows CE.

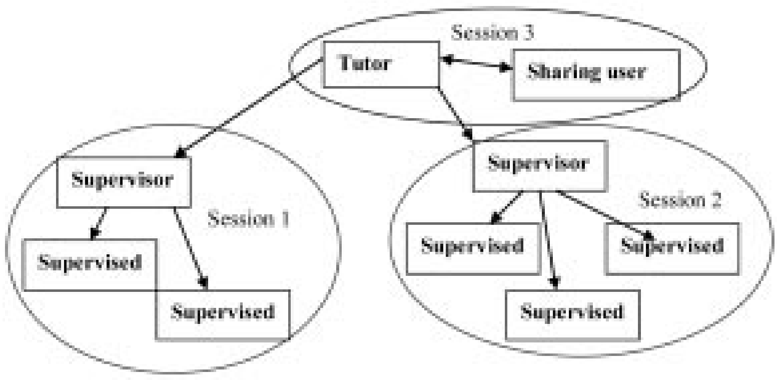

Fig. 3. Control hierarchy—arrows point to lower levels of control.

room, or tutorial-type environments, users work individually or in clusters as teams. However, at any moment, all users are likely to switch to a common sharing session (e.g. when a tutor takes control of the whole class). With thin clients, the programs on which users work must be held centrally (particularly true for large engineering packages) and this implies that several central machines would be required to accommodate this scenario. This would immediately nullify any cost advantages in using small handheld PCs as opposed to tablet PCs. Another important issue arising is that usually such systems are based on screen-sharing techniques and this creates additional issues.

Our initial solution to these challenges is based on the ability of software such as MATLAB to allow several different projects or files to be open simultaneously, although only one project or file can be active at any given time. If it is assumed that each project or file is created by a different user or cluster of users then, in theory, a system such as MATLAB can potentially cycle through these projects, obtain the results and present them to the different users. This, of course, implies that records are being held relating clients to projects. MATLAB and other software packages are not designed to do this but we have modified our development to interact with these packages and achieve just this effect.

In the interest of clarity we will dispense with components that are not essential to the explanation of the process, we will assume an MS Windows operating system and we will also use MATLAB for illustration purposes.

Server

On the server side the Client Control Agents (CCA) administer the connected clients. Each CCA is capable of supporting one or several connections depending whether it serves a single client or a group of collaborating clients. When a CCA is created it also creates an area in the server specific to that CCA, where any files can be uploaded ready for use. Users are notified of their designated area at the outset. Each filename created by users is then prefixed by a CCA_No where No is a number associated with a particular CCA. This is done to ensure that no filename duplication exists when files are uploaded to MATLAB, which can confuse the package. CCAs keep records regarding controlling clients in a group of users. CCAs also inform the collector of the login names of all the users they control and update this information dynamically.

Clients send requests to connect to the system. These are passed to the Connector Controller where they are placed in a first-in-first-out stack together with information relating each request to the corresponding CCA.

The ‘Connector’ and its controller are components whose job is to connect each CCA to the ‘MATLAB Controller’ and the ‘Output Capture’ mechanism and keep records of the activity for as long as required.

When the system is ready for use the controller identifies which CCA is associated with the first request in the stack; an invitation is issued to the clients of this CCA to use the system and a direct connection is established with the MATLAB controller. The stack may be accessed and altered as required by a tutor or supervisor.

The ‘Matlab Controller and Output Capture’ component is really a processing system that captures the windows generated by MATLAB, filters the information and sends it to the CCA to which it happens to be connected. From there the information is passed to the clients.

The ‘Collector’ merely gathers information, ensures that outputs sent to a client are the results of inputs from that client and passes this information to the ‘Support and Analysis’ components so that general feedback can ultimately be provided to participants and tutors.

Upon instantiation of the system, therefore, a tutor or instructor starts MATLAB on the server. Here we can have the following scenarios:

1) **Demonstration:** If a tutor is to demonstrate a process, the whole class is invited to log into the system and view the demonstration. The system is placed into ‘Demonstration’ mode and all clients connect to one CCA. The result is similar to the process where one user (i.e. the tutor)

has control of the package and the students are observers. In this mode clients are allowed to 'vote' if they are presented with a set of options. This can readily be achieved through an intranet website and a forced browser navigation function available to tutors. This can be used to force all client browsers to a specified page on the website. The Support and Analysis system (see below) collects the answers for further processing.

b) **Individual Working:** If users are to work individually the system is placed in 'Indvidual Working' mode. MATLAB users prepare m-files using any editor on their PDAs and then upload the file in the designated area created for them, by the CCA. When an invitation to use the system is received by a user, an acceptance must be issued within 5 s (default) and the user then has another 10 s (default) to specify which file in the designated area is to be used. The system runs the file and relays any output to the user.

c) **Group Working:** The process here is identical to that for individual working, the only difference being that a single CCA administers all the clients in the group. Upon login users identify the groups that exist and request to participate in a given group. The CCA checks the identity of the user and proceeds with the login process for the appropriate group. At any time only one client is in control of the group although the controlling user can change at any time. The CCA will only accept input from the designated controlling client but it will send output to all clients in the group. The controlling client can communicate with the other clients in the group for various reasons such as handing over control to another user. This is done through a chat facility and can be done at any time.

The following are of special importance:

- Tutors or supervisors are potentially super-users with the ability to take control of a session as required. They also are able to communicate with all the users individually, or collectively. This is achieved through the CCAs that control users. Tutors simply point to the user or group of users and the communications are routed through the appropriate CCA. At the moment it is not possible to communicate individually with users belonging to a group (the whole group is usually addressed).
- This system is based on a screen-sharing technology but with the following important differences:
  - i) The original system (described in section 3) shares specified windows only and not whole screens. This is the main reason why it can be adopted in this way as it can easily differentiate on the output sent to different users. Users only see the window associated with their process and cannot see any other processes from other users. Tutors can see the whole of system as required.
  - ii) Another reason that makes this process possible is that unlike other CSCW systems there is no need for a user to be present at the server side for collaboration to occur. Provided that the server is active and running, applications can be accessed remotely. For security reasons only specified applications can be accessed remotely.
  - iii) The command window of MATLAB is always kept visible to track any errors and inform clients. This is done by forcing the command window on the screen to be in a different screen area and by automatically resizing both this window and any other windows as needed. Information in the command window is only being routed to the clients in case of errors. During operations it is possible that errors can occur which would essentially cause the MATLAB to wait for input to continue. This could get the system to 'hang' and not allow other users to continue. Currently, tutors, who always have full information of the state of the system, are expected to intervene and clear the problem.
- File upload by users into their designated areas can take place any time irrespective of whether users have access to MATLAB at the time.
- Switching between modes is an important part of the system design. Users normally login and work on 'Independent Working' mode. If they then need to switch to 'Demonstration' mode, user CCAs are forced to link to the CCA of the tutor who will provide the demonstration. Once the demonstration is completed the individual user CCAs are 'freed' and users return to the original mode. A similar situation occurs for users working in groups.

Effectively this is a development where users share the same copy of MATLAB and take turns to run different files on the system. This process is rather slow and limited in potential but a lot depends on the way the system is operated.

Clients

The clients are handheld PDAs which access the system through wireless access points. For our pilot scheme each access point used was capable of handling a maximum of 20 clients at speeds of up to 11 Mbits per second. The PDAs used were enhanced with specially developed software designed to handle activities normally associated with PCs but which do not exist in PDAs such as.mouse double click. This additional flexibility allows use of existing off-the-shelf legacy software much in the same way as if the package was accessed by a PC.

A maximum of four wireless connection points were used in the experiments and a maximum of four servers each running a copy of MATLAB

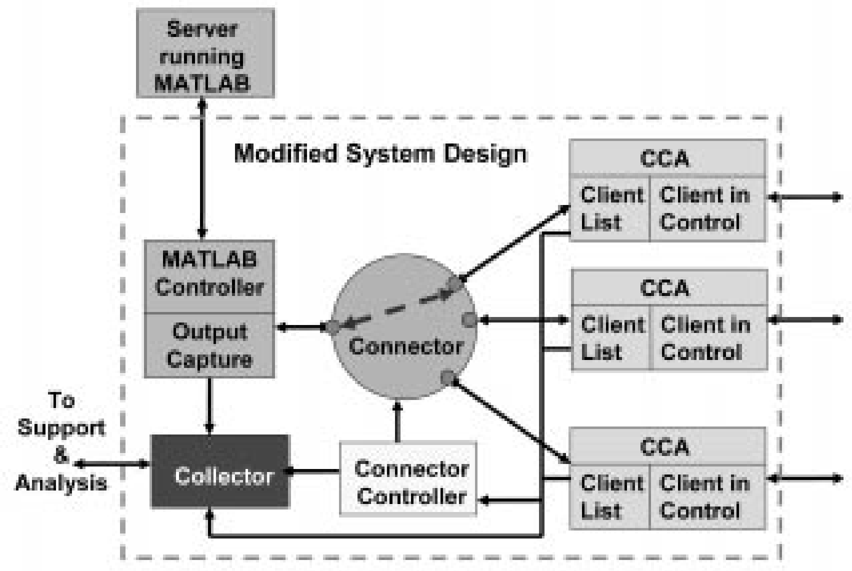

Fig. 4. Main design modification (My positioning. No reference in text. Refer to authors?)

were made available to PDA clients. The system automatically tries to provide a balance so that the number of clients handled by the servers is evenly distributed.

A zoom feature has been added to the PDAs so users can obtain a better view of specific parts of the output presented to them. In this respect, the images sent to the thin clients from the server are full scale images of all the server window(s) relating to the process controlled by the user. When the images are received at the client PDA they are stored in memory and, for display purposes, are scaled to the client's resolution. The user can then specify the centre of the area of the screen to be magnified and zoom is achieved in specified steps. This process is totally independent of the server.

Chat facilities allow users within groups to communicate and arrange the control of the process as well as handing over control to other users. Only controlling users can issue commands to the server and can save files.

A typical illustration of the user interface seen in the thin client is shown in Figs 5 and 6.

Another function which has been imparted to clients concerns PDA browsers. Software exists on each PDA that permits tutors to force-navigate a PDA browser to a specific intranet website and any page within it. This is just a means of ensuring that all students are ready to 'vote' if they are requested to do so.

Support and analysis software

The support and analysis part of this development is still largely incomplete and for this reason we can only provide an overview of the output. Components operate independently of the rest of

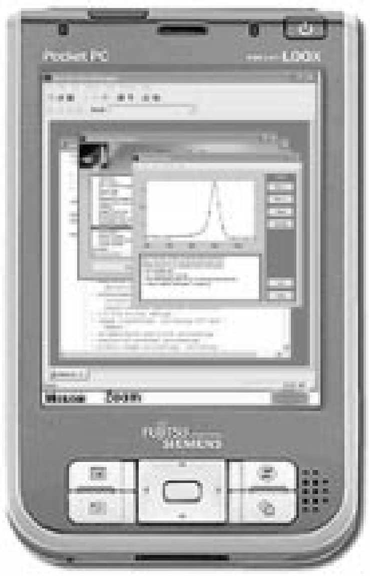

Fig. 5. Thin-client user interface.

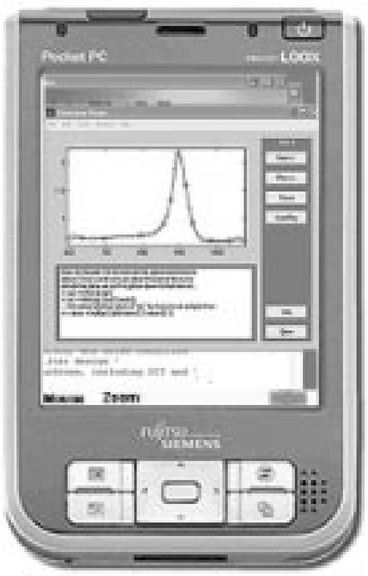

Fig. 6. A zoom-in view.

the system. Their purpose is to receive information from the servers, store it, analyse it and provide feedback to users.

It was felt that feedback could be best provided if the results were presented in the form of web pages in an intranet environment. This approach enables users to access their results at any time during the interactive classroom or tutorial and at any other time thereafter as required. Clients are therefore expected to have a browser pointed at the designated area where results will appear as they are processed.

On completion of the software the intention is that the following functionality will be provided:

a) Permit tutors to record user attendance, the session and the group to which users belonged during a given session.
b) Enable tutors to closely evaluate activities of groups and individuals, the frequency of submitted work, the quality of the work produced and the general individual and collective performance.
c) Provide automatic cross-correlation of attendance records of students with other classes allowing early alerts to poor performance and poor attendance.
d) Allow tutors to monitor and assess performance continuously throughout the period of studies and enable them to provide focused help and support where it is most needed.
e) Permit students direct and immediate evaluation of their performance and help them towards their own personal development.

Currently, the only functions that are available in this part of the system is user attendance recording, logging of user activity, performance evaluation on set tests and rating results, with the last two activities still being very limited.

Performance evaluation and result rating consists mainly of comparisons of user output against expected output. Naturally the comparisons vary depending on the tests carried out but typically the sequence of events is as follows:

1) Check for completion of test and number of attempts by user.
2) When a program has been developed check for efficiency of code (e.g. number of steps to achieve results).
3) If graphs are involved check the points of the graph generated and check them against those expected.
4) Evaluate results.

At present the results that can be made available are collective classroom performance results indicating ranges of success. Individual results are not yet available (even for simple 'voting' procedures) to users due mainly to intranet administrative and operational issues. It is expected that these issues will be resolved soon.

TESTS, EXPERIENCES

This experimental system was tested on four occasions between October 2005 and January 2006 with the number of users ranging between 5 and 35 (the last time). The aims of these tests were predominantly to:

1) Test the software, identify areas where improvements are needed and evaluate robustness.
2) Ascertain and evaluate other operational issues associated with the system.
3) Make an initial assessment of the additional overhead in time and effort required by tutors to run the system.
4) Assess the experience of participants and evaluate ease of use and degree of support towards the system.
5) Evaluate the benefits and costs of operating such a system.

The tests involved various exercises by individuals and groups using MATLAB. The main points of our findings are presented sequentially in the order they were experienced by both tutors and students. It is important to point out that the tests which have been performed to date provide pointers only and cannot provide any conclusive indicators at this early stage of our work.

The 'Demonstration' mode presented very few problems and it does not merit special mention. Tutors were able to run experiments and MATLAB demonstration programs freely and students were able to receive and respond (vote) if needed. The collection of the results was simple and easy to feedback on a collective basis. Any difficulties we met in this area were related to individual feedback. Additional work is needed here to ensure that the information is only available to the recipient for whom it was intended.

The 'Independent Working' mode naturally proved very challenging. At the outset we were aware of two main constraints inherent to this design:

1) Tests needed to be simple and compact and time for execution and output of results was strictly limited to permit a reasonable throughput of users;
2) There could only be a finite number of users per server machine and this is dictated largely by the time allocated for completion of each test.

These constraints implied that, assuming a maximum load of 10 users per server, each test devised to run in the system should execute and present output to users within 15 seconds of submission. Assuming also that users are able to submit their work within 15 seconds (an easy target), the class could be presented with a new test every 5–6 minutes. This of course does not account for the time taken to complete the test and upload it to the designated area. Clearly that meant that all tests given to students to complete had to be both simple and well constructed to demonstrate the concepts that tutors wished to explain. The implications here are that there is an initial overhead for tutors to prepare such material as well as means for assessing this material.

During initial tests users were asked to develop programs using their PDAs.With the exception of very simple and very short tasks, this is not a workable solution in a classroom environment. In latter tests we developed and used a series of templates and asked users to fill-in missing components in programs. This approach had the following direct advantages:

- Students have to resolve the problem first before uploading and running their solution. This resulted in a staggered access to MATLAB servers and therefore imposed less strain on resources.
- Templates could easily be altered to present different problems to users; the overhead to achieve this was very small.
- The time taken for users to complete the task was very small, which helped to increase throughput in our system.
- As a side-effect, we found that a short focused problem helps with the concentration of users which tends to waiver sometimes. Creation of programs appeared to have the opposite effect. Unfortunately so far our study has been very limited to permit assigning serious importance to this.

With this approach and after a few tries we were able during our last series of experiments (with 35 users and four servers) to have a new test every 8.5 minutes on average. However, it was felt that this was still a rather long interval.

In some tests we used a combined strategy. Users were initially provided with a template and a task to complete. Once they all knew the task then they were switched to a 'Demonstration' mode where they were asked to 'vote' on one or two questions relating to the task. With the results of the voting made available they were then placed back on 'Individual Working' mode. The throughput this time was reduced to an average of just over 6.5 minutes whilst the number of correct solutions increased by about 20 per cent. Again the tests are far too limited at this stage for these statistics to have any real meaning.

When working in groups we tested two different scenarios. Users were divided into small groups at random. This meant that users in the same group would not necessarily be physically located next to the other group members and the chat facility was used for communications. The process was abandoned as the groups could not decide upon a leader and information exchange soon became confused.

Users were then allowed to form their own groups, collaborate and designate a leader to communicate the information. The results in this case were very good with all but one of the eight groups completing the tests successfully. With only two inputs per server the time taken for completion was also very short after submission of the work, but without any real reduction in the average overall time taken. This was as a direct result of resolving group dynamics and abilities and having differences of opinion regarding the best solution. It was also noted that this approach has the counterproductive element that some users have a passive role in the group.

We experienced no problems in returning results to users for submitted work provided that the designs and tests were carefully thought out to run within the limitations of the system design. It was only the time taken to solve the problem and submit the files that created some difficulties. This would suggest that provided there is a reasonably good distribution of students on servers, this approach would work reasonably well and could have beneficial results.

Students found difficulties in using PDAs to complete involved tasks. With the tasks being simplified these difficulties became less prominent and users could concentrate more on the questions posed. Naturally it was much easier for all concerned to use a 'voting' environment only, although it was freely admitted that just a 'voting' process would be monotonous and a more challenging environment would be preferable.

As explained earlier, automatic assessment of tests has a considerable overhead attached to it. It is essential that prepared material is simple, easy to use and easy to access. Although we are looking into ways of providing more general ways for automated assessing procedures this is by no means a trivial task. We found that the use of templates has considerable advantages and overcomes many problems including issues of symbolic notation; it can simplify assessment to simple comparisons against accepted solutions.

CONCLUSIONS

The work we have presented here is an experimental system aiming at introducing Interactive Classroom Learning Environment through the use of wireless PDAs. The approach is based on a modified CSCW screen-sharing system. The development was provided for Windows-based servers and a series of PDAs running Windows CE and accessing the server through wireless access points serving a maximum of 20 users at speeds up to 11Mbits per second.

The approach used was both a technology feasibility study as well as cost-cutting exercise. We used one copy of the software (in this case MATLAB) per server and users took turns to access the server. This implied that the tests and exercises had to be simple and needed careful coordination and planning. It also meant that more complex designs involving GUIs such as Simulink could not be used in these trials as the time taken for users to interact with such interfaces would have been prohibitive. A partial solution to this problem is presented in [10] which could, with adaptation, allow a Simulink-type system to become available for PDAs. In the general context however, adaptations for other systems such as Electronic WorkBench and PSpice will also have to be found. A possible approach would be to generate reduced versions of such software which would then create files that can be executed in larger servers as described here. With PDAs becoming more powerful and increased storage capacity this could be possible very soon.

On the plus side the system appeared to provide a workable solution and allow use of off-the-shelf engineering packages in any environment provided wireless access points can be made available. It also seemed to free tutors from more mundane tasks such as taking maintaining class registers.

Naturally a lot of the issues we faced with this first experimental system were direct derivatives of the limitations of the server operating system. Windows is far from ideal—being in reality a singleuser system—for providing multi-user access to engineering packages. Had we used Unix or Linux, combined with a terminal server approach, then several of our constraints regarding users and the time taken to submit their work would not have existed. The reason for using Windows as a first attempt was simply to test how far we could stretch the capabilities of the system and how these limitations would affect students and tutors. What has become clear is that using Windows in a 'Demonstrator' mode and allowing students a 'voting' capability in the manner described in this paper present absolutely no difficulties. However, multi-use of the system will not, in many cases, present an ideal solution—depending on requirements.

This work has also highlighted the difficulties in automatic assessment of

online submitted work when submissions are not structured and are not web-based. Our initial solution to this problem for the trials we run was to provide templates and ask users to fill-in the missing information. By simply comparing submitted work to previous correctly completed templates a fast and accurate assessment could be carried out. When the submitted templates resulted in additional output such as graphs then the number of points in these graphs could be compared to provide an assessment. The number of failed attempts, together with information where failures occurred, also provided good indicators of competence and understanding.

Many of these procedures deployed in this pilot system are clearly dependent on the package used for the tests (in this case MATLAB); the 'Collector' and 'Analysis and Support' components will have to be modified to handle other packages and different methods of assessment. However, it is believed that the principles for assessment used in these tests have a more generic approach and can be applied to other packages such as PSpice and Electronic WorkBench.

This totally new implementation has revealed a number of technical issues that must be addressed before such procedures can become the norm. The advantages of using CSS systems have long been highlighted and our work presented here further indicates both additional benefits and also challenges.

REFERENCES

1. http://umperg.physics.umass.edu/topics/crs
2. C. H. Crouch, A. P. Fagen, J. P. Callan and E. Mazur, Classroom Demonstrations: Learning Tools or Entertainment?, *Am. J. Phys.*, **72**, 2004, pp. 835–838.
3. http://www.engsc.ac.uk/er/features/natalie.asp
4. F. Chen, B. Myers and D. Yaron *Using Handheld Devices for Tests in Classes*. Carnegie Mellon University School of Computer Science Technical Report, no. CMU-CS-00–152 and Human Computer Interaction Institute Technical Report CMU-HCII-00–101. (2000), http://www.cs.cmu.edu/~pebbles/papers/CMU-CS-00–152.pdf
5. B. Myers, The Pebbles Project: Using PCs and Hand-held Computers Together; Demonstration Extended Abstract. *Adjunct Proceedings CHI'2000: Human Factors in Computing Systems*. The Hague, The Netherlands. (2000) pp. 14–15.
6. http://www.cs.cmu.edu/~pebbles/v5/classroom/index.html
7. http://bedu.com/
8. R. J. Dufresne, W. J. Gerace, W. J. Leonard, J. P. Mestre, L. Wenk, Classtalk: A Classroom Communication System for Active Learning, *Journal of Computing in Higher Education*, **17**, 1996, pp. 3–47.
9. L. Petropoulakis. and F. Flood, Design and Development of a General Purpose Collaborative Environment, to appear in 2006 in the *International Journal of Computer Applications in Technology*, special issue on Collaborative Multimedia Applications in Technology.
10. L. Petropoulakis and B. Stephen, Resource Sharing Software for Distance Learning in Engineering Education, IJEE, **19**(3), pp 2003, 371–378.

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, he 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. He has over30 journal and conference publications.

Frances Flood is a Senior Research Fellow at the University of Strathclyde, and is currently undertaking an Enterprise Fellowship with the Royal Society of Edinburgh and Scottish Enterprise. She holds a 1st class Honours degree in Computer Science from the University of Glasgow, and spent six years working in the Investment Banking sector where she was a Senior Analyst Programmer.