Laboratory classes are integral part of scientific education, but they are resource-intensive, imposing significant logistic constraints upon the curriculum. An option to reduce these burdens is to implement remote access laboratory classes, where students remotely interact with devices, instead of in traditional laboratory environment. This kind of remote labs are increasingly popular, but their development has been often focused on specific experiences, to address the needs of a particular lab. Moreover they are often designed for one user at a time, i.e. the user experience can not be shared with others. In the article we discuss a new Web Lab platform that can be easily adapted to a large number of different laboratory equipments and that is based on CSCL (computer-supported collaborative learning) techniques in order to share the lab experience within small groups of remote students.

Keywords: 3.0.b Collaborative Learning Tools, 3.2.b. Computer science education, N.1.d Virtual Labs, 5.3.h Web-based interaction

I. INTRODUCTION

Laboratory classes play a crucial role both in schools and universities. Good pedagogical reasons, such as illustrating and validating analytical concepts, introducing students to professional practice and to the uncertainties involved in non-ideal situations, developing skills with instrumentation, and developing social and teamwork skills in a technical environment, motivate the need for the inclusions in the curricula [1]. CSCL (Computer Supported Collaborative Learning) is expected to provide a remarkable support to the efficacy of this practice. Remote labs indeed have the potential to provide affordable real experimental data by sharing laboratory equipments with a pool of schools [3]. Moreover, they can extend the capability of a conventional laboratory by increasing the number of times and places a student can perform experiments [4, 5] and extending its availability to several students [6, 7]. For these reasons, since 1996 [8] remote laboratories have been increasingly popular, but their development has mostly been driven by technical aspects rather than by the need to reproduce satisfactory user experience. Most of them, for example, have been designed for a single user at a time, so that students engaging in remote learning are isolated. Moreover, current solutions do not manage high level concepts like user privileges and security and safety features, that might result in potential equipment damages. In order to overcome the described problems, we foster the research of a more general, scalable and reusable approach, not limited to a specific laboratory equipment or to a specific experiment. Our goal is “not to reinvent the wheel” each time an equipment goes online, but to design a platform able both to put laboratory equipments online with the minimal effort and to define a satisfactory and effective collaborative user experience.

Our objective is to have a networked laboratory reproducing the best aspects of hands-on experiences. We believe, in particular, that collaborative aspects are essential to reinforce the interaction among the learners: the distance laboratory must foster active learning and comprehension construction by the student. In the social setting of the in-lab experience, the learner interacts directly with other students, the instructor, equipments, activities and other elements. These interactions guide interpretation and construction of concepts. They are key aspects of the laboratory experience that must be included in distance education laboratory experiences as well [9].

For these reasons we have developed the WeColLab (Web Collaboratory) platform, which both enables the remote control of a generic real laboratory equipment (like telescopes or microscopes, robotic arms and their related auxiliary devices) and lets groups of students share their lab experiences over the Web.

Actually WeColLab has inherited the collaborative approach and user interface design from Astronet, a collaborative web-based system that lets students and amateur astronomers to remotely control a real telescope, which has been successfully tested in 15 high schools classes in the south of Italy in 2008. Starting from Astronet’s architecture, WeColLab platform generalizes the application architecture, keeping the user interface design unchanged.

In this paper we will describe the application platform, referring to MicroNet, which is our Web system allowing students to collaboratively control an electron microscope.

The paper structure is: section 2 tracks the research foundations, section 3 describes the WeColLab architecture and user interface. Section 4 concludes the paper.

II. BACKGROUND

Remote laboratories target a large range of devices, from different scientific areas. This means that they are not restricted to a single educational topic, but they are being used for several devices and experiences that can be controlled using a computer [2, 11]. As the remote laboratory platforms are getting mature, we observe that they are still
built without a shared interoperable approach. For example Gravier et al. [11] surveyed 42 different remote laboratories finding that every project implements its own software architecture with no re-use. Nevertheless, all the analyzed projects are built on a common hardware/software structure, composed by:

- the **device** or the equipment to be remotely controlled;
- a **computer** connected to the equipment, i.e. the gateway between the device and the remote user;
- the **middleware**, to interact with remote lab.

Even so, technologies vary a lot from one remote laboratory to another, which prevents reusability and interoperability among the applications. As in [11], we analyzed 25 remote Web labs, finding that:

- the technologies and the programming languages generally used to connect the device and the local computer are often based on proprietary software, mainly Matlab with Simulink and LabView with data-socket, with some software based on Visual Basic or Python [4-9].
- Java is frequently used to link the laboratory equipments with the remote users, but it is often coupled to different technologies like PHP, HTML, CORBA, VRML, etc. Different approaches, based on ASP, ActiveX, Python and C++, or proprietary software based on LabView and Matlab are also exploited by various authors [4-9]. Some remote labs adopted software solutions, such as VNC, as it provides them with the remote control over the local computer connected to the corresponding equipment. Nonetheless, this solution was given up as it lacks security and is bandwidth intensive [12].
- A remote lab project usually implements a stand-alone application. It can be very expensive to build, since it requires a large amount of time, money and specific skills. The software application is usually dedicated and is not supposed to be reused in other similar labs.
- Hands-on laboratories can be composed of several devices, that create an experimental workbench when connected together. The experiences we have compared address only the remote control of one device at a time. In order to provide students with complete workbenches, remote laboratories need to connect different devices, which can be distributed in different places on the Web.

The experiences described in 22 remote labs show that collaborative aspects are still little considered in remote laboratories [6, 11]. In particular we point on the following shortages:

1. **Lack of presence awareness** among the participating students: this feature focuses on the concept of awareness and concrete perception about the physical presence (as in several instant messaging systems) and the activities of other participants, and their interactions with the remote equipment.

2. **Lack of user perspectives**: in a collaborative Web application users should have different interaction experiences according to their role; for example in a Web lab, to moderate the virtual classroom, the teacher and the tutor need widgets and indicators which are not available to students.

3. **Lack of group dynamics**: this concept takes into account how users interact each other (i.e. the presence of a leader or a moderator etc.) and it is very important for student’s evaluation and for evaluation purposes.

III. THE WECOLLAB SYSTEM

In order to define a Web-based system, enabling real laboratory equipments (like a telescope, a microscope, a robotic arm, etc.) to be remotely controlled by a virtual classroom through a domestic Internet connection, we collected the following requirements:

- it must be Web-based: students must access the remote lab without the need of any special software to carry out their experience (but the standard Web browser). The implementation must be based on standard protocols and common components;
- it must be collaborative: like in a multi-conference, small groups of 2 to 20 participants (students and teachers) must be able to see/hear/talk to each other, supported by various collaboration tools (shared whiteboard, picture annotation, chat…), moreover they must see (all together) and remotely control (one at a time) the laboratory equipment;
- a coordinating tutor/supervisor must manage the laboratory session, to authorize the control requests and to protect the equipment against any potentially damaging operation;
- the detailed requirements must be elicited by means of a comprehensive goal-driven approach, like KAOS [15], in order to bring all users, stakeholders, goals and objectives out. The approach must especially consider the main features of a collaborative Web application (i.e. presence awareness, points of view, group dynamics).

The scenario of use we imagine for the WeColLab platform is that of a virtual classroom which must remotely use one or more equipments located somewhere in the world. The virtual classroom is made up of a teacher (describing the lesson and commenting the experiment), an optional tutor (facilitating the use of the platform, supervising the correct and safe use of the equipment, solving technical problems and moderating the discussion) and a group of students (from 2 to 20), all connected via a domestic ADSL. The tutor/teacher owns tools to moderate the discussion, to tell disturbing pupils off the experiment, up till disabling their audio and video. Students can reserve their turn to remotely use the equipment and they can remotely control it. Students can remotely perform the same tasks as in the hands-on lab experiment. The concurrent
access of the users is managed via a sequential queue of requests administered by the tutor/teacher.

External people can optionally watch the lab session, but neither they can participate to the classroom’s discussion nor they can operate with the equipment.

A. The WeColLab Platform

According to the above requirements, as shown in Figure 1, the main components of the WeColLab system are:

- **The Equipment Server** made up of
  - **The Laboratory Equipment**, e.g. the electron microscope;
  - **The Redirector**, which allows the equipment to be remotely accessed and controlled, granting the proper access privileges to users and filtering user’s commands to avoid safety breaches for the lab equipment

- **The Collaboration Server** which manages the collaborative application logic, orchestrating the multimedia streams coming from the equipment server and from the users (students, tutor, teacher and watchers). In our implementation the Collaboration server is based on the MS IIS 5.1 Web Server and on the Adobe Flash Media Server 2. The component is split into two parts:
  - **The WeColLab Collaborative engine**, which implements the multi-videoconference, the shared whiteboard, the audio mixer and all the main features;
  - **The Micronet application**, which is the only equipment-specific part of the Component. We can adapt the Collaboration Server to different remote web labs (like a telescope, a spectrometer etc.) by customizing this piece of software (about 20 to 200 lines of server-side actionscript code).

- **The clients** which represent the students, the teacher, the tutor and external watchers. They receive audio, video and images from other participants and from the Equipment Server via the Collaboration Server.

  The remote control of the Equipment Server consists of three elements: a component which locally processes the commands (keystrokes and mouse coordinates) sent by the clients, another component able to capture the monitor video flow (with 800x600 resolution) and a third component which compresses and sends the video flows to the Collaboration Server in real time.

  According to the Laboratory Equipment type, we have developed two kind of Redirector:

  - **Software**: if the laboratory equipment includes a standard Windows-based system as controller, the Redirector components are deployed on it;
  - **Hardware**: if the Laboratory Equipment is based on proprietary hardware, not including a standard Windows-based system as equipment controller, we add an external computer with the role of I/O redirector, which interfaces the Equipment via an external keyboard & mouse emulator, for input commands, and via a video grabber for the acquisition, the compression and the transmission of the equipment’s video output. This solution is more expensive because of the additional hardware, even if technically safer.

  Thanks to the Redirector we are able to remotely control any lab equipment. The only constraint is that it must have a keyboard or a mouse as input device and a VGA/SVGA display as output device.

  Technically, the server-side components of WeColLab are based on the MS Windows OS, while the WeColLab clients are based on Flash components, are played in Web browsers and are OS independent.

![Collaboration Server](image1)

**Figure 1.** WeColLab’s logic architecture

The Collaboration Server is based on the streaming capabilities of Adobe Flash Media Server 2 (FMS2), which offers a unique combination of multimedia streaming features and a flexible development environment to create interactive and collaborative applications. FMS2 allows all the participants to receive and transmit their own audio/video flows, from WebCams and microphones, like in a multi-conference. The video coming from the Equipment Server is also acquired, compressed and sent in streaming to all the participants by means of FMS2.

In comparison with Microsoft platform and other media servers, only FMS2 owns the following features:
Compatibility with the main operating systems and browsers, which allows to reach more than the 98% of the users connecting to the Web;

Cheap distribution of high quality video content, due to the efficiency of the On2 VP6 codec

Dynamic client’s bandwidth detection and negotiation

Dynamic buffer set up, which reduces the start up time according to the client connection’s speed

Ability of capture and transmit live audio/video flows from any camera and microphone detected by the operating system and connected via USB port.

Multichannel and multi-user streaming and shared object technology to synchronize data among users.

Web administration console, allowing to show:
- Application data, objects and streams
- Server features, like CPU performance and bandwidth usage per application

Robust security model to protect content, which:
- Does not allow streams to be cached;
- Uses the RTMP proprietary transfer protocol, which reduces the stream ripping;
- Supports the SSL protocol;
- Blocks connections and unauthorized access.

In order to capture the Equipment Server’s video output, we chose a virtual camera like the TechSmith Camtasia or the VH Screen Capture driver (www.hmelyoff.com), which allows:

- The optimization of the video screen capture flow, with frame rate control
- The compatibility with Flash
- The control of the screen region to capture

In order to remotely control the equipment, we have developed three components:
1. a server side module, written in Flash, hosted in the Redirector, which captures the equipment server screen, compresses it and sends it to the FMS;
2. a client side application, written in Flash, which allows the remote user to see the Equipment Server screen, thanks to the Redirector, and to send the keystrokes, the mouse coordinates and the mouse events to the Redirector, under the supervision of the tutor;
3. a Keyboard & Mouse Redirector, written in actionscript, which synchronously intercepts, receives, interprets and executes the remote user commands (mouse actions and keystrokes).

The server-side components have been written for the MS Windows OS because equipment servers are often Windows based.

The choice of the FMS2 to manage all the audio/video flows requires that every client must be equipped with Adobe Flash Player. The wide diffusion of this plug-in, available for every operating systems and Web browsers, and installed on most of the computers connected to the Internet, makes WeColLab easily available to whom wants to participate in the remote laboratory sessions.

B. The User Interface

The WeColLab user interface design (Figure 2) includes five tasks areas: on the left hand side there is the multi-video conference area, showing the video (thumbnail size) coming from each participating students (buddy list). The magnified videos of the tutor (in the upper part) or of the student who has been enabled to pilot the equipment are shown in the upper-left corner. The set up console (lower-left corner) is used by the tutor to tune the students audio, to enable/disable the student’s microphone and to supervise the actual audio and video bandwidth measures.

In order to remotely control the equipment, we have developed three components:

1. a server side module, written in Flash, hosted in the Redirector, which captures the equipment server screen, compresses it and sends it to the FMS;
2. a client side application, written in Flash, which allows the remote user to see the Equipment Server screen, thanks to the Redirector, and to send the keystrokes, the mouse coordinates and the mouse events to the Redirector, under the supervision of the tutor;
3. a Keyboard & Mouse Redirector, written in actionscript, which synchronously intercepts, receives, interprets and executes the remote user commands (mouse actions and keystrokes).

The server-side components have been written for the MS Windows OS because equipment servers are often Windows based.

The choice of the FMS2 to manage all the audio/video flows requires that every client must be equipped with Adobe Flash Player. The wide diffusion of this plug-in, available for every operating systems and Web browsers,
Figure 3 shows a screenshot of the MicroNet application during an experimental session where a teacher is lecturing the virtual classroom of ten students about the linoleum’s properties and its microscopic structure. The Tools panel contains three thumbnails, with video streams from the lab, one coming from an ambient camera to show what happens in the lab room, another coming from a Head mounted camera, showing the microscope workbench, and a third one from the Equipment Server remote desktop, reproducing the SmartSEM Application which is the application managing the electron microscope. In Figure 3 the main panel shows linoleum’s texture.

The initial usability tests have been performed in form of a heuristic evaluation [13]. As proposed by Nielsen [14], we asked five usability experts to inspect the systems and indicate possible problems and drawbacks. General feedback was very positive. The inspectors rated the interface design as intuitive, and easy to understand and operate.

In order to evaluate the actual effectiveness, intuitiveness and simplicity of the interface, from September 2007 to June 2008 we tested Astronet’s user interaction in 15 high school classes in the Apulia Region. We asked 30 students to participate in our study (16 males and 14 females, age from 14 to 17, two students from each one of the 15 classes). Most of them had experience with chat and video games and no experience in operating a telescope. Our session started with a 30 minutes’ lesson about astronomy followed by a 30 minutes’ description of Astronet and sky navigation. For the evaluation we set up four tasks of increasing difficulty that users had to solve operating with the system. Since the proposed tasks were of increasing difficulty, we expected that all users would solve the first two but would have some troubles with the latter two, due to the implicit complexity of the astronomical topic. Surprisingly, all students were able to perform the four tasks, and were very satisfied about the experience. All of them said the interface was very intuitive and simple to use and functionalities easy to memorize.

As a side result, teachers were also very satisfied because they said they gained high attention from students and high didactic efficacy.

### IV. Conclusions and Future Works

In the paper we presented WeCoLab, a platform for remotely control a lab equipment in a collaborative Web space. We have described the platform in the case study of an electron microscope, but we have also successfully tested the same platform with the campus telescope and a robotic arm.

In the next academic year we have already scheduled a second usability and efficacy tests during an MS teachers’ advanced classes. From the technical point of view, WeCoLab is evolving into a service oriented architecture able to connect several remote equipments and several virtual classes in a virtual workbench over the Web.

### References


