ABSTRACT

Internet-based learning systems, or e-learning, are widely available in institutes, universities, and industrial companies, hosting regular or continuous education programs. The dream of teaching and learning from anywhere and at anytime becomes a reality due to the construction of e-learning infrastructure. Traditional teaching materials and methods are shifting to the new paradigm. In higher education, laboratory work is playing an important role in the area of training students and helping students to absorb more knowledge. With the goal of bringing e-learning to the traditional laboratory experiment, in this article, we present an architecture for an online laboratory e-learning system to facilitate the design and deployment of lab-based courses for e-education. The article provides an overall view of the system design and implementation so the Internet-based laboratory can be easily integrated with the e-learning infrastructure.

Keywords: e-learning; learning management system; online laboratory; reusable learning objects; SOAP; UML; XML-based messaging

INTRODUCTION

The Internet and Web-related technologies are affecting more and more person’s lives and work around the world in many positive ways. They are also bringing many changes to the education domain. A major change in this area is the way educational materials are designed, developed, and delivered to the student. Internet-based learning, so-called e-learning, is widely available in institutes, universities, and industrial companies as regular or continuous education program, such as Blackboard (Blackboard Inc., n.d.), Swift Author (Gemini Inc., n.d.), Macromedia Authorware (Macromedia Inc., n.d.), and TopClass Publisher (WBT Systems, n.d.). Most of e-learning systems provide the services of searching, downloading, and delivering learning content, which includes text, audio, animation, applets, flash, or video clips to their users in order to enhance learning experiences.

Internet-based laboratory (i.e., online laboratory) is a rapidly growing research in
universities. In some cases, it also is called virtual lab, remote lab, Internet lab, or Web lab. In existing e-learning systems, the hardware equipment is not supported by the e-learning infrastructure, although IMS Learning Design Best Practice and Implementation Guide (IMS, ver1.0, 2003) presented some use cases of a virtual laboratory. These use cases have been chosen to validate the conceptual model of a learning system. However, it does not provide the details of design and implementation of such a lab-based learning platform in a systematic way.

In this article, we have presented the architecture and modules of an online laboratory system, or OnlineLab for short. The main objective is to combine OnlineLab researches and e-learning infrastructure in order to achieve the goal of increasing and enhancing learning opportunities and experience for students. We first briefly present an overview of the e-learning system followed by a research review on Internet-based laboratories. Next, we provide details of the proposed architecture for OnlineLab, which is based on the current Shareable Content Object Reference Model (ADL, SCROM) learning model. Finally, we demonstrate sample implementations based on the proposed model followed by concluding remarks.

OVERVIEW OF E-LEARNING SYSTEMS

The following equation represents a typical e-learning system (Maish Nichani, 2001):

\[ \text{LCMS} = \text{LMS} + \text{CMS} [\text{RLOs}] \]

In this equation:

- A Learning Content Management System (LCMS) is a “multi-user environment where learning developers create, store, reuse, manage, personalize, and deliver digital learning content from a central object repository” (elearningpost, n.d.).
- The main purpose of a Learning Management System (LMS) is to manage students and learning events and to collate data on learner progress.
- The objective of a Content Management System (CMS) is to simplify the creation and administration of online content (articles, reports, pictures, etc.) used in publications.

In a CMS, complete learning courses are assembled from several self-contained chunks called content components. These content components, when used in the learning domain, are called Learning Objects (LOs). One important benefit of the LO approach is reusability. Learning objects could be combined to form a hierarchy of lesson, module, course, or curriculum in order to provide a rich learning environment and to reduce the time, instructor skill, or cost associated with development. In this case, it is Reusable Learning Objects (RLOs). With the LCMS, learners not only receive the instructions when they desire (just-in-time learning) but also receive only the portion of the instruction that they desire (granular learning, or just-enough learning).

Many organizations are working in one or more phases of the process in order to develop industrywide standards that ensure interoperability of learning solutions. Some are as follows:

- **Aviation Industry CBT Committee (AICC)** was formed out of a need for hardware standardization of CBT delivery platforms in 1988. It has published a variety of recommendations, including hardware and software configuration. Their computer-managed instruction (CMI) guidelines have had the greatest impact.
- **Advanced Distributed Learning (ADL)** is an initiative launched in 1997 by the U.S. Department of Defense and the White House Office of Science and Technology Policy. Their recently released Shareable Courseware Object Reference Model (SCORM) provides one of the best and most recent examples of the application and integration of these learning standards.
- **IMS Global Learning Consortium (IMS)**. Headquartered in Burlington, Massachusetts, it focuses on the development of XML-based specifications. These specifications
describe the key characteristics of courses, lessons, assessments, learners, and groups.

- **IEEE Learning Technology Standards Committee (LTSC)** formed in 1996, is developing and promoting instructional technology standards. The most widely acknowledged specification is the Learning Object Metadata (LOM) specification, which defines element groups and elements that describe learning resources. The IMS and ADL both use the LOM elements and structures in their specifications.

**REVIEW OF RESEARCH ON THE ONLINE LABORATORY**

Benetazzo (1999) described the specification and design of a geographically distributed system based on commercial standard components. González-Castaño et al. (2000) designed an Internet access laboratory that provides remote access to real equipment on SBC68K—a single-board computer based on a Motorola MC68000 microprocessor used in a Computer Architecture laboratory. Their system is based on object distribution paradigm in CORBA.

A low-cost Internet-based telerobotic system was applied in remote robotic education via China Internet by Song You et al. (2000). Some issues involving time delays associated with the Internet also have been addressed.

Chi and Chen (2001) conducted an experiment on a frequency modulation for students taking a course on communication principles at the National University of Singapore (NUS). Similarly, the REAL (Remotely Accessible Laboratory), a virtual laboratory accessible through the Internet, was implemented as a telematic service to allow remote control on mobile robots (Guimaraes, 2003). Professor Lambertus Hesselink (2003) from Stanford introduced a similar design for a remote laboratory, the so-called CyberLab, on an optical processor.

Almost all proposed systems for an online laboratory basically provide remote access to the experiment for researching or teaching purposes. From the e-learning point of view, these systems do not include Learning Management System and Content Management System support. In other words, no universal platform has been developed so far in a way in which to provide a rich learning environment nor has there been proposed any standard development for designing and deploying lab experiment.

However, the research works discussed previously provide a good ground for the design and implementation of an online laboratory system from many aspects, such as system architecture, real-time issue, and Internet time delay. They also have presented a great deal of example applications widely used by universities, which can be better utilized to provide a more attractive and useful e-learning platform if some form of integration can be done.

**HOW A LAB EXPERIMENT IS CONDUCTED**

In this section, procedures for conducting a laboratory experiment are introduced. Table 1 shows a typical procedure for conducting an experiment in a real laboratory. We use the same procedure when we design our online lab.

All the steps listed in Table 1 can be done via the Internet, if a proper interface and structure are defined. In this article, by traditional e-learning, we actually refer to the lessons, courses, or curriculum presented in text, audio, animation, applets, flash, or video clip format. They can run independently or can be downloaded from the LMS to the client machine. The significant difference between the online laboratory and traditional e-learning is that the former also contains the real hardware equipment. The equipment is not independent in such case and cannot be operated without the LMS’s Run-Time Environment support. For a deeper analysis, the use cases of lab-based courses are discussed.

**Use Cases Analysis**

Developing a model for a system prior to its construction or renovation is as essential as having a blueprint for a large building. The use case, borrowed from Unified Modeling Language (UML), is adopted in this article. Similar
to the real laboratory, lab-based e-learning courses must provide the same services for the student. Based on the IMS’s Learning Design Best Practice and Implementation Guide for a virtual lab, the use cases of lab-based courses in the e-learning domain are presented in Table 2.

As the lab-based courses involve use of real hardware equipments, it is possible that unpredictable results or exceptions occur during a lab session. Table 3 summarizes some of the exception use cases.

The proposed design for lab-based e-learning system courses is based on the use cases in Tables 2 and 3.

### PROPOSED ARCHITECTURE FOR ONLINE LABORATORY E-LEARNING SYSTEM

In this section, we introduce the architecture of the proposed online laboratory. Figure 1 shows an overall system architecture. The architecture is based on the SCORM
specification for an e-learning system with additional module (Apparatus-LMS), which deals with functionalities for hardware-based learning systems.

The SCORM specification by ADL deals with the launching, communicating, and tracking of content between the learning resources and the learning management system. It provides means by which learning resources can be reusable and interoperable across multiple LMS/LCMS systems. It consists of the following three components:

- **Learning Resources.** Represent Assets (Web page, JavaScript, XML document, Flash object, picture, etc.) and sharable content object (SCO) (a collection of one or more assets).
- **LMSAPIs.** The communication mechanism between LMS and SCO. These are used for collecting and logging of learning-related data.
- **SCROM Run-Time Environment.** A Learning Management System that manages students and learning events to collate data on learner progress.

### Apparatus LMS Modules

To extend the LMS to the lab-based courses, the proposed Apparatus LMS adds the following modules:

- **Apparatus Virtual User Interface.** App-VUI for short, the remote control panel for the student to control the real apparatus and observe the experiment status during the lab session.
- **Apparatus Run-Time Environment.** App Run-Time for short, the standard and uniform apparatus’ LMS environment, which provides laboratory services for students.
- **Apparatus APIs.** AppAPI for short, the communication mechanism for App Run-Time Environment and apparatus to exchange data. They provide all the necessary functions for initiating a connection to the actual apparatus and for delivering data and control commands.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exceptions</td>
<td><strong>For sys-admin:</strong>&lt;br&gt;• System does not have experiment resources available. In this case the system must:&lt;br&gt;  ➢ Provide an alternative route to another similar equipment or,&lt;br&gt;  ➢ Display some relevant document.&lt;br&gt;<strong>For lab manager:</strong>&lt;br&gt;• The student meets problem and leaves lab session in incomplete state. The lab-manager must:&lt;br&gt;  ➢ Provide the synchronise assistance by instant chat message or online phone calling,&lt;br&gt;  ➢ Reply the student by asynchronous method, that is, e-mail or BBS.&lt;br&gt;<strong>For students:</strong>&lt;br&gt;• Students lost the connection or the experiment equipment is in improper running status. Then:&lt;br&gt;  ➢ Student closes lab session,&lt;br&gt;  ➢ System preserves lab environment for the student to continue the experiment,&lt;br&gt;  ➢ Student continues with other activities within the learning path.</td>
</tr>
</tbody>
</table>

Table 3. Use cases in exception
Web-Enabled Apparatus. The real hardware equipment, which consists of both the hardware and software that controls the hardware.

The most important element of the architecture is the App Run-Time Environment. It provides many standard functions, including lab session management, load balancer, report generator, and multi-user collaboration subsystem. These functions are reusable and shared by all learning objects, which are the experiments in the lab-based e-learning scenario.

MORE DETAILS OF THE ARCHITECTURE

In this section, we present the details of the architecture modules.

Apparatus Virtual User Interface (App-VUI)

To carry out an experiment from the Internet, students need to work in an interactive environment. The App-VUI provides multiple students with a real-time environment to observe and control the experiment. The App-VUI allows the student to issue control commands through the user interface. Meanwhile, the experiment feedbacks, such as the apparatus response and status, are all shown in the App-VUI display area.

To start a lab session, the App-VUI sets up an HTTP connection with the App Run-Time Environment using SOAP (Simple Object Access Protocol) protocol (W3C’s SOAP, n.d.). Once the connection is authorized, the student starts to conduct the experiment.

Communication Using SOAP. SOAP is a lightweight protocol for exchange of information in a decentralized environment. It defines a base communication protocol in order for clients to exchange XML messages with the server. Figure 2 shows the diagram of a SOAP messaging binded with HTTP.

In the proposed system, after the connection is established, the student issues the control commands, such as changing the parameter, reading the apparatus status, and downloading the experiment data. In most cases, the interaction between the student and LMS is a two-phase (request-response) message exchange. The student issues a request message, and the App Run-Time Environment replies with a response message. The response message carries back the status of the initial request (success or failure).
The following example shows a SOAP Message of Query the Apparatus Running Status. In this case, a GetApparatusRunStatus SOAP request message is sent to the LMS.

**Example 1. SOAP Message to Query the Apparatus Running Status**

```
  <soap-env:Header/>
  <soap-env:Body>
    <st:GetApparatusRunStatus xmlns:st="URI">
      <AppRunStatus>isRunning</AppRunStatus>
    </st:GetApparatusRunStatus>
  </soap-env:Body>
</soap-env:Envelope>
```

The following feedback is an example response SOAP message containing the apparatus running status flag.

**Example 2. SOAP Response Message with the Apparatus Running Status Flag**

```
  <soap-env:Header/>
  <soap-env:Body>
    <st:ApparatusRunStatus xmlns:st="URI">
      <Status>Running</Status>
    </st:ApparatusRunStatus>
  </soap-env:Body>
</soap-env:Envelope>
```

The information exchange between the user and the App Run-Time environment is based on XML-Messaging. This will enhance the readability for both the human being and the machine. Meanwhile, it is fully extendable for different experiments in different LMSs.

**XML-Based Messaging.** XML is a markup language for documents containing structured information. Using XML-based messaging, the App Run-Time environment will be interoperable with any XML-based lab equipment. Thus, the lab experiment can be integrated into any LMS in an easy plug-and-play fashion.

To make sure that the LMS server has received the message, a confirmation will be sent back to the client. In such case, a series of confirmation status codes is proposed. Table 4 shows the status codes we used in our system.

### App Run-Time Environment

As discussed in the previous section, the App Run-Time Environment is the key to achieve the e-learning goal in lab-based courses. Figure 3 depicts the proposed model.

There are three types of interfaces for the App Run-Time Environment to communicate with other modules:

- **Message Processor.** Responsible for interacting with the student. Its functions include receiving the control commands and giving feedback on the experiment data and status to the student.

- **AppAPIs.** Provides communication channel for App Run-Time Environment and Apparatus to exchange data. These APIs provide all the necessary functions for initiating a connection to the actual apparatus and delivering data and commands.
LMS Interface. Used to exchange information about the student and learning progress.

The App Run-Time Environment can be considered as a container (or framework) for the Laboratory Learning Objects (LLO). Every LLO that is integrated into the environment can share the resources provided by the framework.

Apparatus APIs. Defined in ADL SCORM Run-Time Environment (Version 1.3, 2004), a common mechanism for learning resources to communicate with an LMS and a predefined language or vocabulary form the basis of the communication. From the specification, only one communication scenario was proposed, which is Learner → LMS. However, the lab-based courses have one more type of communication: LMS → Apparatus. It is a dual communication in such case.

To design a reusable and interoperable lab-based experiment across multiples LMSs, there must be a common way to start, stop, and control the apparatus. Table 5 defines the App Run-Time Environment’s communication APIs in detail.

Normally, the LMS provides the Web interface in order for the student to select the online experiment. Then, the App Run-Time will connect to the apparatus that the student selected via AppAPIs. Once the App Run-Time is

---

**Table 4. Status code and description**

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Condition</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>No Error, successful operation execution</td>
</tr>
<tr>
<td>Communications Failure</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>Connecting with LMS failed</td>
</tr>
<tr>
<td>102</td>
<td>Apparatus is off-line</td>
</tr>
<tr>
<td>103</td>
<td>Disconnected with apparatus</td>
</tr>
<tr>
<td>104</td>
<td>Disconnected with LMS</td>
</tr>
<tr>
<td>105</td>
<td>Sending data time out</td>
</tr>
<tr>
<td>106</td>
<td>Receiving data time out</td>
</tr>
<tr>
<td>Data Errors</td>
<td>Can be extended</td>
</tr>
<tr>
<td>201</td>
<td>Invalided data in some form of format</td>
</tr>
<tr>
<td>202</td>
<td>Invalided data in some form of value</td>
</tr>
<tr>
<td>...</td>
<td>Can be extended</td>
</tr>
</tbody>
</table>

**Figure 3. Proposed app run-time environment**

- LMS Interface. Used to exchange information about the student and learning progress.
connected to the apparatus, then the student
can issue the control commands and receive
the experiment data via the established com-
munication socket.

These AppAPIs fulfill the requirements
for reusability and interoperability. They pro-
vide a standardized method for an LMS to com-
municate with real lab experiments.

**Table 5. Apparatus APIs definition**

<table>
<thead>
<tr>
<th>API</th>
<th>Execution</th>
</tr>
</thead>
</table>
| LOInitComm   | **Description:** LMS initialises the communication with the Apparatus. It
              |   must be executed before calling other APIs.                             |
|              | **Parameter:** The Apparatus’s system ID that are used to establish the
              |   connection with the apparatus. The ID is unique in such case.          |
|              | **Return Value:** Long integer indicates the connection handler if success.|
|              |   Otherwise, return 0.                                                   |
| LOCLOSEComm  | **Description:** LMS closes the connection handler.                       |
|              | **Parameter:** The return value of LOInitComm.                            |
|              | **Return Value:** Boolean. “True” indicts the close action was successful.|
|              |   “False” means some error occurred when close the connection.            |
| LOSendCmd    | **Description:** LMS sends the command to the apparatus.                  |
|              | **Parameter:** The return value of LOInitComm, Command String (defined
              |   in next section)                                                       |
|              | **Return Value:** Boolean. “True” indicts the send action was successful.|
|              |   “False” means some error occurred when sending the command.            |
| LOGetStatus  | **Description:** LMS gets the status of apparatus, that is, running status,|
|              |   apparatus’s indicator status.                                           |
|              | **Parameter:** The return value of LOInitComm, status code (defined by
              |   the different apparatus)                                               |
|              | **Return Value:** String. The status of the apparatus.                   |
| LOLastError  | **Description:** This function returns an error code resulting from the
              |   previous API call. The code can be retrieved many times and will be
              |   kept unchanged until the new API call is made.                        |
|              | **Parameter:** The return value of LOInitComm.                            |
|              | **Return Value:** A pre-defined integer number, error description string.|
|              |   More information refers to the XML-based message.                      |

**Apparatus Queue.** Considering the man-
gagement of a real laboratory, all students who
enrolled in the course share the lab resources
by a scheduled timetable. In this article, the
following two basic and important principles
are refined:
• For a real lab session, only one user is permitted to conduct a particular experiment at a time.
• Any user can conduct a particular experiment in a specified time period.

Based on these principles, we have proposed a first-come-first-served apparatus queuing system with a time-out feature to ensure it is working in the App Run-Time Environment. Any user who wants to conduct a particular experiment must join the apparatus queue. For system identification, a unique UID corresponding to the user will be generated. After the first user who is conducting the experiment is timed-out or terminated, the next user in the queue can start the experiment. The design guaranteed that only one user has the chance to access the apparatus at a time. Meanwhile, an unauthorized user cannot access control of the apparatus due to absence of UID.

**Session Management.** In most cases, the lab technician is responsible for guiding the student to conduct the experiment and to collect the experiment results. In the App Run-Time Environment, the session management is acting in the same role. Sometimes, it provides synchronous assistance (by instant chat message or online phone calling) for the student when the important experiment is in progress and the student has a problem. On the other hand, the asynchronous assistance (by e-mail or BBS) also is included in some unimportant situation. To ensure high availability of lab-based courses to as many students as possible, the session management module is responsible for monitoring the lab sessions in order to do the following:

• Terminate the experiment once time-out occurred
• Reset the apparatus for a new user
• Inform the lab manager in case of any hardware failure

**Load Balancing.** In the real world, lab resources, such as lab schedule, opening hours, equipment, and lab assistants, are always limited. Although lab resources now can be accessed 24 hours a day, 7 days a week through e-learning infrastructure, lab equipment is still limited. Hence, some form of load balancing is necessary. The purpose of load balancing is to increase utilization and enhance apparatus availability. On the other hand, high availability can be defined as redundancy. In this case, a few lab apparatuses that have the same functions and I/O need to be developed and made available in the App Run-Time Environment. So, when several requests come to a same experiment, the system can redirect the request to point to a different apparatus according a prescheduled rule. Also, the whole e-learning system can offer different services for special users, such as researchers and LO testers. It may become a useful and special feature for some commercial e-learning provider.

**Report Generator.** Once the student completes the experiment, he or she may need to download the experiment results and control command data for further analysis or report. In general, these data are all ASCII-based. To enhance the readability for human being and machine, the proposed report generator processes these data according to the different XML schema.

**Multi-User Collaboration.** When conducting the online experiment, students may feel isolated from both the teacher and their classmates, especially when the students have some questions about the experiment or have problems continuing with the experiment. Hence, the App Run-Time Environment must have a mechanism for remote collaboration in order to allow students to conduct the experiment collaboratively. The benefits for the students working together in such a remote environment are the development of teamwork and the enhancement of the learning experience. Collaborative environment is achieved by including conferencing facilities, forum, and chat services to the system. Once the instructor initiates a lab session, many students can join the session. They can monitor the lab experiment.
by receiving the live video streamed by a camera that is pointed to the actual apparatus. At the end of the experiment, all the students in the same lab session can download the experimental results.

**More Modules.** Besides the previous considerations, more useful features can be developed to enhance proposed model usability. Live video, discussion forum, Q&A whiteboard, and e-notification (by e-mail) can be embedded into the App Run-Time Environment. For the convenience of users, a new feature — SMS (short message service) reminder — also is introduced. SMS reminder provides an easy way to remind the students who is in the queue and when the apparatus is idle. Additionally, the administrator can get information about the laboratory by SMS in the area without Internet access. Normally, two ways can be employed to achieve SMS function via either SMS gateway (Internet) or GSM (Global System for Mobile Communication) modem. The second way is adopted in our scheme for the future system upgrade and migration.

**Web-Enabled Apparatus**

Current approaches to design online laboratory courses mainly are divided into the following two groups (González-Castaño, 2001):

- Use of educational simulators, and
- Remote access to real laboratory equipment.

In the first case, software simulations are used to simulate the behavior of real hardware apparatus. In the second case, the real hardware apparatus is used to allow the student to interact with the equipment. The proposed e-learning environment can work with both groups. Basically, the lab experiments are treated as a black box in both cases. These black boxes just provide the interface to communicate with the e-learning system. All data (i.e., apparatus status, experiment feedback) are transferred via the interface.

A Web-enabled apparatus is a real hardware with a controller (i.e., electrical device, medical equipment, optical experiment) that can be connected to the network via TCP/IP protocol. The controller can be any form of embedded or PC-based device with a communication module that enables the connection to the network. For the software part, it executes local control of the hardware and also collects the experiment’s data. As a whole unit, the controller must have the physical interface, such as DA/AD converter or data acquisition card, in order to link with the actual equipment.

Pastor (2003) presented an XML-based framework, **RE mote La boratory Extended (RE-LATED)**, for the development of Web-based laboratories. The idea is to define an abstract entity called RLAB system by an XML DTD so that lab experiments can be described by RLAB. Then, the RLAB schema is published on the server for general use. An interesting research (Nacimiento’s VIML [Nacimiento, n.d.]) proposed a Virtual Instrumentation Markup Language (VIML), which is used to describe location, protocol, and device information for a network of virtual instrumentation devices and/or systems.

**Sample Application**

In the proposed system, XML is used to describe the structure and attributes of the Web-enabled apparatus. As shown in Figure 4, a coupled tank apparatus has been developed. It consists of two small perspex tower-type tanks mounted above a reservoir that functions as storage for water. Two independent pumps pump water into the top of each tank. The ap-
paratus is designed for teaching elementary feedback control principles. A FieldPoint (FP2000) running LabVIEW (National Instruments Corporation, n.d.) functions as the controller for the coupled-tank apparatus. The objective of this experiment is to maintain water levels in the tanks at specified heights. From the control point of view, it can be treated as a Multi-Input-Multi-Out (MIMO) system (in this case, it is two-inputs [pumps] and two outputs [sensors] plant). The definition of the coupled tank is described by XML shown as the following:

Example 3. XML Definition of the Coupled Tank Apparatus

```xml
<?xml version="1.0"?>
...
<AppXML>
  <Apparatus type="LLO">
    <name>Coupled Tank</name>
    <communication>Socket</communication>
    <ipaddr>192.168.0.1</ipaddr>
    <port>2020</port>
  </Apparatus>
  <UniqueID type="LLO-ID">
    <value>app-cpt-01-00</value>
  </UniqueID>
  <Input type="LLO-IN" name="Pump1">
    <property name="type">float</property>
    <property name="min">0</property>
    <property name="max">5</property>
  </Input>
  <Input type="LLO-IN" name="Pump2">
    <property name="type">float</property>
    <property name="min">0</property>
    <property name="max">5</property>
  </Input>
  <Output type="LLO-Out" name="Level1">
    <property name="type">float</property>
    <property name="min">0</property>
    <property name="max">5</property>
  </Output>
  ...
</AppXML>
```

The XML definition file has the following four significant sections:

- **<Apparatus> section.** Gives information of the hardware equipment, such as apparatus name, communication mechanism, and so forth.
- **<UniqueID> section.** A unique number across all LMS systems (may not necessarily be unique across all LMSs, but at least in any one LMS system). In this case, it is dynamically generated by the LMS and will be used for identification, operation, and so forth.
- **<Input> section.** Defines the input variables to be used to accept the control commands. Each input can have an arbitrary list of associated properties, each with a value.
- **<Output> section.** Defines the output variables to be used to send the apparatus data. Each output can have an arbitrary list of associated properties, each with a value.

An XML parser module corresponding to a definition file was developed in App Run-Time environment. Hence, the laboratory experiment can be recognized easily and identified by the e-learning system. The changing, modifying, or upgrading of the experiment can be done easily by editing the XML definition file.

In a normal condition, the controller of the apparatus actually is listening to the request from a TCP port. Once the command is received, verification will be processed to make sure the format and value of the command are correct and legal. Otherwise, the command will be discarded. As we can see, it is easy to have personalized learning objects. They can provide a different learning content or sequence to the student according to his or her learning experience.
This App-VUI allows the student to experiment with different controllers, such as Manual, ON/OFF, PID (Proportion-Integral-Derivative) control, and to study the effect of adjusting the various parameter settings available in the controller. As this is a relatively slow process, the water levels and the control signals to the pump can be charted on the App-VUI as the experiment progresses. The student also can download all data collected by the (NTU’s OnlineLab) server for further analysis when the experiment has been completed.

CONCLUSION

E-learning is having a significant and positive impact on education. In order to understand how theoretical knowledge can be applied to real-world problems, experimental practical exercises are essential. In this article, we have proposed the architecture of an online laboratory e-learning system. The purpose is to combine the researches in Internet-based laboratory and e-learning infrastructures. We have provided the details of the modules. The system is based on XML messaging, which provides extensibility and plug-and-play fashion. We also have provided one example of the implementation. We are now conducting further studies to improve the performance of the system in terms of delays, efficiency, and scalability. The main important feature of our system is that it easily can be interoperated with any e-learning system that follows the standards.

REFERENCES


Duan Bing received his BEng in the School of Communication Engineering from Chong Qing University (China) in 1999. He was a database administrator from 1999 to 2001 in China Telecom and software engineer in Singapore. He is now pursuing his master degree in ICIS, School of Electrical Electronic Engineering, Nanyang Technological University of Singapore. His major research interests are focused on e-learning framework for lab-based subjects, XML-based instrumentation markup language, and network traffic analysis for real-time control.

Dr. Habib Mir Hosseini received his BSc in electronic engineering from Isfahan University of Technology (Iran) in 1988, his MSc in digital electronics from Sharif University of Technology (Iran) in 1991, and his PhD in electrical engineering from the Adelaide University (South
Australia) in 1998. He joined Iranian Telecommunication Research Centre (ITRC) as a design engineer in 1988. In 1992, he joined Behineh Pardaz Company as Software Engineer. Since December 1997 he was a research fellow with the Network Technology Research Centre (NTRC), School of EEE, Nanyang Technological University of Singapore, and he later joined the School of Electrical and Electrical Engineering of NTU in August 2000 as an assistant professor. His research interests include multimedia encryption and copyright protection (digital watermarking), pattern recognition, e-learning models and document understanding and recognition.

Dr. Ling Keck Voon obtained his DPhil from University of Oxford, United Kingdom and BEng from National University of Singapore. He is currently an associate professor at the School of Electrical & Electronic Engineering of Nanyang Technological University, Singapore. His research interest is in control and automation.

Professor Robert Kheng Leng Gay obtained his BEng (1965), MEng (1967), and PhD (1970), in electronics engineering from the University of Sheffield. Professor Gay is the director of IT research infrastructure, and also the director and CEO of the Managed Computing Competency Centre (MC3) of Nanyang Technological University, Singapore. He was a member of the team that was awarded the LEAD Award for CIM (Leadership, Education, and Application in Development of Computer Integrated Manufacturing) by the Society of Manufacturing Engineers, USA in 1992. His recent research interests are focused on advanced knowledge-based systems, e-learning, agent-based technology, and industrial IT.