# Software Technologies, Architectures and Interoperability in Remote Laboratories

Mehmet Efe Özbek, Ali Kara
Electrical &Electronics Eng. Dept.
Atılım University
Ankara, Turkey

Musa Ataş
Barla Computer
Ankara, Turkey

***Abstract*— A remote laboratory aims to provide access to remote lab equipment over the internet. The design and development of the client software is subject to various requirements. In this paper we discuss these requirements and propose solutions. Implementations are demonstrated for some remote experiments developed in the ERRL project. An XML format for experimental results is proposed as an attempt towards standardization of remote laboratory interfaces and enhancing the interoperability of diverse remote laboratory applications.**

***Keywords*—remote laboratory, on-line learning, ATML**

## I. INTRODUCTION

Several software architectures and technologies for remote laboratories have been proposed and implemented over the last years. Each solution has its advantages and disadvantages. Organizations usually choose and adopt one solution based on their needs, previous experience, available software infrastructure, available development tools, skills and expertise of developers employed.

The design and development of remote laboratory software is subject to different requirements. Besides supporting the didactic strategy of the remote laboratory, the software application should also satisfy some technical requirements.

In this paper we discuss and propose solutions to some technical requirements in the design and implementation of software for remote laboratory applications. We concentrate on a remote experiment that has been developed within the scope of European Remote Radio Laboratory (ERRL) project supported by the Leonardo da Vinci Program of the European Commission. ERRL project aims to enable learners access expensive radio frequency equipment which cannot be afforded by many educational institutions, via internet.

## II. REQUIREMENTS

In this section we describe the requirements that led to the design of the software prototype for the remote experiments in ERRL project. While these requirements originated from an analysis of particular experiments, many of them are applicable to the majority of remote laboratory experiments.

### *A. Interoperability*

One important merit of a remote laboratory is the flexibility that it provides to the trainees in getting laboratory experience at anytime and from anywhere. The remote laboratory client software is supposed to operate in a vastly heterogeneous environment. Different browsers, plug-in or operating systems may be present at the client side. Therefore operability across a wide variety of hardware, operating systems and web browsers is an important criterion. Solutions adopted in a remote laboratory application must be supported by various client platforms to a large extent.

### *B. LMS Integration*

A Learning Management System (LMS) is an indispensable tool in e-learning. Among its functions are online delivery of the content, management of learners, keeping track of their progress and performance in training activities and etc. [1]. By incorporating an LMS in a remote laboratory solution, interactive features of LMS are made available to the users. LMS integration is an important requirement to be addressed in development of the remote laboratory application software [2]. A typical use case where the need for interoperability between the remote laboratory application and the LMS becomes evident is the tracking and assessment of learner's performance. The remote laboratory software application should facilitate automatic assessment and tracking of the learners' performance in laboratory activities by the LMS.

A large number of commercial or open source LMS alternatives are available and different organizations adopt different LMS for their e-learning initiatives. Among the have been developed to facilitate interoperability and reusability between LMS's, the Sharable Content Object Reference Model (SCORM) is most widely used [3]. Conforming to standards in the development of a remote laboratory can improve the portability of the application to different e-learning platforms.

### *C. Access Limitations*

Users of a remote laboratory may be accessing the lab servers through low bandwidth internet connections. In such cases a software solution which requires a high bandwidth will inevitably result in serious user inconvenience.

The *same origin policy*, which aims to prevent *cross-domain scripting* fraud, is implemented in almost every client-side platform [4]. By this policy, a client application is not allowed to access a remote laboratory server if it originates from a different server. This may be a serious obstacle for remote laboratories with distributed servers.

### D. Resource Sharing

A remote laboratory may provide access to laboratory equipment which are otherwise not available. Efficient sharing of the lab facilities among many users is important especially when the laboratory equipment is very expensive and limited in number, as in the case of a radio laboratory.

### E. Educational Issues

Educational requirements such as Tracking and assessment of learner performance are handled by the LMS. The client software application for a remote laboratory should facilitate the assessment of the learners' performance in laboratory activities.

## III. Proposed Solutions and Implementation

### A. Interoperability and LMS Integration

In ERRL project, Moodle has been the LMS of choice. Moodle is an open source LMS and supports the SCORM standard. The learning content of the pilot experiment has been designed to be SCORM conformant for easy integration into any SCORM conformant LMS. The content has an active part which is responsible for data retrieval from the workbench server, presentation of the experimental results to the user in an interactive manner and storage of experimental data in the local file system.

During the design of the software architecture we have experienced a fundamental shift in our decision about client-server partitioning. The initial tendency was to implement a server-side application which would consume the laboratory web service to retrieve the experiment results. Presentation of the retrieved data to the learners would be left to the client application. In order to avoid repeated logging-in of the learners both to the server application and to Moodle, an authentication scheme was designed where the server application would obtain the learner's credentials by directly accessing Moodle's database, after being launched by a dynamic link generated by Moodle. This approach was abandoned later, since it turned out that the database structure was significantly different in every version of Moodle, and such a server application would not be portable even among different Moodle versions.

Furthermore, solutions which make use of server-side scripting and database access are not covered by SCORM standard and are not guaranteed to be portable across different LMS platforms. Therefore we have decided to implement a java applet client which retries data from the laboratory web service and handles the presentation and storage of the data as well. In this way all the functionality has been put in the client application. There were several reasons for preferring a Java applet. First of all, a Java applet, when combined with good programming practices, offers portability to virtually any client platform. It can be enclosed in a SCORM content package and can communicate with the LMS using SCORM API to store the learner's progress information or to access the data about the learner.

### B. Overcoming Access Limitations

Many solutions to overcome the limitations imposed by the 'same origin policy' have been applied by web programmers and most of them are mere workarounds that open security holes in the system. In an ideal solution, the access limitations imposed by the 'same origin policy' should only be removed after establishing a trust between the user and the target site which the user is attempting to access.

Signed applets provide a way to verify that the applet is downloaded from a reliable source and can be trusted to run with the permissions granted in the policy file. In ERRL, the 'same origin policy' limitations have been eliminated by using signed java applets. When activated, the applet asks the user whether the source of the applet is a trusted identity and access to the remote laboratory servers is only permitted after the user's confirmation.

### C. Resource Sharing

Two broad categories of remote experiments are "Interactive experiments" and "batched experiments" [5]. Interactive experiments are those which can be monitored and controlled by the user during its execution. It must be noted that considerable delay may be introduced by slow internet connections and full interactivity is not always possible. Unless their duration is very short, interactive experiments should be scheduled, since only one user can control the experimental setup at a time.

Batched experiments are those in which the entire course of the experiment can be specified before the experiment begins. Most experiments can be designed as batched experiments or as consisting of several *batched steps*. These batched steps can be queued for execution on the server allowing many users to submit their request simultaneously. If the execution times of the batched steps are sufficiently small, the users may feel as if they are performing the experiments interactively. In this way efficient use of the lab can be maximized.

The remote experiments are controlled and executed in the physical laboratory of the ERRL system by the so-called *work bench servers*. The detailed design considerations for the development of the server applications will be reported in a separate paper. The servers can handle multiple client requests at the same time by queuing them in a first-in-first-out basis. The consequent improvement in the utilization efficiency of lab resources is very important since ERRL system is comprised of very expensive equipment. Also the user inconvenience associated with scheduling is avoided in this way and the laboratory is made available to all the users "from anywhere and at any time". Experience has shown that the latency introduced by queuing is not very significant since a typical batched job is completed in not more than 10 seconds on the average and the arrival of many requests to the workbench server at almost the same time in not very likely.

## IV. Web Services Technology for Remote Laboratory Applications

An important part of the work on remote laboratories has been devoted to the development of an infrastructure in which distributed laboratories can be made available to the on-line access of diverse users (see for example [6]). In recent remote laboratory projects solutions based on Web Services technology seems to be preferred [7]. The main characteristic of Web Services is the interoperability with different software platforms. Web Services can be developed easily on diverse Server platforms and client interfaces can be realized in different software environments such as Java Applets, Flash or AJAX . Data is marshaled into XML request and response documents and moved between software packages using HTTP and SOAP protocols. The transparency of Web Services technology makes this an obvious choice for remote laboratory experiments. The lab side services may need to run on a different hardware and software platform than the client-side student software [8].

Web services technology facilitate resource sharing among institutions by providing an open architecture where many institutions can access the laboratory resources with diverse client software applications tailored to their own technical and didactic requirements. Intuitions that own expensive laboratory equipment can expose their facilities to open access using Web Services.

A major advantage of Web Services technology over other traditional remote access protocols, such as RPC, CORBA, and RMI, is its use of the SOAP protocol which can tunnel through firewalls. It may be argued however that SOAP protocol may increasing loose this advantage in future since this feature does not express a balance between security and functionality but simply comes from the fact that today's firewall's see SOAP messages as "Web page requests" rather than powerful application messages. As this protocol is increasingly used to send procedure calls to internal applications to servers within an organization, firewalls may increasingly filter its use, since an underlying goal of firewall security is to block external access to internal functionality to prevent a potential intrusion [9]. In the course of ERRL project we have experienced a case where a user was unable consume the laboratory web services on a remote server from within the campus of a partnering institution. This has been attributed to the presence of a SOAP filtering mechanism since the user could establish HTTP communication with the same server to visit some web pages.

The wide-spread use of Web Services technology will establish an excellent basis for interoperability of remote laboratories as it provides seamless and automatic connections from one software application to another. Major software development platforms support Web Services technology making it quite easy to implement a Web Services interface for a remote laboratory application. Similarly, consuming web services is a straight forward task with every kind of programming language and development tool. Once the HTTP address for the WSDL definition of a web service is provided, proxy classes are automatically generated by development tools. The use of proxy classes are not much different than local objects but they make the all the communication with a laboratory server transparent to the developer.

## V. The Need to Supplement Web Services Technology

Web Services solve the syntactic interoperability problems by making diverse systems capable of communicating and exchanging data easily. However issues regarding the semantic interoperability still remain. We use the term semantic interoperability as the ability to automatically interpret the information exchanged meaningfully and accurately in order to produce useful results as defined by the users of both systems. To achieve semantic interoperability, all parties must refer to a common information exchange reference model.

Semantic interoperability is closely related to how XML technologies are used in remote laboratory applications. XML is actually a meta-language for describing markup languages and specifies no predefined tag set and therefore no preconceived semantics. XML only provides a facility to define tags and the structural relationships between them. In Web Services infrastructure, programming language objects are serialized into XML strings before they are transmitted to the other end. If no other specifications exist, data types and objects are converted into XML elements using the encoding rules laid in section 5 of the SOAP specification [10]. Alternatively, mapping of language classes into XML elements can be done in a preconceived way by defining an XML schema which determines the structure of the serialized XML string. An XML schema may be designed for defining a formal agreement between a service and a client that describes how the data is going to be exchanged.

Currently there is no common convention about how the measurement data would be serialized before it is conveyed to the client by the web service and each remote laboratory application uses its own style. Without supplementary explanations it is quite difficult to figure out how to programmatically consume this service and develop an associated client software. Many ambiguities may be observed by a consumer of the service, such as the physical meaning of the measured quantities, measurement units used, devices used in the experiment etc. The lack of a common, agreed-up-on XML based format not only impairs the interoperability of remote laboratory applications but also impedes code and design reuse since utilizing every different remote lab service requires a new custom client software development. If a specific markup language could be developed for data exchange, it could be possible to develop generic software components which would simplify client software development.

In ERRL project an attempt is made towards standardization of remote laboratory web service interfaces, by defining an XML based format for describing experimental data. An excellent source has been the IEEE's "Automatic Test Markup Language (ATML) [11] for this endeavor. ATML is a collection of XML schemas that allows test information to be exchanged in a common format adhering to the XML standard. IEEE's ATML working has defined standards for to increase interoperability between different Automatic Test System (ATS) software applications and test platforms. The ATML working group has defined the data formats for each of eight ATS related areas; Test Results, Diagnostics, Test Description, Instrument Description, Test Station Description, Test Adapter Description, UUT Description, and Test Configuration.

In the laboratory web services of ERRL system the XML format used for the description of experiments results is similar to the definition of test results format in ATML standard. The proposed ExperimentsResults schema which specifies this format (not included due to space problems) is generated by borrowing elements from IEEE's TestsResults Schema [12].

TestResults schema provides data elements to capture identifying information for: the unit under test, the test station, the test program (via reference to a Test Description document); ambient environmental conditions at the time of the test, test equipment calibration data, test program input data (i.e. parameters) and ancillary textual comments.

The *ExperimentsResults* schema references the *Common.xsd* schema [13] of ATML standard which provides unique types and attributes for all ATML schemas. The motivation for the use of *Common.xsd* schema was the fact that it addresses the need for standardization in automatic test systems, a domain which is a very similar to remote laboratories in terms of the types and attributes used. In *Common.xsd* the primitive types such as *double* and *integer* are extended to include the *unit* property besides the *value* property.

In Appendix B, the experiment results for the ERRL's scattering parameters (S-parameters) remote experiment are described as an XML document which conforms to the *ExperimentsResults* schema. Some of the optional elements specified in the schema are not included in the document in order not to clutter the Figure. Inspection of this XML document quickly reveals the self-describing nature of the format that has been used.

The proposed format can be used for saving the remote experiment results to local or remote file systems. A standard way of serializing experimental data into a text file makes the exchange of experiment results much easier. For example after performing a remote experiment, a learner can save the results in this standard format and submit to the teaching staff for evaluation or to demand their help with a problem. The results can viewed using an compliant reader, a general purpose XML viewer or even a text editor since the format is XML based and therefore human-readable. In case an LMS is being used for course management activities, the client application can be designed so as to upload the experiment results file in response to an online assignment created on the LMS.

The object returned by the S-parameters experiment web service is an instance of this class populated with measured values and additional metadata. Serialization of the object into XML as well as its transmission is handled by the .NET platform. At the client side the received XML message is desterilized into a Java object with a similar structure by the java run time engine. Across all mature development platforms XML serialization and deserialization for the purpose of exchanging instances of data types is transparent to the developer and requires little or no work. Some immature development tools with limited support for Web Services , however, may only provide functions for XML node processing. In this case stripping the SOAP message and parsing will have to be accomplished by lower level programming.

## VI. CONCLUSION AND FUTURE WORK

In this paper some technical for remote laboratory software development have been discussed. The proposed solutions have been demonstrated on an example remote experiment of the ERRL project. An XML based format is proposed for description of experiment results which may enhance the interoperability of diverse remote laboratory applications by supplementing the Web Services technology. As future work we plan to develop an XML schema as a standard for describing the experimental setup of a remote laboratory. Another XML schema for describing an experiment including the experimental procedure, test objects, measurement devices and input parameters can be designed. We hope that such studies will serve to increase interoperability of diverse remote laboratory applications.

The ERRL remote experiments can be reached at the web address http://errlmoodle.atilim.edu.tr

## REFERENCES

[1] S. Rapuano, F. Zoino, "A Learning Management System Including Laboratory Experiments on Measurement Instrumentation", *IEEE Trans. on Instrumentation and Measurement*, vol. 55, pp. 1757–1766, October 2006

[2] S. Kolberg, D. Courivaud, M. E. Özbek "LMS and Interactivity - Technical Issues for Remote Laboratories", *The 18th Annual IEEE Int. Symposium on Personal, Indoor and Mobile Radio Communications* (PIMRC'07)

[3] Sharable Content Object Reference Model (SCORM) 2004 2nd Edition Addendum Version 1.1, Advanced distributed Learning, http://www.adlnet.org.

[4] Cross-Domain Scripting Issue, Version: 1.0, Advanced Distributed Learning, http://www.adlnet.org

[5] "iLab: A Scalable Architecture for Sharing Online Experiments" *International Conference on Engineering Education October 16–21, 2004, Gainesville, Florida.*

[6] F. Davoli et al., "LABNET: Towards Remote Laboratories With Unified Access", IEEE *Transactions on Instrumentation and Measurement*, vol. 55, no.5, October 2006

[7] A. Baccigalupi, C. De Capua, A. Liccardo, "Overview on Development of Remote Teaching Laboratories: from LabVIEW to Web Services", IMTC 2006 Instrumentation and Measurement Technology Conference, Sorrento, Italy 24-27 April 2006

[8] V. J. Harward et al "The iLab Shared Architecture:AWeb Services Infrastructure to Build Communities of Internet Accessible Laboratories", vol. 96, no. 6, June 2008 Proceedings of the IEEE

[9] C. C. Albrecht, "How Clean is the Future of Soap? Communications of the ACM, vol. 47, no. 2, February 2004

[10] http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[11] http://grouper.ieee.org/groups/scc20/tii/

[12] http://grouper.ieee.org/groups/scc20/ATML/TestResults.xsd

[13] http://grouper.ieee.org/groups/scc20/ATML/Common.xsd

## VII. APPENDIX B. XML DOCUMENT FOR S-PARAMETERS EXPERIMENT RESULTS

```
<?xml version="1.0" encoding="UTF-8"?>
<ExperimentResults xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://errl.atilim.edu.tr/XML/ExperimentResults
ExperimentResults.xsd"
 xmlns:c="http://www.ieee.org/ATML/2006/Common"
 xmlns="http://errl.atilim.edu.tr/XML/ExperimentResults"
 uuid="b9afa2d05aa74493b8e8b3d7105a716e"
 name="ERRL">
<ResultSet >
   <Experiment name="Scattering Parameters">
      <Description>
         Scattering parameters of the device measured in ERRL lab. using remote
experiment facility
      </Description>
      <Parameters>
      <Parameter name="StartFrequency">
        <Data>
          <c:Datum xsi:type="c:double" value="1.0e9" standardUnit="Hz"/>
        </Data>
      </Parameter>
      <Parameter name="StopFrequency">
        <Data>
           <c:Datum xsi:type="c:double" value="12.0e9" standardUnit="Hz"/>
        </Data>
      </Parameter>
      <Parameter name="NumberOfPoints">
        <Data>
          <c:Datum xsi:type="c:integer" value="21" />
        </Data>
      </Parameter>
      <Parameter name="IFBandwidth">
        <Description>
          IF bandwidth is set to the default value since no value is specified by the
requester
        </Description>
        <Data>
          <c:Datum xsi:type="c:double" value="40.0" standardUnit="Hz"/>
        </Data>
      </Parameter>
    </Parameters>
    <ExperimentResult name="S11">
      <Description>
        S11 Parameters of the test object in polar form. Array in ExperimentData
node contain freqency, magnitude and angle in each row.
      </Description>
      <ExperimentData>
        <c:Collection>
          <c:Item name="Frequency" >
            <c:IndexedArray  dimensions="[11]" xsi:type="c:doubleArray"
standardUnit="Hz">
              <c:Element position="[0]" value="1e9" />
              <c:Element position="[1]" value="2e9" />
              <c:Element position="[2]" value="3e9" />
            </c:IndexedArray>
          </c:Item>
          <c:Item name="Magnitude">
            <c:IndexedArray dimensions="[11]" xsi:type="c:doubleArray"
standardUnit="dB">
              <c:Element position="[0]" value="10.0" />
              <c:Element position="[1]" value="10.3"/>
              <c:Element position="[2]" value="12.5" />

            </c:IndexedArray>
          </c:Item>
          <c:Item name="Phase">
            <c:IndexedArray dimensions="[11]" xsi:type="c:doubleArray"
standardUnit="rad">
              <c:Element position="[0]" value="0.3" />
              <c:Element position="[1]" value="0.4"/>
              <c:Element position="[2]" value="0.5" />

            </c:IndexedArray>
          </c:Item>
        </c:Collection>
      </ExperimentData>
    </ExperimentResult>
  </Experiment>
</ResultSet>
</ExperimentResults>
```