Web Service Based Remote Development Environment for an e-Learning System

Ivan Madjarov\textsuperscript{1}
Abdelkader Betari\textsuperscript{2}
Zohra Bakkoury\textsuperscript{3}

\textsuperscript{1} Laboratoire des Sciences de l’Information et des Systèmes (LSIS) - UMR CNRS 6168, Université d’Aix-Marseille
Domaine Universitaire de Saint-Jérôme
Avenue Escadrille Normandie-Niemen
13397 Marseille Cedex 20, France
Telephone: +33 4 91 17 79 17
Fax: +33 4 91 17 79 22
E-Mail: ivan.madjarov@iut-gtr.univ-mrs.fr

\textsuperscript{2} Laboratoire d’Informatique Fondamentale de Marseille (LIF) – UMR 6166 CNRS, Université de la Méditerranée
163 Avenue de Luminy case 920
13288 Marseille cedex 9, France
Telephone: +33 4 91 17 79 20
Fax: +33 4 91 17 79 22
E-Mail: abdelkader.betari@lif.univ-mrs.fr

\textsuperscript{3} Université Mohammed V – Agdal, Ecole Mohammadia d’Ingénieurs
BP 765, Agdal, Rabat, Maroc
Telephone: 00 212 37 77 19 05
Fax: 00 212 37 77 88 53
E-Mail: bakkoury@emi.ac.ma
Web Service Based Remote Development Environment for an e-Learning System

ABSTRACT: The objective of e-Learning is to improve the quality of education not to replace traditional ways of teaching. Our work is centred on the development of an e-Learning environment by proposing semantic supports based on XML standards and technologies for creation, presentation and storage of learning objects (XESOP project). This paper presents an essential element in an e-Learning system: the possibility of running remote exercises on the Web. The solution put forward uses Web Services for the animation of computer science exercises in a programming environment developed by the author of the course. The exercises are accessible for an on-line and real time execution without having a development tool or a compiler on its computer.

KEY WORD: e-Learning, Interactive e-Learning, Web Services, XML.

1 Introduction

There exist several research and development projects in e-Learning field. The existing e-Learning platforms are similar to each other by their functionalities and their multiple extensions. That is due especially to the fact that standardization organizations (LOM, IMS, SCORM) concentrate their efforts on structuring and re-use of the teaching documents by defining a basic element called Learning Object (LO) [ADL-SCORM, 2004]. On the other hand, the problems of contents of these objects and re-use of the applications functionalities, which animate them are not treated.

The main objective of e-learning is to improve the quality of teaching and not to give a substitute to traditional modes of education. There are various complementary and independent means to achieve this goal: access to various resources, remote tutoring and communication tools, exercise resolution, remote collaboration.

E-Learning projects can be classified in four categories [Bouthry A. & Jourdain C., 2003]:

- Tools for assisting teachers who use the network to prepare and follow a traditional course. That relates to the multiple courses and exercises put on line in static HTML format;
- Tools that make knowledge available through portals giving access to learning documents. They are, in the majority of the cases, content management systems (CMS) able to generate courses and exercises upon request;
- Remote guides defining teaching paths. These are more complex e-Learning systems and platforms which concentrate units of contents creation, follow-up and evaluation of the results and define for some implementations the learning path according to the results of the learner. This characterizes in general the learning content management systems (LCMS);
- Courses integrated into the management devices of competences and knowledge where the e-Learning comes in support to management, making available the knowledge of the company and also comes in support to the development of competences. This characterizes in general the enterprise e-Learning systems and some e-Learning portals.

Our XESOP project proposes a study and solutions on technological means of e-Learning environments. The results of this project can be situated between “the means to make knowledge available” and “remote guides”. The essential elements of the project consist on one hand on semantic supports for creation, presentation and storage of learning objects based on XML standards and technologies [Madjarov I. & Betari A., 2004] and on the other hand on
means for the student learning path management [Madjarov I. et al., 2004] and the execution of remote exercises.

Being an essential element in e-Learning course, the interactive learning object exercise is equipped with a remote development environment. For an exercise on algorithms and programming, for example, this means that the learner would be able to carry out the remote exercise without having a development tool or a compiler on its computer.

This paper presents our motivation and a possible scenario of implementing a technology and the use of our work by presenting the side of the teacher and the side of the learner.

The paper is organised as follow: in section 2 we are interested in creating and diffusing existing relevant educational material; section 3 is concerned by the design of remote development environment; in section 4 we study how to integrate the RDE operation with Web Service and section 5 presents our conclusions and future work.

2 Motivation and scenarios

In the current e-Learning systems, the creation of quality didactic material remains a difficult task. One of the aspects that allow reducing this complexity is related to the process of didactic material authoring for e-Learning. This operation can be more efficiently carried out if it is handled as a cooperative process. In this process, a team of educators are brought together to design, author and deploy the courses. This allows benefiting from the technologies, standards and available open source products in order to build more efficient e-Learning systems.

Our work meets an existing need in any educational institution where collaboration between several teaching entities is essential. To maintain a certain level, already acquired, the involvement in classroom learning (c-Learning) of several teachers is required, but is not always possible.

2.1 The usual practice

The usual practice shows that teachers rarely exchange information on the contents of their courses, even when they are similar. There are certainly many reasons to this; some are purely of technical nature, and others of psychological nature. The technical reasons concern primarily the format of the teaching documents. Those were created in the past by different computing means incompatible with current formats, and thus are not compatible with more recent documents and e-Learning standards. Others assert that the teacher may think that by sharing his course, he will lose its contents or he will be evaluated by his peers. The technical reason put forward is certainly not the main obstacle. With our work we try to improve the technical "atmosphere" by hoping that a collaborative practice would lead to an improvement of the psychological "atmosphere". E-Learning systems and platforms in their essence induce sharing knowledge not only with learners, but also collaboration within teaching teams. E-Learning is by definition a complex and collaborative process.

Our studies located another problem. There exist on Internet a considerable number of on line courses. The teacher is able to evaluate qualities of these courses to advise them with the students like a complementary support. Their direct use, on the other hand, is difficult especially because of their low compatibility or in certain cases total incompatibility with the standards of e-Learning and more precisely with SCORM. That makes difficult their re-use in various lessons and practically impossible their use in learning content management systems (LCMS), where the structure of the contents is subjected to a strict standardization.

2.2 A generic and standardized environment

Our research work in e-Learning field [Madjarov I. & Betari A., 2004] [Madjarov I. et al., 2004]
and our experiment and participation in several projects brought us to the following conclusion: teachers are in need of a generic and standardized environment, related to new technologies supporting the exchanges, such as XML technologies. The design of such an environment can be based on the existing standards treating the structure of a learning document (SCORM, IMS, LOM) [IMS-CP, 2004]. For example, using an XML semantic editor, the teacher can, independently of his colleagues or with their collaboration, create the contents of his course using the learning objects of LOM [IEEE-LOM, 2003]. That generates the tree structure of a generic learning document and a validation grammar of XML Schema type [W3C-XSD, 2001].

Figure 1 shows the aspect of developed XML semantic editor. Authors can see three views of their documents: the source view showing the XML code, the structure view displaying the DOM tree, and the third view shows a formatted representation of the edited document.

![Figure 1: View of XML, MathML, and SVG editors](image)

It is rare that the contents of a course are only made of raw text. According to the courses, the teacher may need to present diagrams, mathematical formulas or data in tables. It is well recognised that document elements such as figures or tables can be more concise than a long discourse and have high pedagogical value [Vercoustre A. M. & McLean A., 2003].

For these purposes we found around XML technologies practical meta-languages for the design of mathematical expressions editor (MathML) [W3C-MathML, 2003], or for the creation of vector graphics editor (SVG) [W3C-SVG, 2003] and schema for table producing (Figure 1). For the creation of multimedia presentation the teacher would need an SMIL editor [W3C-SMIL, 2005]. For the student progressation evaluation the teacher would need a QTI editor [IMS-QTI, 2004] for the creation of questionnaires. In this case XML [W3C-XML, 2001] is used for encoding non-textual information [Quint V. & Vatton I., 2004] such as vector graphics, mathematical expressions, synchronized multimedia documents, complex forms, etc.

Formatted texts, diagrams, formulas, tables and still images constitute basic structure of most learning materials. In our system, the binary data of multimedia contents is embedded directly into the course XML data itself. As a result, the course collection can be managed easily since all materials relating to the course are stored in a single XML file [Ampornaramveth V et al., 2004]. During the editing process, if the user inserts an image or any binary data into the edited content, the semantic editor (Figure 1) will encode it using Base64 encoding method (RFC2045, RFC3548). The encoded binary is surrounded by XML tags marking its filename, content type
and encoding method. For example, `<image filename="name" type="gif|jpg|png" code="Base64"> ... </image>` tags can be used for inserting an image into the course content. Currently three image formats are supported: JPG, GIF, and PNG. The Base64-encoded binary content of the image (in Figure 2 - `java-boutique.gif`) are placed between the XML tags as shown in the example in Figure 2.

```xml
<image filename="java-boutique" type="gif" code="Base64">
R0lGODlhlwBxAOYAAIuLjGOYzJeh0/r6+XFxcVBQT6fC6PSyaM/LAFyHr
XClzv/HAP7mAlkWAdjY2T1SLYI+OA4aXYyou66v7ZAHFvCKOjpDIXMsTExP
m4U15pKNPd/xANvxxzMuCP3txba6yziVh/zMVMbQ/2md0e.................................
………………………………………………………………………………………………………
………………………………………………………………………………………………………
……………3Rrb+MY4zvEfAgEAo==
</image>
```

Figure 2: Base64 representation of image binary data in course XML document

In Figure 3 is present the structure of created course with the XESOP project’s semantic tools. It is in conformity with SCORM standards.

![Figure 3: The course structure](image)

The content aggregation specification [ADL-SCORM, 2004] comprises two models: a metadata model specifying the metadata elements of learning resources, and a content packaging model representing content structure. Both of them are hierarchical, which is convenient for representing data consisting of many elements and sub-elements. XML is perfectly suited for representing hierarchical models, as exemplified by the LOM [IEEE-LOM, 2003] and content packaging XML binding specifications published by IMS [IMS-CP, 2004], both of which are adopted in SCORM.

Thus created course can be saved as an XML file and shared simply by making local copies, or using HTTP and WebDAV protocols in the distributed Internet environment [Pinheiro M. K. et al., 2001]. That does not imply the creation of a course warehouse, ready to be shared by several teaching entities. According to the evolution of the course, teachers may need to modify its contents. Therefore, the correct operation of a collaborative system for course creation imposes the storage of learning documents. The use of a database contributes to a better re-use and diffusion of these documents.

The choice of an appropriate database is essential: we have chosen a native XML database which allows the storage of XML documents in their origin format, without mappings. Our
choice, in opposition to that of a relational database, is explained by the nature of learning documents which are in general of narrative types i.e. document-centric and not data-centric.

Although relational database products today provide built-in XML document and query support, native XML databases are arguably the best choice for metadata storage. As far as query language is concerned, IMS recommends XQuery [W3C-XQuery, 2005], but in current implementations, XPath [W3C-XPath, 2005] is more frequently used. Though it is considered that XPath does not have enough expressiveness to function as a database query language, it continues to be popular and could be easily replaced with XQuery whenever necessary. In our work we chose the open source native XML database Xindice [Apache-Xindice, 2003] which support the XPath query language.

To achieve maximum flexibility, XML is used for internal representation of the learning material. A large number of media-rich contents can be stored using the abundant set of XML schema available. Also by providing proper XSLT transformation files [W3C-XSLT, 2005], the content in XML can be presented in many other forms, such as HTML for web-based learning, PDF for printing [Apache-FOP, 2004], etc (Figure 4).

A course is intended to a group of students which is not always homogeneous by its level of knowledge. The case of a course of algorithmic and programming is relevant. The teacher in computer science is confronted to problems of various levels for various options followed by a group of students. It is obvious that this type of course must be adapted to the required level. That implies in our system the possibility of extracting (by simple marking) selected parts of a course from the database for contents tailored to the needs of each case (slide presentation of a course in SMIL format, Web pages for on-line courses, PDF or RTF files for later printing).

The flexible nature of hypertexts and on-line materials offers new opportunities and challenges for learning support that can guide the learner in a more personalised way. In particular, when the content is split into smaller units (Learning Objects), the learning system is expected to provide some guidance as to which part to read next [Vercoustre A. M. & McLean A., 2003]. The dissemination of learning collections from the database uses Web Services and parsers which are part of XML technologies (Figure 4).

![Figure 4: The learning content management in XESOP project](image)

**2.3 The interactive learning object exercise**

Integral part of a course, the exercise brings a learning pedagogy and a follow-up of knowledge acquisition. The existing e-Learning systems provide different means of exercise construction in the form of questionnaires of a QTI type. Most e-Learning systems also include tools to verify
that the student has effectively learned what it was supposed to learn. The student can be asked to answer a few questions, to solve a problem or to write a program [Brusilovsky P. et al., 1996].

These kinds of exercises miss interactivity and immediate feedback to the student. In our conception, the exercise is an exclusive learning object that ensures interactivity and an immediate answer. The idea, developed by Knuth in 1992, is that the documentation used for learning a programming language should include active examples of what the language offers. By active we mean examples that the user can test and get results that are immediately included in the embedding document [Knuth D., 1992].

For a teaching in computer science the author of the course defines the exercise (in programming) and its place in his presentation. The course published by a Web server, becomes available to the students via a simple Web browser. On the site of the exercise, an applet framework appears. It is a "remote development environment" (RDE), carried out in the client side to reach the resources of an application server. The real time execution of the exercise is associated to a tool based on a Web service. The significance of this environment resides in the independence of the student from his operating system while bringing an effective solution to the steps of setting up, configuring compilers and licenses managing.

Within the framework of teaching in programming according to the language, a copy of the interpreter or compiler for the student’s operating system must be installed and configured. He will also need a text editor preferably with syntax colouring (syntax highlighting) of the code for a better legibility and comprehension of its content. In [De Bra P. & Calvi L., 1998] they discuss the use of colour annotations and link hiding to provide better guidance to the student. Finally, he has to know the basic options of commands of these tools. That also requires some knowledge of the operating system for files, directories and user account management.

All these "minor problems" which are quite cumbersome and do not relate to the essence of the course pedagogy, can distract the student and divert him from the objective of the learning. This is the case when the student must abandon the course to seek the integrated development tools recommended by the teacher. If he does not find the right version or if the installation of the software is incomplete it is likely to be confronted to messages and uncommon problems. These tools can in some cases prove to be inaccessible because expensive, or impossible to install on a given machine, that is because of a constraint on resources, operating system or other constraint. This brings up the need for a tool which will enable us to avoid these difficulties and improve the remote learning environment.

The solution brought by RDE solves this kind of problems. In its first version this solution is provided with basic functionalities compared to a traditional programming environment available in a classroom (like Eclipse, JCreator or NetBeans), but brings a better synchronization with the pedagogy of the course compared to a free installation of a programming environment (distributed on CD or downloaded from Internet).

3 Design of remote development environment

The remote development environment (RDE) is a programming environment intended on the one hand for the authors of courses and on the other hand to the students. The teacher can define an exercise within the framework of his course and the student can carry out this exercise while pursuing his course. For students in computer science for example, this tool proposes a set of basic functionalities.

3.1 Basic functionalities

1. A source code editor with syntax colouring schemes to recognize keywords, classes and functions for several programming languages, such as C, C++, Java, Perl, etc;
2. Exchanges with the local system (clipboard), to carry out the exchange operations of course contents and the exercises;

3. A program runner written with the text editor to recover the standard output and error (stdout and stderr) devices of operating system;

4. The environment interacts in real time with the user by showing the output of the program execution and by connecting the standard input (stdin) of the user machine with the standard input of the program.

The tools of this environment are developed in Java as applet (light client). They are configurable with a minimum of parameters and can be launched under several remote learning platforms.

### 3.2 Implementation strategy

On a technical level of implementation, we noticed that Java applets are not often used on the Net. The major inconvenience that is often put forward is the slowness of loading or running some of them; moreover, compatibility problems exist between Web browsers. These difficulties come from the various JVM (Java Virtual Machines) available according to the versions or types of Web browsers.

For obvious security reasons, the applets are not entirely Java applications. In fact, the applet has not access to local file system and cannot establish its own network connection. The applet has on the other hand the ability to react to actions from the mouse, the keyboard or the operating system.

The problem of security does not arise in our design, since it is not expected to develop a client application for the presentation of a remote course. Thus the use of a standard and free Web browser appeared judicious to us because, it is the main technology used by e-Learning platforms. Therefore the development and the use of an applet to carry out a light client are fully justified.

Program execution without the user having development tools becomes possible by the creation of a Web Service [Chauvet J.M., 2002] which will aim to provide the necessary functions to compiling and executing these programs. The development tools (compilers and interpreters for various programming languages) are placed on a Web Server. This technology is open and is developed for the standard protocols on the Internet (SOAP, TCP/IP).

In this way programs written in various programming languages and various platforms can communicate. The realization of the Web service in our work is based on Apache project (Web Server/Tomcat Application) [Apache-Tomcat, 2004] with Java programming interfaces [Loy M. et al., 2002].

### 3.3 Client-server interactions and Web services architecture

Web services enable “partners” to re-use easily applications via the Internet. A Web service is essentially an interoperable stand-alone software component that has a unique URI (Uniform Resource Identifier) and that operates over the Internet and particularly the Web [Snell J. et al., 2001]. Web services are commonly based on standards; currently, the most common ones are the XML-based specifications SOAP (Simple Object Access Protocol), UDDI (Universal Description, Discovery and Integration), and WSDL (Web Services Description Language).

SOAP is a technology for sending messages between two systems. SOAP uses XML for representing the messages and HTTP is the most common transport layer for the messages. UDDI is a specification for an online registry that enables publishing and dynamic discovery of Web services. WSDL is an XML representation that is used for describing the services that are registered with a UDDI registry.
SOAP is a protocol for message transmission. It defines a set of rules to structure messages which can be used in simple one-way transmissions, but it is particularly useful to carry out request-response communications RPC (Remote Procedure Call). It is not related to a particular communication protocol and it is not either related neither to an operating system nor to a programming language [Chauvet J.M., 2002].

The approach adopted in our project for a SOAP development (Figure 5) is to encapsulate the code contents intended for this service, in a Java method and then, to start a process which listens to the messages addressed to this service (listener). SOAP messages enclose the service name and the required parameters. The transport layer is HTTP protocol. The listener decodes entering SOAP message and converts it into a call method. Then it gets back the result and encodes it in a SOAP message answer as shown in Figure 5.

![Figure 5: SOAP development approach](image)

The implementation of this concept in the case of our service passes by writing the service method in Java which connects itself to the application able to process the transmitted code. The listener process is a Java Servlet running as a server application. The client is a standard Web browser. Our implemented model is presented in Figure 6.

![Figure 6: The Web service model for exercise execution](image)

Implementation on the client side (the applet) requests a conscientious investigation of the details of SOAP serialization and HTTP encoding. To this end we use a SOAP package by extending its functionalities to ensure the applet steadiness and the learner interaction with the system. We call the service using the appropriate method of SOAP package by specifying the URL of the service, its name and all the necessary parameters.

The server returns an encoded HTTP message enclosing SOAP answer. On the client side, we use the same SOAP package to carry out the opposite operation. The package decodes HTTP message and extracts SOAP-XML message, then deserialize it to obtain the return value.
4 The RDE operation

Configuring parameters of the Web service are saved in an XML file which contains necessary information for program execution in each specified language. As an example we present the configuration for programs processing under three languages: C/C++ and Java.

4.1 XML configurations tag

The root element of the configuration document is `<coursconfig>`. It may contain one or more elements `<language>`. Each element has one attribute `name` for the name of the language supported by the service. A language definition may contain one or more elements (`<step>`) each one describing the command line for code processing. The `<command>` tag defines the command line (`name = "lcc"`) by a relative or absolute path to the compiler on the server side. The `<file>` tag defines the file name with attribute `id`, the file extension with attribute `extension = "cpp"` and mandatory contents by the attribute `hasContent = "yes"`. The case of C/C++ languages contains three "step" (stages) to carry out. The first one defines the source code compilation starting from a "cpp" file. The second one defines assembly process in "obj" files and creation of the executable file. Finally the third one defines the program execution. In case of errors in compilation or in the following "step" the process is stopped by the attribute `stopOnErrors = "yes"`.

```xml
<?xml version = "1.0"?>
<coursconfig>
  <language name = "c">
    <step stopOnErrors = "yes">
      <command name = "lcc"/>
      <file id = "filename_of_C_source" extension = "cpp" hasContent = "yes"/>
      <file option = "-Fo" id = "fichierObj" extension = "obj"/>
    </step>
    <step stopOnErrors = "yes">
      <command name = "lcclnk"/>
      <file id = "fichierObj"/>
      <option name = "-o"/>
      <file id = "fichierExe" extension = "exe"/>
    </step>
    <step>
      <file id = "fichierExe"/>
    </step>
  </language>
</coursconfig>
```

The example below - for programs written in Java language - illustrates the operation of the attribute `folderid = "testFolder"` of the `<file>` tag, which indicates to Web service to place the processed file in a separate directory.

```xml
... <language name = "java">
  <step stopOnErrors = "yes">
    <command name = "javac"/>
    <file id = "FichierJava" name = "test.java" folderid = "testFolder"
```
4.2 The signature procedure

We already mentioned that the use of an applet on the client side is subject to several security constraints. An applet evolves within Web browser and cannot have direct access to the system resources unless exception related to the procedure of "signature" of access rights. In our case, we need to reach network and system resources of the client computer. We add to this the need of file creation on the server side. These possibilities will be available provided that the client accepts the signature upon request from the browser.

The signature procedure is a technique carried out by Java packages with tools for encrypted keys creation for certificate each connection of the client (keytool). The signature itself is done by signing the archive .jar file where the classes accessing the authorized resources (jarsigner) are located.

Figure 7: The source code applet-editor

To facilitate this procedure we created a tool named genkey. Therefore the users will be able to create in a database their own keys. Client identification is then performed starting from this database with the username and password.
Figure 7 presents the screen that the student gets after accepting the applet signature. Program code editing and its execution are accessible by the menu-functions from the applet-editor.

5 Conclusion and future work

In this paper, we presented an e-Learning environment mainly based on open tools and standards. This allows obtaining low cost systems applicable in almost any real world situation and deployable at a large scale. Besides this, the proposed environment allows to take advantage of modern technologies brought through the latest Web standards. We have shown how XML has been used: in XESOP semantic editor, MathML and SVG plug-ins for learning objects creation, in native XML databases for storing course collections and in SOAP protocol communication for application re-using. The authoring part of this environment is designed to offer a support for authors producing collaboratively e-Learning materials through Web services.

We presented in this paper the remote development environment (RDE) for running the learning object exercise in an e-Learning system. RDE extends interactive learning objects and makes it possible to students to follow various teachings without having a development tool or a compiler on its computer. It is an original and open source solution which represents an example of Web services implementation in an e-Learning system for exercises simulation. Existing solutions are generally based on DHTML technologies (Javascript), CGI (ASP, PHP) [Park U. et al. 2004] or Servlets (JSP) and do not offer a synchronized interaction.

The presented work is still under construction and can be developed for a generic tool, in order to be able to handle exercises of any type like questionnaire which is based on QTI technology (Question & Test Interoperability) of IMS [IMS-CP, 2004]. Some features are in a more advanced development stage, like semantic editors, native database management, remote exercise editing and running, Web service exchanges, and others are at their initial stage like SMIL multimedia creation tool.

The structure and the design of our work are compatible with LOM specification. Its conformity with IMS standards will not therefore raise any problem to its integration in a project based on LCMS (Learning Content Management System).

Our work provides tools for the creation and integration of interactive learning objects and courses into a LCMS. A Learning Content Management System facilitates the access to the courseware and monitors the learners’ interactions with the instructional material, while offering them pedagogic advice by means of the intelligent learner modelling system.

It is being used regularly by a group of test staffs at our institute, and will serve as a test bed for research on development of on-line learning environment in the future.

6 Bibliography

ADL-SCORM [2004], Advanced Distributed Learning, The SCORM Content Aggregation Model, On-Line: http://www.adlnet.org/scorm/
Web Service Based Remote Development Environment for an e-Learning System


W3C-SMIL [2005]. Synchronized Multimedia Integration Language (SMIL 2.0) - [Second
Web Service Based Remote Development Environment for an e-Learning System


