Applying XML-based technologies to developing online courses: the case of a prototype learning environment

Joanna Jedrzejowicz and Jakub Neumann
Institute of Mathematics, University of Gdansk, Gdansk, Poland
Email: Joanna.Jedrzejowicz@math.univ.gda.pl, jj,kuba@math.univ.gda.pl

Abstract

Purpose – This paper seeks to describe XML technologies and to show how they can be applied for developing web-based courses and supporting authors who do not have much experience with the preparation of web-based courses.

Design/methodology/approach – When developing online courses the academic staff has to address the following problem – how to keep pace with the ever-changing technology. Using XML technologies helps to develop a learning environment which can be useful for academics when designing web-based courses, preparing the materials and then reusing them.

Findings – The paper discusses the benefits of using XML for developing computer-based courses. The task of introducing new versions of existing courses can be reduced to editing appropriate XML files without any need for program change and an author can perform this task easily from a computer connected to the internet. What is more – using XML makes it possible to reuse data in different teaching situations.

Research limitations/implications – The environment has only been used for two years and further research is needed on how user-friendly the system really is and how it can still be improved.

Practical implications – The paper describes the environment which can be used to develop and reuse online materials, courses, metadata etc.

Originality/value – The paper offers practical help to academics interested in web-based teaching.

Keywords: Internet, E-learning, Authoring languages

Paper type: Research paper

1. INTRODUCTION

This article on XML technologies is mainly based on our experience within the project WebCoCT – Web-based Courses Creating Tools (Berg et al., 2003). The aim of the project was to develop a learning environment to introduce web-based courses. At all stages of the project we made extensive use of the XML technology. Here we describe our perspective on using this technology. The article is structured as follows: We start with an overview of XML language and XML applications used in education; then we describe WebCoCT – an application based on XML. Finally the benefits of integrating XML into educational systems are discussed.
2. XML LANGUAGE

The World Wide Web Consortium released the XML recommendation in 1998. The initial proposal for XML (Extensible Markup Language) was based on an already existing standard called the Standard Generalized Markup Language (SGML) used for document processing. It appeared that SGML was too complex to be used on the Web and therefore XML was a successor (Carey and Blatnik, 2002).

XML was designed not as a special-purpose language but as a generic metalanguage, which allows different user groups to define their own specific languages. Some of these languages are domain-oriented and define exchange data formats within an industry sector, science or another community. For example, Tutorial Markup Language (Daum and Merten, 2003) defines an interchange format designed to separate the semantic content of a question from its screen layout or formatting, and MathML is an XML application for describing mathematical notation and capturing both its structure and its content (Daum and Merten, 2003). Other languages are task oriented and act as languages for specific technical or application-oriented tasks. For example, DocBook (http://www.docbook.org) is an XML vocabulary for computer documents.

The base technology is the same for all of these languages; they differ as to tags and their attributes. Thus they are easy to learn and use. The other advantage of using these languages is that they can be processed with the same basic tools. Transformations between these different languages can be achieved by using a special XML based language for document transformation – XSL (Extensible Style Language).

XML is a markup language. This means that an XML document can be structured with the help of syntactical markup elements similar to those used in HTML. Identifiable elements within a document are enclosed between start and end tags. The word “extensible” in the name of XML stands for the possibility of introducing users’ own tags and attributes. This is in contrast with HTML where all those elements are predefined. Using a markup language makes it possible to create documents of an arbitrarily complex structure. Each document is organized as a tree structure with a distinguished root element. In the example below <tutorial> is the root element (see Figure 1):

Notice that by using the semantic markup, it is possible to make the document self-describing. What is more, the designers of XML intended that user groups, industry associations and other communities would agree on certain document schemes so that each XML document can be tied to a certain type. At present, two main formats of schemes are used: Document Type Definition (DTD) and XML Schema. If any of these is supplied with an XML file, then a validating parser can check the document structure against the scheme.

DTDs were a standard way to define a scheme up to 1999 when the XML Schema working draft was first published. Now, most XML communities are using XML Schema, which is an XML document itself. Thus XML tools can be used to edit, validate, parse and transform it. Besides, XML Schema copes with some of DTD deficiencies – the Schema supports namespaces, scheme modularization and reuse. It allows one to introduce basic data types as used in programming languages as well as user-defined types. The XML Schema for the above document can be defined as follows:

From the XML Schema (see Figure 2), it follows that each tutorial is a sequence of any number of quizzes. A quiz can either be of TrueFalse type or MultipleChoice. In the first case, the tutorial contains a sequence of questions, each with a required attribute, called “answer”
which returns the logical value of the question (that is true or false). In the case of a MultipleChoice quiz, the XML document contains questions with a choice of possible answers. In order to identify each quiz, the document is equipped with the attribute “id” which is unique. Other attributes such as “name”, “description”, “language” are not obligatory.

The document from Figure 1 can be validated with the Schema from Figure 2. It contains one quiz of each type. The TrueFalse quiz is defined as one false question and MultipleChoice quiz contains one question with three suggestions of answers, of which only the first one is true.

Document transformation is an important issue for XML. Since XML allows the storage of information in a presentation neutral format, it is necessary to transform XML documents into a presentation format such as HTML, PDF or other. The XSL (eXtensible Stylesheet Language) specifications define how XML documents can be transformed into another format.

3. XML APIs

The acronym API stands for Application Programming Interface. An API is used to develop an application based on the given technology. An API also describes a common way of interfacing the application with that technology.

The main tool used for processing XML documents is a parser. Parsers can read XML input from different
sources and recognize tags, attributes and other XML language elements. Parsers can also validate the document. The two most popular APIs for manipulating XML documents are the SAX and DOM parsers (McLaughlin, 2001).

SAX means Simple API for XML. SAX was developed by the members of the xml-dev mailing list. It is an event-based parser, which can process the document sequentially: generate an event for each encountered XML language element; and call appropriate handler function. A programmer using this API defines the handler functions. A major disadvantage of this approach is that SAX does not support random-access manipulation of the document – the programmer sees the element once, in document order. It is also not possible to modify existing documents via SAX. However it does not require much memory and is easy to learn and use.

DOM stands for the Document Object Model and is being developed by the World Wide Web Consortium. The DOM API provides a standardized, versatile, tree-like view of a document's contents. It allows programmers to navigate the document tree, retrieve content, modify, delete or add new elements. After processing, the resulting document tree can be written to an output stream. In contrast to SAX, DOM stores the complete document structure in memory and is relatively heavy on resources. This API is very rich – DOM knows 17 different collections of functions (called interfaces) related to some XML language elements.

These APIs are different in provenance, in scope, and in programming style, and thus each has strengths and weaknesses. Almost all modern programming languages provide at least one of these standard XML APIs.

XSL Transformation Language (XSLT) is a declarative language, based on DOM, and represents a tool for processing XML documents on a high level.

4. WEBCoCT

The aim of the WeBCoCT project (Berg et al. 2003; Brekiewicz et al. 2004) was to develop a learning environment to introduce web-based courses. The project makes use of the ideas introduced in (McCormack and Jones, 1998) and an earlier project described in (Jedrzejowicz, 2004). The system is designed as a three-tier architecture where the course design functions are separated from the authoring tools and teaching functions.

The presentation tier of WeBCoCT can be accessed by two groups of users, the authors of courses and students. The database tier contains the XML Schema files describing the course and the course parts suited for several forms of learning, such as tests, tutorials, interactive exercises, past exams, etc.; metadata containing information about the author; domain of the course, coverage, educational objectives and XSL files responsible for the presentation of each part of the course. The third tier, the XML processing tier, contains tools for documents parsing and transforming. It is possible to exchange components in this tier and use other tools, if desirable.

WeBCoCT runs under the Linux operating system and makes use of web server Apache and the system Postgres for the database capabilities. In the implementation process the mark-up languages HTML and XML as well as programming tools and languages – PHP, Java, Java Script – and XML processors were used (McLaughlin, 2001). A SAX-like parser from the PHP language was also used. DOM was not available for the PHP version when WeBCoCT was being created. Now it is strongly recommended to use DOM for similar tasks. SAX was not flexible enough for manipulating XML documents contents.

The database accompanying WeBCoCT contains the information on users and courses for administration purposes. It allows the administrator to restrict access to authoring tools only to a group of authors, and parts of courses only for students with logins and passwords – some parts of courses can be reached without any limitations.

During an author's session to create a new course—after a successful authorization—the author designs the form of the course, which requires inputting the following data:

- metadata describing the course,
- number of tests and their type,
- test questions and answers,
- names of files containing tutorials and/or past exams,
- data for interactive exercises.

The information on the new course is entered into the database. The system generates XML files containing the entered information in real time. In the next step, the XSLT processor generates HTML pages using appropriate XML and XSL files. The numbers 1, 2, and 3 on Figure 3 mark the consecutive steps of course creation.

HTML pages are later rendered to students during the teaching sessions. Some additional files are also generated. For example, for each test a file with correct answers is generated which is useful for the evaluation of user answers.

The users-students have access only to selected HTML pages, which display suitable parts of courses. In the dynamic parts of courses, such as tests and exercise, there are comments and evaluation messages displayed, as well.

5. BENEFITS OF USING XML

The benefits of using XML for developing computer-based courses have already been documented in several projects (Lucke et al., 2002; Wiest and Zell, 2001). XML
is a standard that allows the definition of the structure and semantic features of data and information. Another advantage is portability. What Java did for portability of code, XML claims to do for data. In the case of WebCoCT, as was already mentioned, XML Schema files are used to describe the logical structure of courses and their parts: tests, exercises, and tutorials. This allows for the distinction between the content of a document in an XML file and its layout and allows to strictly enforce the structure of documents. Furthermore as the documents are marked-up with tags, tools can automatically parse the documents to check whether they are well formed and valid with strict conformance to the structure constrained by XML Schema. And this takes place in WebCoCT as shown in Figure 3.

The task of introducing new versions of an existing course can now be reduced to editing appropriate XML files without any need for program change. For example, replacing an XML file containing a test by another one, will result in offering students new opportunities for self-testing. What is more, an author can perform this task easily from a computer if it is connected to the Internet. Probably the main advantage of using XML is the possibility to reuse data in different teaching situations. Using the abstract XML description of a course enables new authors to compose individual courses from this material and introduce new features. For example, exam questions can be generated from tests and exercises. The same applies to preparing teaching materials (handouts, slides, presentations, etc.) used in face-face education alongside online teaching.

6. CONCLUSION AND FUTURE WORK

The data regarding the effectiveness of using the system WebCoCT for creating new on-line courses is limited due to the relative newness of the system and a limited number of users. So far the system has been used for creating one course on databases and one on formal languages.

In the case of the database course, the use of user-friendly tools allowed the group of authors to increase, since upgrading exercises is simple and easy. An author can either edit or extend existing exercises, or introduce new courses. In either case, an author works in a graphic environment, filling out forms, that enables the system to define an XML document. For example, when the author wants to define an exercise teaching SQL, he has to define the following:

- the task for the student,
- SQL command (usually it is not unique), which performs the task,
- and probably a hint, which is optional.

During an authoring session, the author has online access to all the views and tables created for exercises, as well as dictionary tables. This helps authors to set-up an exercise, introduce new tables if necessary, and check answers before submitting the answers into the system. The author only needs to enter the data into the system: the description of the exercise (metadata), the task for the student, a hint (if necessary), the correct answer, names of tables in the database, if applicable. The session is concluded with the generation of an appropriate XML file, as shown in Figure 3.

In Figure 4 we show the structure of an exercise and in Figure 5 two examples of suitable XML documents are displayed. To help authors with no experience, we can assume that doc1.xml is provided during the authoring session and used to create doc2.xml.

The second course on formal languages contains topics on finite automata and regular languages. The idea
behind exercises is similar – to make most of the inter-
activity. The tasks for users are of type: define an automa-
ton such that..., describe the language of the automaton given
by the graph, convert the given automaton into an equivalent
one which... etc. The answers for these types of questions
need not be unique; therefore it is not possible to check
their correctness by the literal comparison of suitable
strings. However, there are well known procedures for
checking the equivalence of two finite automata, or reg-
ular expressions (Hopcroft, Ullman 1979). Thus during
the authoring session, the teacher provides one of
possible answers and during the teaching session the
equivalence test is performed on the student’s solution.
This feature of regular expressions and finite automata,
which allows to check the equivalence of two given
objects, is very attractive for computer based learning
because it allows to make full use of interactivity.
Unfortunately, for more complicated automata, like
pushdown or Turing machines the equivalence problem
is not decidable and no such algorithms exist.

WebCoCT has now been used for two years. The gath-
ered experience proved that using the XML technology
allowed the creation of tools, which are easily extensible
and versatile. Further experiments of implementing new

DOC1.xml:
<?xml version="1.0" encoding="ISO8859-2"?>
<?xml-stylesheet type="text/xsl"
href="Exercises.xsl"?>
<Exercise
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ex.com/namespace
Exercise.xsd">
<FileName>exer1</FileName>
<Question>
<Number>1</Number>
<Task>Create the table Employee which
contains the following attributes:ID,
name, surname, dept, insur. The
attribute insur is boolean, all the
other are strings.</Task>
</Question>
</Hint>Use CREATE TABLE command
</Hint>
<Correct_answer>Create table
Employee(id primary key, name
varchar(20), surname varchar(20), dept
varchar(10), insur boolean)
</Correct_answer>
</Exercise>

DOC2.xml:
<?xml version="1.0" encoding="ISO8859-2"?>
<?xml-stylesheet type="text/xsl"
href="Exercises.xsl"?>
<Exercise
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ex.com/namespace
Exercise.xsd">
<FileName>exer2</FileName>
<Question>
<Number>2
</Number>
</Task>Create the view EmplIns which
contains names, surnames, dept of those
from table Employee who are insured.
</Task>
</Hint>Use CREATE VIEW command
</Hint>
<Correct_answer>Create view EmplIns as
select name, surname, dept from Employee
where insur='t'
</Correct_answer>
</Exercise>

Figure 4 The structure of an exercise

Figure 5 Two exercise documents

Jedrzejowicz and Neumann: Applying XML-based technologies to developing online courses
courses with the existing software are planned. Also new developments are considered:

• following the ideas of several researches (Kotsiantis and Pintelas, 2004), (Sobecki, 2003) on implementing Machine Learning algorithms to manage the feedback from users-students and using the students’ profile make suggestions concerning further work for single students;
• to set up a subject-independent template for exercises so that introducing exercises for a new course will only require developing a subject-specific module to check student’s answers; and
• performing a survey among users (authors) on how user friendly the system really is and how can it be still improved.

REFERENCES


Hopcroft, J. E. and Ullman, J. D. (1979), Introduction to Automata Theory, Languages and Computation, Addison-Wesley Publ. Co., Reading, MA.


WEB SITES

DOM: http://www.w3.org/DOM
API: http://en.wikipedia.org/wiki/API

Joanna Jedrzejowicz holds a PhD in Computer Science. She works as a professor in the Institute of Mathematics, University of Gdańsk, Poland. She has published widely in computer science conferences and journals. Her research interests include theoretical computer science and computer-based learning. She conducted research and was a co-author of a project and implementation of a computer-based system for teaching the theory of automata and formal languages – the system is still used for teaching and research. Recently, she took part in a project WebCoCT dedicated to developing tools for creating web-based courses.

Jakub Neumann holds a Master’s degree in mathematics and works as an assistant in the Institute of Mathematics, University of Gdańsk, Poland. His research focuses on theoretical computer science, in particular various logics and tree automata. He has a special interest in XML-based web technologies. He took part in the WebCoCT project dedicated to developing tools for creating web-based courses.