Implementing XML-based Product Manuals*

Zsolt Barczikay
Department of Measurement and Information Systems,
Budapest University of Technology and Economics,
Muegyetem rkp. 9 Bldg. R., H-1521 Budapest, Hungary,
barczy@mit.bme.hu

Tamás Mészáros
Department of Measurement and Information Systems,
Budapest University of Technology and Economics,
Muegyetem rkp. 9 Bldg. R., H-1521 Budapest, Hungary,
meszaros@mit.bme.hu

ABSTRACT: The management of the electronic information is a very important field nowadays. Relational database systems (RDBS) are widespread solutions, but text-based documentation, like product manuals, could not be squeezed into rows, columns, and cells. The Extensible Markup Language (XML) is a new standard that describes text documents in a structured format. It is based on the SGML ISO standard, and was primary designed for the World Wide Web. In this paper we introduce how web-based electronic product manuals can be build using XML technology. After a short presentation of the XML components, we concentrate on the main advanced features in product manuals using XML. Finally, we present a prototype system for implementing XML-based product manuals.

1. INTRODUCTION

An ongoing research project at the Budapest University of Technology and Economics targets the creation of Intelligent Product Manuals (IPM) [1]. The project involves four universities: University of Wales, Cardiff (UK), Budapest University of Technology and Economics (BUTE, Hungary), Technical University Clausthal (Germany), University of Rousse (Bulgaria), and two industrial partners: Excel Csepel Manufacturing Ltd from Hungary, and Sparky PLC from Bulgaria.

The aim of the IPM project is to develop a methodology for providing of intelligent decision-support throughout the product life cycle, from product installation, through its maintenance and diagnosis to its dismantling and recycling [2]. The project staff at BUTE is working on the prototype IPM of Excel-Csepel SL 320/600 CNC turning center [3].

The main objective of the design of the architecture is to define a common information framework for integrating all product-related data stored in various electronic formats, integrating knowledge-based and multimedia systems to use and present these data and finally to develop IPM prototypes.

Taking the needs of industrial partners into consideration the architecture should support simple and affordable IPM solutions as well as an enterprise-wide information framework for integrating all available data and tools.

An important field of intelligent product manuals is to effectively manage electronic documentation. A suitable technology was needed that can be the basis of the document management system, and has suitable features and applications for building product manuals. XML has proven to be a good choice for this purpose.

2. IPM ARCHITECTURE

The main innovation of the project lies in the application of state-of-the-art techniques and tools. It utilises existing standards and tools from all possible fields. It was also a project rationale that the incorporated techniques and tools should be widely accessible and supported.

The following general architecture has been proposed:

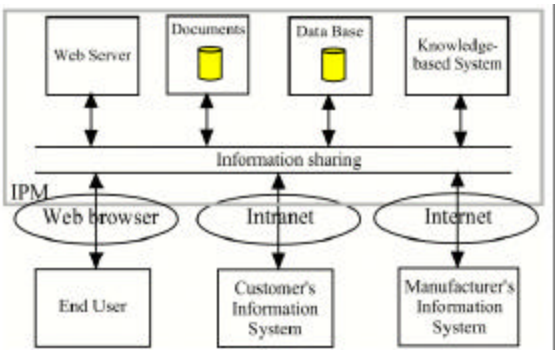

Fig 1. IMP architecture

The system utilises web technology to provide an easy-to-use multimedia user interface, document storage and access, and communication methods. Knowledge-based modules are added to perform more complex tasks, for example machine diagnosis.

The operation of the IPM can be categorised into three main operation modes:

- **Standalone IPM.** A low cost solution, which runs on a multimedia PC and provides basic documentation and knowledge-based tools from a CD-ROM. It also can be portable.
- **Intranet-based level.** This level is a multi-user environment containing the full documentation, knowledge-based tools for diagnosis and supervision, and databases of product related data. It is integrated with the customer's information management system.
- **Internet-based level.** This level connects the IPM to the manufacturer's information system and to other external sites. This supports additional features like automatic document updates, product history monitoring, and information retrieval from external data sources.

* This work was partly sponsored by the OTKA F030763 research project

3. XML TECHNOLOGY AND METHODS

The World Wide Web Consortium (W3C), responsible for web standards, had worked out the Extensible Markup Language [4], and brought out as a recommendation in 10 February 1998. XML is a meta-language, a language to describe applications-specific languages for representing text documents. It plays two important roles.

The XML language is very similar to the Standard Generalizes Markup Language (SGML), which is an ISO standard for electronic document management since 1986. The main advantage of this technology is that it provides an opportunity to plan the structure of the information, and to store the data according to the developed structure. It can be used for processing text-based information. The application of the SGML standard dates back several decades. It was primary used by the large-scale industry [5].

XML has plenty of advantages for using in electronic manuals:

- Long lifetime of the syntax: XML is vendor independent – we can define a document language based on XML that does not depend on particular document editing software.
- Structured: we are able to build structured text documents, and use structured queries to retrieve information from the documents.
- Content validation: XML has techniques to automatically validate the content of a document based on the document type definition.
- XML documents are reusable: we are able to take out, or change the required parts, without reading the whole text.
- We can regain the invested energy of document design many times at the side of the utilization.

XML, as a meta-language, can help in defining languages and describe the information content of a document. We can define the available elements and the hierarchy between them. The formal definition of elements, their usage and relations, is called Document Type Definition (DTD) [6].

XML is a markup language. It stores not only text fragments, but also their types. The stored string of the electronic markup is called tag. The information is stored between starting and ending tags, as shown in the example below.

```
<title>Accuracy Test Reports</title>
<testdata>
   <product>
      <productname>EXCEL CNC LATHE</productname>
      <producttype>SL-320/600</producttype>
      <identifier>EM1234567890</identifier>
   </product>
   <testman>Mr. John Smith</testman>
   . . .
</testdata>
```

As the example shows us, the XML language does not specify the appearance of the text. It is entirely separated from the content. The W3C defined its own technology rules called eXtensible Stylesheet Language XSL [7] and XSL Transformation XSLT [8].

Structuring a document means to analyze the sense of its content. We have to examine what kind of content elements can we distinguish, what are the characteristics of these elements, and what are the relations between them.

For instance the following text is a part from the product manual. Let us try to see it through the glasses of the XML.

```
The opening of the chip conveyor is 280 mm
wide, while its height of ejection is 1200 mm.
```

In this sentence it is worth to mark up the physical units together with their numerical values. So we are able to query the technical information about the chip conveyor, and the software systems will be able to process these values and use them in other calculations. We can mark up the information in the following way:

```
<P>The opening of the chip conveyor is
<Measure><Value>280</Value><Unit>mm</Unit>
</Measure> wide, while its height of ejection is
<Measure><Value>1200</Value> <Unit>mm</Unit>
</Measure>.</P>
```

The planning of the documentation structure consists of a collection of similar ideas, thoughts and requirements. We have to take into consideration the separable content elements and possible logical structure among them.

The next step is to formulate these requirements in a formal way to create the document type definition. The following example shows the DTD of our example for measures:

```
<!ELEMENT P (#PCDATA | Measure)* >
<!ELEMENT Measure  (Name? , Value , Unit) >
<!ELEMENT Name  (#PCDATA) >
<!ELEMENT Value  (#PCDATA) >
<!ELEMENT Unit  (#PCDATA) >
```

We can notice that we have to define the name of the tag and describe the content. `#PCDATA` means that the content is a text. It also states that the `Name`, `Value` and `Unit` elements can not be further divided. The `Measure` element consists of the name of the measure, the measured value and the unit. The '?' symbol at the end of the name specifies that the element occurs once or never. The `P` element contains simple text, or the `Measure` element. The '|' symbol specifies the 'or' relation, and the '*' symbol at the end tells us that this structure can occur several times. In other words the measure can appear anywhere in the text.

When the DTD is ready, we can create the document with an XML editor, like XMetal [9]. XML editors use the DTD, and force us to keep the rules of the document structure.

4. THE SKELETON OF AN XML BASED SYSTEM

The XML-based product manual can be integrated at various levels into the main architecture. In this paper we only concentrate on the internet-based level, as this is the most typical application area at the moment.

After we had created the documentation in XML format, the documentation managing system had been designed to present it to the users (document authors and end users as well).

There are numerous ways to access and present the information stored in XML documents. There are simple document viewers that can show the documentation to the

user. In addition to conventional document viewers, these systems have advanced presentation facilities to display the documentation in different ways, using different formats, and filtering the required parts. They have advanced filtering and search capabilities based on the document structure and markingups.

We can also create programs to access and handle the information stored in XML format. This opens a wide spectrum of possible applications. These applications can access the documentation in a similar way to how access databases. They can form queries and evaluate them over the documentation.

In the IPM project we used small program modules to build our prototype system. Some of these modules are based on freeware APIs implementing basic XML functionality. During the development of these modules, our primary goal was to demonstrate how to query the content, and how to find marked text elements. Last but not least the XML modules had to suit the main IPM prototype, which is client-server based, and can be accessed via the internet.

The server stores text, figures, numerical, and other data about the product. The user sends a query to the server, that processes the query using XSL filtering, generates a web page, and sends it back to the user in HTML format. At clients' side a standard web-browser is used to present the results to the user.

Figure 2 shows the scheme of the XML subsystem. At client side we use a standard web-browser, like Netscape Navigator, or Internet Explorer to access the web pages of the product manual. The browser provides the opportunity to read the manual, and also to submit a query form. To formulate our request, we can write the name of the tag, or give a query expression in the XPath language [10]. This case is for demonstration only, since the user has to know the structure of the documents and tags used in the documents. The query form contains the list of available tags to help the user.

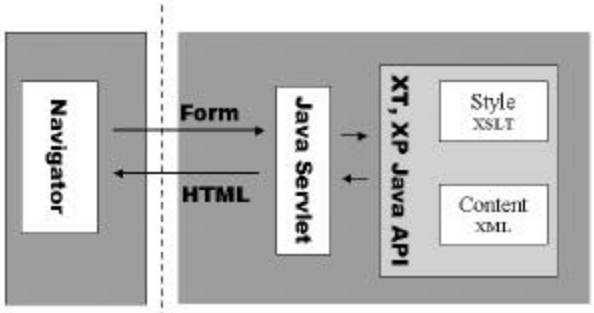

Fig 2. Prototype XML system architecture

The XML Path language (XPath) is another W3C technology to describe the location path of elements in XML based document. XPath treats the structured document as a tree, and one can navigate in this document tree. We can use relations like child, parent, ancestor and descendant. The leaves of the tree are text elements of the document. Using multiple relations in a chain we can address any marked piece of the text.

5. COMPONENTS AND TOOLS

The XML module was implemented using the Java programming environment and its Servlet technology [11]. The Java Servlet technology makes it possible to connect our programs to the web-server to enhance its functionality. When the user visits the manual pages the web-server automatically calls our Java function and establishes a channel through which we can send data directly to the end user in the HTML format. In the project the Apache HTTP Server [12] is used. To connect to our system, only its configuration file had been set.

The server side program module uses the XT [13] and XP [14] Java APIs by James Clark.

The XP API is an XML parser. It is able to read an XML document. It separates the tags from simple text. The XT API is also a parser, but it interprets XSL tags and XPath descriptions instead of XML tags. Both programs are implemented as Java classes.

In the following example two classes are created and the XSL parser is set up, that uses the specified XML parser inside. The *outputHandler* class deals with the output of the parsers, and transfers it to the web-server through the *ServerDestination*. The *xslPatterns* contains the XSL rules and the *XMLSourceFile* the XML-based documents.

```
XSLProcessorImpl xsl = new XSLProcessorImpl();
Object parserObj = Class.forName
    ("com.jclark.xml.sax.CommentDriver")
    .newInstance();
xsl.setParser((Parser) parserObj);

OutputMethodHandlerImpl outputHandler = new
    OutputMethodHandlerImpl(xsl);
xsl.setOutputMethodHandler(outputHandler);
Destination dest=new ServletDestination(res);
outputHandler.setDestination(dest);

xsl.loadStylesheet( xslPatterns);
xsl.parse( XMLSourceFile);
```

With the help of the XP parser we read through the documents and find the required element. The result of the search is still in XML form, which should be then transformed into HTML before being sent to the web browser.

Using XSLT we can describe transformation rules for XML documents. It makes it possible to pick up some pieces of the document, to change the order of elements, and to insert elements into the document. With the help of this technology we can convert our XML document into HTML. The following is an XSLT example, in which we transform the Title element into HTML format.

```
<xsl:template match="descendant::Title">
   <h1><center>
      <xsl:apply-templates/>
   </center></h1>
</xsl:template>
```

The syntax of XSLT follows the XML standard. This fact also shows the power of the XML language. The first line of this rule describes that we would like to match the `Title` element. The matching pattern is formulated in the XPath language. The next line specifies the output. In our example the `<h1><center>` are standard HTML tags. The `<xsl:apply-templates/>` marker tells the XSL processor to display the text surrounded by the `<Title>` tags in the

XML document. Finally, the `</center></h1>` tags come up at the end of the output. If the input was:

```
<Title>ACCURACY TEST REPORTS</Title>
```

The output will be:

```
<h1><center> ACCURACY TEST REPORTS</center></h1>
```

The XT parser can do this conversation automatically. We only have to describe the transformation rules for every content element. Using this technique we can achieve that our XML-based documentation can be processed at server side, and can be visualized in HTML browsers at client side.

6. THE USER INTERFACE

Figure 3 shows the result at client side running Internet Explorer. In the main window we can see the text of the product manual. The smaller window to the left shows general information and the selected component that will be highlighted in the main window. The lower window contains an input area for XPath expression, and a list of the most important content elements.

The basic aim of this designed was to demonstrate the power of XML-based product manuals. It makes it possible for programs to interpret the structure and meaning of the product documentation, and it can also help in processing complex content searching. We have demonstrated tools and techniques to build XML-based product manuals, and to generate document pages automatically based on user queries and the documentation.

Fig 3. Screenshot of the prototype XML system

Another advantage of the introduced system and the solution is that the XML technology remains hidden to the user. He only experiences very dynamic and useful web pages. According to the current trends in web system design, it is expected that in the near future web-browsers will implement the XML technology so we won't need to convert XML into HTML format.

The field of XML technology is developing rapidly. Some technology elements (for example XML linking) are still under development by the W3C. To implement and use these new technology elements into our system are the next developing steps.

7. SUMMARY

This paper summarized how we can use XML in electronic product manuals. It introduced briefly the XML language, its syntax, and authoring of the XML documents. We have followed the steps of authoring, and presented application examples. We have also demonstrated how to analyze and how to structure a simple text.

The paper described a prototype system, which was developed to display XML based documentation to product manual readers. We have shown the architecture of the system, its main components, and theirs tasks and the co-operation. In that way we can get acquainted with the tools belong to the eXtensible Markup Language. The main software tools used were an Apache web-server and the XT and XP APIs. We used Java Servlet technology to link these together.

We can conclude that a system based on the XML technology can provide essential tools to improve the efficiency of the processing of the text-based information. Such system, through the extensive usage of the XML related technologies, seems to be flexible and scalable to larger and even more diversified applications.

REFERENCES

[1] Deliverable 1.1: **Analysis of the Problem Domain**, Intelligent Product Manual, INCO-COP 96/0231, January 1998

[2] Deliverable 1.2: **Outline Solution that Illustrates the Capabilities of Intelligent Product Manuals**, Intelligent Product Manual, INCO-COP 96/0231, January 1998

[3] Excel Csepel Machine Tools Ltd. http://www.marathon-excel.com

[4] World Wide Web Consortium: Extensible Markup Language (XML) 1.0 http://www.w3c.org/TR/1998/REC-xml-19980210

[5] Chet Ensign : **$GML : The Billion Dollar Secret** (Charles F. Goldfarb Series on Open Information Management) Prentice Hall Computer Books, 1997

[6] Simon St. Laurent: **XML Elements of Style** McGraw-Hill, 2000

[7] Extensible Stylesheet Language (XSL) http://www.w3c.org/TR/WD-xsl

[8] XSL Transformation http://www.w3c.org/TR/xslt

[9] XMetal, Softquad Inc. http://www.softquad.com

[10] XML Path Language http://www.w3c.org/TR/xpath

[11] Ken Arnold, James Gosling, **The Java™ Programming Language Second Edition**, Addison Wesley Longman Computer & Engineering Publishing Group, ISBN 0-201-31006-6, 1998

[12] Apache Project http://www.apache.org

[13] XT API, James Clark http://www.jclark.com/xml/xt.html

[14] XP API, James Clark http://www.jclark.com/xml/xp/index.html