Multidimensional XML*

(ExtendedAbstract)

Yannis Stavrakas[1], Manolis Gergatsoulis[1], and Panos Rondogiannis[2]

[1] Institute of Informatics & Telecommunications,
National Centre for Scientific Research (N.C.S.R.) 'Demokritos',
153 10 Aghia Paraskevi Attikis, Greece.
{ystavr,manolis}@iit.demokritos.gr
[2] Department of Computer Science
University of Ioannina,
P.O. Box 1186, GR45110, Ioannina, Greece.
prondo@cs.uoi.gr

Abstract. The Extensible Markup Language (XML) tends to become a widely accepted formalism for the representation and exchange of data over the Web. A problem that often arises in practice is the representation in XML of data that are context-dependent (for example, information that exists in many different languages, in many degrees of detail, and so on). In this paper we propose an extension of XML, namely *MXML or Multidimensional XML*, which can be used in order to alleviate the problem of representing such context-dependent (or *multidimensional*) data. Apart from its usefulness in the semistructured data domain, MXML also reveals some new and promising research directions in the area of multidimensional languages.

1 Introduction

The Extensible Markup Language (XML) [1, 6, 19, 11] is a data description language which tends to become standard for representing and exchanging data over the Web. Although elegant and concise, the syntax of XML does not allow for convenient representation of multidimensional information. Suppose that one wants to represent in XML information that exists in different variations (for example in different languages, in various degrees of detail, or in different time points). With the current XML technology, a solution would be to create a different XML document for every possible variation. Such an approach however is certainly not practical, because it involves excessive duplication of information (especially if the number of variations is high and there exist large identical parts that remain unchanged between variations).

* This work has been partially supported by the Greek General Secretariat of Research and Technology under the project "Executable Intensional Languages and Intelligent Applications in Multimedia, Hypermedia and Virtual Reality" of ΠENEΔ'99, contract no 99EΔ265.

In this paper we propose a solution to the above problem, based on ideas that originate from the area of *multidimensional programming languages* [4, 14]. More specifically, we propose the language *Multidimensional XML (MXML)* which extends traditional XML with the capability of representing context dependent information in a compact way. The development of MXML was influenced by the ideas behind the design of *Intensional HTML* [20, 8, 7]. In contrast to IHTML, which aims at handling multidimensional information at a document level, the goal of MXML is to provide a formalism for representing and exchanging context-dependent data over the web. Apart from its applications in the XML domain, the study of MXML revealed new ideas that are of interest in the area of multidimensional languages.

2 Preliminaries

2.1 Extensible Markup Language

The basic component in XML is the *element*, which is a piece of data bounded by matching tags (markup) of the form `<element-name>` and `</element-name>`, called *start-tag* and *end-tag* respectively. Element names in XML are defined at will so that they best represent data domains. Inside an element we may have other elements, called *subelements*.

Markup encodes a description of the storage layout and the logical structure of the document. An XML document [6, 19] consists of nested element structures, starting with a root element. The data in the element are in the form of character data, subelements, or attributes.

XML allows us to associate *attributes* with elements. Attributes in XML are declared within element start tags and have the form of a sequence of name-value pairs. The value of an attribute is always a string enclosed in quotation marks. Unlike subelements, where a subelement with the same name can be repeated inside an element, an attribute name may only occur once within a given element.

Example 1 shows a piece of XML document that contains information about a book.

Example 1. A sample XML document.

```
<bibliography>
     <book isbn="12345678" publisher = "pb1">
           <author>
                 <firstname> Manolis </firstname>
                 <lastname> Gergatsoulis </lastname>
           </author>
           <author>
                 <firstname> Panos </firstname>
                 <lastname> Rondogiannis </lastname>
           </author>
```

```
            <title> Multidimensional Programming Languages </title>
            <price currency = "USD"> 100 </price>
            <year> 2000 </year>
      </book>
      ...other book elements ...

      <publisher id = "pb1">
            NCSR Demokritos
      </publisher>
      ...other publisher elements ...

</bibliography>
```

In this example, the element `book` has the attributes `isbn` and `publisher`. It also has five subelements. The first two represent the authors of the book, while the rest represent the title, the price and the publication year of the book. The attribute `publisher` of the `book` element refers to the `publisher` element.

A key feature of XML is the capability to specify the structure of XML documents through the use of *Document Type Definitions* (DTDs). A DTD declares constraints on elements and attributes and can be viewed as a context free grammar for XML documents, or as a kind of schema for XML data. A document may refer to a DTD to which it conforms.

2.2 Multidimensional Formalisms

The idea of "dimension enabled" languages is not new. Possibly the first multidimensional programming language is the (functional) language GLU [3, 4]. GLU allows the user to declare dimensions, and to define multidimensional entities that vary across these dimensions. So, a two-dimensional entity can be thought as an infinite table, a three-dimensional one as a cube extending infinitely across the three dimensions, and so on. One can perform various operations with arguments such higher-dimensional entities, or even define functions that take them as parameters and return new entities as results. Moreover, the language supports intensional operators that work along each different dimension. A language in the spirit of GLU has also been developed in the logic programming domain [14].

An area in which multidimensionality appears to offer significant benefits is the area of *version control* [16]. The intensional versioning approach described in [16] has recently found applications in the evolving area of Internet computing. One example application in this domain is the development of the language IHTML (Intensional HTML) [20, 8, 7, 18], a high-level Web authoring language. The main advantage of IHTML over HTML is that it allows practical specification of Web pages that can exist in many different variations. Web sites created by IHTML are easier to maintain and require significantly less space when compared to the sites created by cloning conventional HTML files.

Finally, we should mention the language ISE [17] which is a multidimensional version of Perl. ISE is more general purpose than IHTML and it is expected to have a broader range of applications.

3 Multidimensional XML

When modelling the real world, the same entity may often have multiple facets. For example, a technical document concerning a car may vary according to the language, the metric system (i.e. miles, or kilometres), the price currency, etc. of the potential customer. In classical XML one has to write different XML documents, each one representing a possible variation. In multidimensional XML we are allowed to specify elements that may exhibit varying content and structure, by assigning them dimensions. This is done by extending the syntax of XML.

3.1 Syntax of Multidimensional XML Documents

A Multidimensional XML document (MXML document) is presented in example 2.

Example 2. A sample multidimensional XML document:

```
<bibliography>
     <book isbn="12345678" publisher = "pb1">
          <author>
                   [language = English]
                         <firstname> Manolis </firstname>
                         <lastname> Gergatsoulis </lastname> [/]
                   [language = Greek]
                         <firstname> Μανόλης </firstname>
                         <lastname> Γεργατσούλης</lastname> [/]
          </author>
          <author>
                   [language = English]
                         <firstname> Panos </firstname>
                         <lastname> Rondogiannis </lastname> [/]
                   [language = Greek]
                         <firstname>Πάνος</firstname>
                         <lastname> Ροντογιάννης </lastname> [/]
          </author>
          <title>
               [language = English]
                   Multidimensional Programming Languages [/]
               [language = Greek]
                   Πολυδιάστατες Γλώσσες Προγραμματισμού[/]
          </title>
          <price currency= "USD">
```

```
            [period = discount client = regular]
                100 [/]
            [period = normal client = regular]
                120 [/]
            [period in {discount, normal} client = special]
                100 [/]
        </price>
        <year> 1999 </year>
   </book>
   ...other book elements ...

   <publisher id = "pb1">
        [language = English] NCSR Demokritos [/]
        [language = Greek] ΕΚΕΦΕ Δημόκριτος [/]
   </publisher>
   ...other publisher elements ...

</bibliography>
```

The document in example 2 is a multidimensional extension of the document in example 1. The elements `author` and the elements `title` and `publisher` have two versions each. One corresponding to the value `English` of the dimension `language` and the other corresponding to the value `Greek` of the same dimension.

The element `price` depends on two dimensions namely `period` and `client`. Notice that in general an element or an attribute may depend on zero, one or more dimensions. The dimensions are considered orthogonal in the sense that the value of one dimension does not depend on the values of another dimension. Each dimension may be assigned to more than one elements or attributes.

By the term *context* we refer to a set of dimension-value pairs assigned to a given element or attribute.

The syntax of XML is extended as follows in order to incorporate the use of dimensions. In particular, an element in MXML has the form:

```
<element_name attribute_specification>
      [context_specifier_1]
            element_contents_1
      [/]
            .
            .
            .
      [context_specifier_n]
            element_contents_n
      [/]
</element_name>
```

where `element_contents_i`, with $1 \leq \mathtt{i} \leq n$ is the contents of the element for the context specified by `context_specifier_i`. Note that all the alternative `element_contents_i` above, are within the same element "`element_name`". In addition, all contents of an element must occur inside a context specifier; in other words context specifiers cannot occur freely inside element contents, instead they are "attached" to element names. A *context specifier* is of the form:

```
dimension_1_specifier, ..., dimension_m_specifier
```

where `dimension_i_specifier`, for $\mathtt{i} = 1$ to `m` is a *dimension specifier* of the form:

```
dimension_name  specifier_operator  dimension_value_expression
```

A *specifier_operator* is one of $=$, $!=$, `in`, `not in`. If the *specifier_operator* is either $=$ or $!=$ then the *dimension_value_expression* consists of a single dimension value. Otherwise, if the *specifier_operator* is one of `in`, `not in` then the *dimension value expression* is a set of the form $\{\mathtt{value_1}, \ldots, \mathtt{value_k}\}$, with $k \geq 1$.

For example the following expressions are valid context specifiers:

```
[language in {Greek, English, French}]
[language = Greek]
[language != French]
```

Note that we assume that the dimension-value expressions that correspond to a specific element must be formulated in such a way so as no two of them are concurrently satisfied.

An attribute may also depend on dimensions, in which case, the syntax is similar to that of elements. Thus, the `attribute_specification` is of the form:

```
attribute_name =
      [context_specifier_1]
            attribute_value_1
      [/]
            .
            .
      [context_specifier_n]
            attribute_value_n
      [/]
```

4 On the Semantics of MXML

In this section, we discuss in brief some points related to the semantics of MXML.

4.1 Scoping of Dimensions

A dimension assigned a value in an element retain its value in all the descendant elements of this element. However, if a dimension which has been assigned a value in an element E, is assigned a new value in a descendant E1 of E, then the active value of that dimension for E1 and its descendants is the value of the dimension assigned in element E1.

Dimensions for attributes are independent of the dimensions of elements. Moreover, the values of the dimensions of an attribute apply only to the specific attribute.

4.2 Multidimensional OEM

Object Exchange Model (OEM) is a simple graph-based data model for semistructured data designed at Stanford as a part of the Tsimmis project [9]. Due to the semistructured nature of XML, it has also been adopted for modelling XML documents. In OEM, an XML document is represented as a graph with a single root. The nodes represent the element contents while the edges represent the element names, with leaves holding character data. Attribute values are given on corresponding nodes, and attribute references are depicted as dashed lines [2].

In order for OEM to model MXML documents we have to adapt it so as to represent elements and attributes whose contents vary according to dimension values. For this, we extend OEM by introducing another type of node-edge pair, which is represented by a thick line departing from a square. The square is called *context node* while the thick line is called *context edge*. The MOEM model of the MXML document of example 2 is shown in figure 1.

For each multidimensional element (whose value depends on a number of contexts), there corresponds a context node and a set of context edges departing from this node. Each context edge represents a possible context and leads to the content of the element for the specific context.

Attributes whose values depend on contexts are represented in a similar way. Attribute values qualified by the dimension values are given on corresponding nodes. In the case of attribute references, the dashed line of the OEM model leads to a context node from which context attribute edges depart (denoted by thick lines). Each context edge leads to the element node to which the attribute refers for the specific context.

It is easy to see that, from a given MXML document we can obtain a set of ordinary XML documents, each one corresponding to different combination of choices of context edges.

5 Discussion and Future Work

The MXML formalism presented in this paper attempts to remedy what seems to be a shortcomming of XML, namely its inability to represent context-dependent information in a concise way. Although MXML has inherited many features from

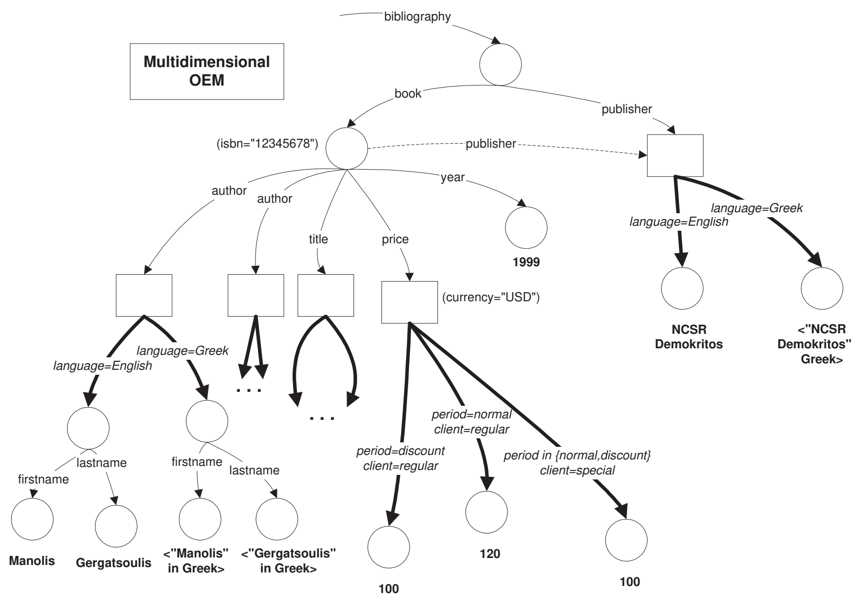

Fig. 1. The Multidimensional OEM model for example 2.

existing multidimensional formalisms, it also appears to lead to new interesting and unexplored issues of multidimensionality. In the following we discuss the relationship of MXML with other formalisms and also outline new directions for future research.

Relationships with Existing Formalisms: MXML has been mainly influenced from the work on IHTML [20, 8, 7] (Intensional HTML). In fact, the work on MXML started as an attempt to transfer the research results of the IHTML project, to the more data-oriented formalism of XML. The main difference between IHTML and MXML is a projection of the difference between HTML and XML. IHTML is focused on the composition and presentation of hyperdocuments while MXML is focused on data representation mainly for information exchange purposes. In this context MXML has to deal with problems such as assigning dimensions to the arbitrary structure of XML data. Other XML-ralated issues such as DTDs and XML query languages give an interesting research direction to XML.

Future Research Problems: There are many aspects of the MXML formalism that the authors would like to further investigate. In the following we list some of them:

– In the current discussion we have made the assumption that a MXML document exists without reference to an existing DTD (Document Type Definition). DTD are formalisms similar to *types* in programming languages and their purpose is to impose restrictions on what a particular document can

contain. For example, a DTD can specify that a document describing a book can only contain a single `<title>` element. In this paper we have not considered DTDs with respect to MXML documents. It is quite possible that MXML will require the definition of MDTDs (Multidimensional DTDs).
- We have not considered any particular query language for MXML. Research in query languages for XML is especially active [12, 13, 5]. We believe that a query language for MXML would have to take *contexts* into account.
- A potential application of MXML which we are currently investigating concerns the representation of time-dependent information. Much work has been carried out in the past on temporal databases [15]. The need to incorporate time information is also present in the case of semistructured data [10]. However to the best of our knowledge, no such extension has been considered for XML.
- There is currently no existing implementation of the ideas presented in this paper. The authors plan to undertake such an implementation of MXML together with an associated query language.

We believe that further research in the relationships between semistructured data and multidimensionality, will reveal many issues from which both worlds will gain significant benefits.

References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, 2000.
2. Serge Abiteboul. On views and XML. In *Proceedings of the 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Data Base Systems*, pages 1–9. ACM Press, 1999.
3. E. A. Ashcroft, A. A. Faustini, and R. Jagannathan. An intensional language for parallel applications programming. In B.K.Szymanski, editor, *Parallel Functional Languages and Compilers*, pages 11–49. ACM Press, 1991.
4. E. A. Ashcroft, A. A. Faustini, R. Jagannathan, and W. W. Wadge. *Multidimensional programming*. Oxford University Press, 1995.
5. A. Bonifati and S. Ceri. Comparative analysis of five XML query languages. *SIGMOND Record*, 29(1), March 2000.
6. T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible markup language (XML) 1.0. http://www.w3.org/TR/REC-xml, 1998.
7. G. D. Brown. IHTML 2: Design and Implementation. In W. W. Wadge, editor, *Proceedings of the 11th International Symposium on Languages for Intensional Programming*, 1998.
8. G. D. Brown. Intensional HTML 2: A practical Approach. Master's thesis, Department of Computer Science, University of Victoria, 1998.
9. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of IPSJ Conference, Tokyo, Japan, October*, pages 7–18, 1994.
10. S. S. Chawathe, S. Abiteboul, and J. Widom. Representing and quering changes in semistructured data. *Theory and Practice of Object Systems (TAPOS)*, 2000. Special Issue on Object-Oriented Technology in Advanced Applications (to appear).

11. Sudarshan S. Chawathe. Describing and manipulating XML data. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 22(3):3–9, September 1999.
12. A. Deutch, M. Fernández, D. Florescu, A. Levy, and D. Suciu. XML-QL: A query language for XML. http://www.w3.org/TR/NOTE-xml-ql, 1999.
13. A. Deutsch, M. Fernández, D. Florescu, A. Levy, D. Maier, and D. Suciu. Quering XML data. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 22(3):10–18, September 1999.
14. M. A. Orgun and W. Du. Multi-dimensional logic programming: Theoretical foundations. *Theoretical Computer Science*, 158(2):319–345, 1997.
15. G. Ozsoyoglu and R. T. Snodgrass. Temporal and real-time databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):513–532, August 1995.
16. J. Plaice and W. W. Wadge. A New Approach to Version Control. *IEEE Transactions on Software Engineering*, 19(3):268–276, 1993.
17. P. Swoboda and W. W. Wadge. Vmake and Ise General Tools for the Intensionalization of Software Systems. In M. Gergatsoulis and P. Rondogiannis, editors, *Intensional Programming II*, pages 310–320. World Scientific, 2000.
18. W.W. Wadge, G. D. Brown, m.c. schraefel, and T. Yildirim. Intensional HTML. In *Proceedings of the Fourth International Workshop on Principles of Digital Document Processing (PODDP '98)*, Lecture Notes in Computer Science (LNCS) 1481, pages 128–139. Springer-Verlag, March 1998.
19. Norman Walsb. A guide to XML. *World Wide Web Journal "XML:Principles, Tools and Techniques"*, 2(4):97–107, 1997.
20. T. Yildirim. Intensional HTML. Master's thesis, Department of Computer Science, University of Victoria, 1997.