Representing Changes in XML Documents using Dimensions

Manolis Gergatsoulis[1] and Yannis Stavrakas[2]

[1] Department of Archive and Library Sciences, Ionian University,
Palea Anaktora, Plateia Eleftherias, 49100 Corfu, Greece.
manolis@ionio.gr
http://www.ionio.gr/~manolis/
[2] Knowledge & Database Systems Laboratory
Dept. of Electrical and Computing Engineering
National Technical University of Athens (NTUA), 15773 Athens, Greece.
ys@dblab.ntua.gr

Abstract. In this paper, we present a method for representing the history of XML documents using Multidimensional XML (MXML). We demonstrate how a set of basic change operations on XML documents can be represented in MXML, and show that temporal XML snapshots can be obtained from MXML representations of XML histories. We also argue that our model is capable to represent changes not only in an XML document but to the corresponding XML Schema document as well.

1 Introduction

The management of multiple versions of XML documents and semistructured data is an important problem for many applications and has recently attracted a lot of research interest [3, 13, 4, 5, 16, 17]. One of the most recent approaches appearing in the literature [16, 17] proposes the use of Multidimensional OEM (MOEM), a graph model designed for *multidimensional semistructured data (MSSD)*[15] as a formalism for representing the history of time-evolving semistructured data (SSD). MSSD are semistructured data that present different facets under different *contexts*. A context represents alternative *worlds*, and is expressed by assigning values to a set of user-defined variables called *dimensions*. The basic idea behind the approach proposed in [16, 17] is to use MOEM with a time dimension whose values represent the time points under which an OEM object holds. In order to use MOEM to represent changes in OEM databases, a set of basic change operations for MOEM graphs as well as a mapping from changes in an OEM database to MOEM basic change operations has been defined. An interesting property of MOEM graphs constructed in this way is that they can give temporal snapshots of the OEM database for any time instance, by applying a process called *reduction*. Queries on the history of the changes can also be posed using MQL [18], a query language for MOEM.

Following the main ideas presented in [16, 17], in this paper we address the problem of representing and querying histories of XML documents. We propose the use of *Multidimensional XML* (MXML) [7,8], an extension of XML which shares the same ideas as MSSD, in order to represent context-dependent information in XML. MXML is used as a formalism to represent and manipulate the histories of XML documents. The syntax particularities of XML require to adapt the MOEM approach described in [16,17] so that they are taken into account. The main contributions of the present work can be summarized as follows:

1. We consider four basic change operations on XML documents and show how the effect of these operations on (the elements and attributes of) XML documents, can be represented in Multidimensional MXML. We also show how our representation formalism can take into account attributes of type ID/IDREF(S) in the representation of the history of the XML document.
2. We demonstrate how we can obtain temporal snapshots that correspond to versions holding at a specific time, by applying a process called *reduction* to the MXML representation of the document's history.
3. We argue that our approach is powerful enough to represent not only the history of an XML document but also the history of the document's schema, expressed in XML Schema, which may also change over time. The temporal snapshots of the schema are also obtained by applying the reduction process.

2 Related Work

a) Representing and querying changes in semistructured data: The problem of representing and querying changes in semistructured data has also been studied in [3], where *Delta OEM* (DOEM in short) was proposed. DOEM is a graph model that extends OEM with annotations containing temporal information. Four basic change operations, namely $creNode$, $updNode$, $addArc$, and $remArc$ are considered by the authors in order to modify an OEM graph. Those operations are mapped to four types of annotations, which are tags attached to a node or an edge, containing information that encodes the history of changes for that node or edge. To query DOEM databases, the query language *Chorel* is proposed. Chorel extends *Lorel* [1] with constructs called *annotation expressions*, which are placed in path expressions and are matched against annotations in the DOEM graph.

A special graph for modeling the dynamic aspects of semistructured data, called *semistructured temporal graph* is proposed in [13]. In this graph, every node and edge has a label that includes a part stating the valid interval for the node or edge. Modifications in the graph cause changes in the temporal part of labels of affected nodes and edges.

b) Approaches to represent time in XML: An approach for representing temporal XML documents is proposed in [2], where leaf data nodes can have alternative values, each holding under a time period. However, the model presented in [2], does not explicitly support facets with varying structure for non-leaf nodes. Other approaches to represent valid time in XML include [9,10]. In [6] the XPath data model is extended to support transaction time. The query language

of XPath is extended as well with transaction time axis to enable to access past and future states. Constructs that extract and compare times are also proposed. Finally, in [14] the XPath data model and query language is extended to include valid time, and XPath is extended with an axis to access valid time of nodes.
c) Schemes for multiversion XML documents: The problem of managing (storing, retrieving and querying) multiple versions of XML documents is examined in [4,5]. Most recently [20,19], the same authors proposed an approach of representing XML document versions by adding two extra attributes, namely `vstart` and `vend`, representing the time interval for which this elements version is valid. The authors also demonstrate how XQuery can be used to express queries in their representation scheme. The representation employed in [20,19] presents a lot of similarities with our approach which, however, is more general, and overcomes some limitations of the approach in [20,19].

3 Multidimensional XML

In a *multidimensional XML document* (*MXML document* in short), dimensions may be applied to elements and attributes. A *multidimensional element/attribute* is an element/attribute whose content depends on one or more dimensions.

3.1 Incorporating Dimensions in XML

The notion of *world* is fundamental in MXML. A world represents an environment under which data in a multidimensional document obtain a meaning. A world is determined by assigning values to a set of *dimensions*.

Definition 1. *Let $\mathcal{S}$ be a set of dimension names and for each $d \in \mathcal{S}$, let $\mathcal{D}_d$, with $\mathcal{D}_d \neq \emptyset$, be the domain of d. A* world *W is a set of pairs (d, u), where $d \in \mathcal{S}$ and $u \in \mathcal{D}_d$ such that for every dimension name in $\mathcal{S}$ there is exactly one element in W.*

MXML uses *context specifiers*, which are expressions that specify sets of worlds. Context specifiers qualify the variants (or *facets*) of multidimensional elements and attributes, called *context elements/attributes*, stating the sets of worlds under which each variant may hold. The context specifiers that qualify the facets of the same multidimensional element/attribute are considered to be *mutually exclusive* in the sense that they specify disjoint sets of worlds. An immediate consequence of this requirement is that every multidimensional element/attribute has at most one holding facet under each world. A multidimensional element is denoted by preceding the element's name with the special symbol "`@`", and encloses one or more *context elements*. Context elements have the same form as conventional XML elements. All context elements of a multidimensional element have the same name which is the name of the multidimensional element.

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

```
<@element_name attribute_specification>
  [context_specifier_1]
    <element_name attribute_specification_1>
      element_content_1
    </element_name>
  [/]
  . . .
  [context_specifier_N]
    <element_name attribute_specification_N>
      element_content_N
    </element_name>
  [/]
</@element_name>
```

To declare a multidimensional attribute we use the following syntax:

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

For more details on MXML the reader may refer to [7].

As an example of MXML consider information about a book which exists in two different editions, an English and a Greek one. In Example 1, the element `book` has six subelements. The `isbn` and `publisher` are multidimensional elements and depend on the dimension `edition`. The elements `title` and `authors` remain the same under every possible world. The element `price` is a conventional element containing however a multidimensional attribute (the attribute `currency`) as well as the two multidimensional elements `value` and `discount`. Now two more dimensions appear, namely the dimensions `time` and `customer_type`. Notice that the value of the attribute `currency` depends on the dimensions `edition` and `time` (as to buy the English edition we have to pay in USD, while to buy the Greek edition we should pay in GRD before `2002-01-01` and in EURO after that date due to the change of the currency in EU countries). The element `value` depends on the same dimensions as the attribute `currency`, while the element `discount` depends on the dimensions `edition` and `customer_type`, as students are offered higher discount than libraries.

Example 1. Multidimensional Information about a book encoded in MXML.

```
<book>
  <@isbn>
    [edition = greek] <isbn>0-13-110370-9</isbn> [/]
    [edition = english] <isbn>0-13-110362-8</isbn> [/]
  </@isbn>
  <title>The C programming language</title>
  <authors>
    <author>Brian W. Kernighan</author>
    <author>Dennis M. Ritchie</author>
  </authors>
  <@publisher>
    [edition = english] <publisher>Prentice Hall</publisher> [/]
    [edition = greek] <publisher>Klidarithmos</publisher> [/]
  </@publisher>
  <@translator>
    [edition = greek] <translator>Thomas Moraitis</translator> [/]
  </@translator>
  <price currency = [edition = greek, time in {start .. 2001-12-31}]GRD[/]
                   [edition = greek, time in {2002-01-01 .. now}]EURO[/]
                   [edition = english]USD[/]>
    <@value>
      [edition=greek,time in {start .. 2001-12-31}]<value>13.000</value>[/]
      [edition=greek,time in {2002-01-01 .. now}]<value>40</value>[/]
      [edition=english]<value>80</value>[/]
    </@value>
    <@discount>
      [edition = greek,customer_type = student]<discount>20</discount>[/]
      [edition = greek,customer_type = library]<discount>10</discount>[/]
    </@discount>
  </price>
</book>
```

3.2 Encoding MXML in XML

The introduction of contexts in XML documents can also be achieved by using standard XML syntax instead of the syntax proposed in the previous subsection. In fact a number of papers have appeared [2, 9, 10] in which time information (that we express here through a time dimension) is encoded either through additional attributes or by using elements with special meaning. Using similar ideas we can encode our MXML documents using the constructs offered by standard XML syntax. For example, our multidimensional elements could be encoded in standard XML by employing a construct of the form:

```
<mxml:group name = "p">
    <mxml:celem>
        <mxml:context> ... </mxml:context>
        <mxml:elem> <p> ... </p> </mxml:elem>
    </mxml:celem>
        ...
    <mxml:celem>
        <mxml:context> ... </mxml:context>
        <mxml:elem> <p> ... </p> </mxml:elem>
    </mxml:celem>
</mxml:group>
```

where a multidimensional element whose name is `p` is denoted by the special MXML element name `mxml:group`, and each element `mxml:celem` corresponds to an element facet together with its context specifier, belonging to that multidimensional element. In a similar way we could encode appropriately in standard XML the content of the element `mxml:context` (i.e. the context specifiers of MXML). We, however, keep using in the rest of the paper, the syntax that we have proposed for MXML, as it offers a clear distinction between context information and the corresponding facets of elements/attributes, resulting in documents that are more readable by humans. Moreover, MXML documents are shorter in size than their corresponding XML representation. We should, however, note that one could use the syntax that we propose for MXML and then transform it automatically into standard XML through a preprocessing phase.

3.3 Obtaining XML Instances from MXML

An important issue concerning the context specifiers of a multidimensional element/attribute is that they must be mutually exclusive, in other words, they must specify disjoint sets of worlds. This property makes it possible, given a specific world, to safely reduce an MXML document to an XML document holding under that world. Informally, the reduction of an MXML document D to an XML document D_w holding under the world w proceeds as follows:

Beginning from the document root to the leaf elements, each multidimensional element E is replaced by its context element E_w, which is the holding facet of E under the world w. If there is no such context element, then E along with its subelements is removed entirely. A multidimensional attribute A is transformed into a conventional attribute A_w whose name is the same as A and whose value is the holding one under w. If no such value exists then the attribute is removed entirely.

Example 2. For the world `w={(edition,greek), (customer_type,student), (time,2002-03-03)}`, the MXML document in Example 1 is reduced to the conventional XML document that follows:

```
<book>
  <isbn>0-13-110370-9</isbn>
  <title>The C programming language</title>
  <authors>
     <author>Brian W. Kernighan</author>
     <author>Dennis M. Ritchie</author>
  </authors>
  <publisher>Klidarithmos</publisher>
  <translator>Thomas Moraitis</translator>
  <price currency = EURO>
     <value>40</value> <discount>20</discount>
  </price>
</book>
```

The above process can be generalized for producing an MXML document for a set S of worlds. In this generalization, called *partial reduction*, the MXML document produced is a subdocument of the original MXML document, which encompasses all XML documents that hold under some world in S.

Example 3. For the worlds expressed by `w={(edition,greek),(customer_type, student)}`, the MXML document in Example 1 is partially reduced to an MXML document similar to the document of Example 2 except for the element `price` whose partially reduced version is given below:

```
<price currency = [time in {start .. 2001-12-31}]GRD[/]
                  [time in {2002-01-01 .. now}]EURO[/]>
   <@value>
      [time in {start .. 2001-12-31}]<value>13.000</value>[/]
      [time in {2002-01-01 .. now}]<value>40</value>[/]
   </@value>
   <discount>20</discount>
</price>
```

4 Representing Histories of XML Documents in MXML

In order to represent the changes in an XML document we encode this document as an MXML document in which a dimension named `d` is used to represent time. More specifically, instead of retaining multiple instances of the XML document, we retain a single MXML representation of all successive versions of the document. We assume that the time domain T of `d` is linear and discrete. As seen in Example 1, we also assume a) a reserved value *start*, such that $start < t$ for every $t \in T$, representing the beginning of time, and b) a reserved value *now*, such that $t < now$ for every $t \in T$, representing the current time.

The time period during which a context element/attribute is the holding facet of the corresponding element/attribute is denoted by qualifying that context element/attribute with a context specifier of the form `[d in` $\{t_1 \,..\, t_2\}$`]`. In context specifiers the syntactic shorthand $v_1..v_n$ for discrete and totally ordered domains means all values v_i such that $v_1 \leq v_i \leq v_n$.

In the following three subsections we consider the representation of the histories of XML documents without attributes of type IDREF(S). The case of XML documents in which attributes of type IDREF(S) do appear is discussed in subsection 4.4.

4.1 Basic Change Operations on Elements

We consider three primitive change operations, namely *update*, *delete*, and *insert*, on XML documents and demonstrate how their effect on XML elements can be represented in MXML:
a) **Update:** Updating the value of an XML element can be seen as the replacement of the element with another element which has the same name but different content. The way that we represent the effect of update in the MXML representation of the history of the XML document is depicted in Figure 1. The value of the element `r` in the XML document on the left part of the table is updated at time `t1` from `v2` to the new value `v4`. The effect of this operation is shown on the right side of the table. The element `r` has now become a multidimensional element with two facets. The first facet is valid in all time points of the interval `start..t1-1`, while the other is valid in all time points of the interval `t1..now`.

```
<p>
   <q> v1 </q>
   <r> v2 </r>
   <s> v3 </s>
</p>
```

```
<p>
   <q> v1 </q>
   <@r>
      [d in {start..t1-1}] <r> v2 </r> [/]
      [d in {t1..now}] <r> v4 </r> [/]
   </@r>
   <s> v3 </s>
</p>
```

Fig. 1. Applying the operation **update** on the element `r` at time `t1`.

Note that a subsequent update of the same element `r` at a time point `t2` will be represented in the MXML document as follows: a) by simply adding a new facet in the multidimensional element `@r` holding in the interval `t2..now` and b) by changing the value of the dimension `d` of the most recent facet of `@r` from `t1..now` to `t1..t2-1`. The same process is also used to update complex elements.

b) <u>**Delete:**</u> The deletion of the element `r` at time `t1`, is represented in MXML by a multidimensional element with a single facet holding during the interval `start..t1-1`, as shown in Figure 2. In case, however, in which the element that we want to delete is already multidimensional, the deletion is representing by simply changing the end time point of the most recent facet of the element (the facet for which the end point of the value of `d` is `now`) from `now` to `t1-1`. The operation *delete* applies equally to both simple and complex elements. In the later case, the entire content of the element is considered as deleted without any need to transform the elements it contains to multidimensional elements. This is a consequence of the fact that the contexts of parent elements in MXML are inherited and combined with possible contexts of child elements [15].

```
<p>
   <q> v1 </q>
   <r> v2 </r>
   <s> v3 </s>
</p>
```

```
<p>
   <q> v1 </q>
   <@r>
      [d in {start..t1-1}] <r> v2 </r> [/]
   </@r>
   <s> v3 </s>
</p>
```

Fig. 2. Applying the operation **delete** on the element `r` at time `t1`.

c) <u>**Insert:**</u> As depicted in Figure 3, the new element `r` inserted at the time point `t1`, is added as a multidimensional element with a single context element holding during the interval `t1..now`, in the MXML representation of the document.

Notice that the inserted element may be a complex one (as shown in Figure 3). In this case, all the descendants of the inserted element inherit the same context.

```
<p>
   <q> v1 </q>
   <r> v2 </r>
</p>
```

```
<p>
   <q> v1 </q>
   <@r>
      [d in {t1..now}]
         <r> <u> v3 </u> <w> v4 </w> </r>
      [/]
   </@r>
   <r> v2 </r>
</p>
```

Fig. 3. Insert the element `<r> <u> v3 </u> <w> v4 </w> </r>` after the element `q` at time `t1`.

Notice also there may exist other subelements with the same element name (`r` in our example) without any confusion.

4.2 Basic Change Operations on Attributes

The basic change operations on attributes are similar to those on elements:
a) **Update**. Consider the element:

```
<p a1 = "9"> v1 </p>
```

and suppose that we update the value of the attribute `a1` to the new value `8` at time point `t1`. Then we obtain the following MXML element:

```
<p a1 = [d in {start..t1-1}]"9"[/][d in {t1..now}]"8"[/]> v1 </p>
```

b) **Delete**. Consider the element:

```
<p a1 = "9"> v1 </p>
```

and suppose that we want to delete the attribute `a1` of `p` at time point `t1`. Then we get the following element in MXML:

```
<p a1 = [d in {start..t1-1}]"9"[/]> v1 </p>
```

c) **Insert**. Consider the element:

```
<p a1 = "9"> v1 </p>
```

and suppose that we want to add at time point `t1` a new attribute whose name is `a2` and whose value is `3`. Then we get the following element in MXML:

```
<p a1 = "9" a2 = [d in {t1..now}]"3"[/]> v1 </p>
```

4.3 A more Complex Example

Consider six successive versions (a)-(f) of an XML document shown in Figure 4. Here, each version is obtained from the previous version by applying a basic change operation as follows: (b) is obtained from (a) by inserting the subelement `<q> v2 </q>` at time `t1`. Version (c) is obtained from (b) by inserting, at time `t2`, the attribute `a = "1"` to the first occurrence of the element `q`. Version (d) is obtained from (c) by updating the value of the second occurrence of `q` at time `t3`. Version (e) is obtained by deleting `<s> v4 </s>` at time `t4`. Finally, version (f) is obtained from (e) by updating the value of the element `u` at time `t5`.

One can easily verify that all versions of the XML document of Figure 4 are represented by the MXML document in Figure 5.

```
<p>
  <q> v1 </q>
</p>
```
(*a*)

```
<p>
  <q> v1 </q>
  <q> v2 </q>
</p>
```
(*b*)

```
<p>
  <q a = "1"> v1 </q>
  <q> v2 </q>
</p>
```
(*c*)

```
<p>
  <q a = "1"> v1 </q>
  <q>
    <u> v3 </u>
    <s> v4 </s>
  </q>
</p>
```
(*d*)

```
<p>
  <q a = "1"> v1 </q>
  <q>
    <u> v3 </u>
  </q>
</p>
```
(*e*)

```
<p>
  <q a = "1"> v1 </q>
  <q>
    <u> v5 </u>
  </q>
</p>
```
(*f*)

Fig. 4. Six successive versions of an XML document.

```
<p>
  <q a = [d in {t2..now}] "1" [/]> v1 </q>
  <@q>
    [d in {t1..t3-1}] <q> v2 </q> [/]
    [d in {t3..now}]
      <q>
        <@u>
          [d in {start..t5-1}] <u> v3 </u> [/]
          [d in {t5..now}] <u> v5 </u> [/]
        </@u>
        <@s> [d in {start..t4-1}] <s> v4 </s> [/] </@s>
      </q>
    [/]
  </@q>
</p>
```

Fig. 5. MXML document representing the XML document versions of Figure 4.

4.4 Representing the Basic Change Operations in the Presence of Attributes of Type IDREF(S)

So far we have assumed that attributes of type IDREF(S) do not appear in the XML document. However, attributes of this type are often used in XML documents as they offer a means to avoid unnecessary repetition of information. In this subsection, we adapt our method of representing changes of XML documents so as to take into account the presence of attributes of type IDREF(S). This adaptation is based on the following rules:

- When a conventional element E, which has an attribute A of type ID, is transformed into a multidimensional element (as a result of a basic change operation), then A becomes an attribute of the multidimensional element.

– The basic change operations are not allowed to modify the values of attributes of type ID (they are, however, allowed to modify attributes of type IDREF(S)).
– Context elements do not have attributes of type ID. All the facets of a multidimensional element share the same ID attribute of that multidimensional element (if there is such an attribute).

The manipulation of an ID attribute during an update operation is shown in Figure 6. The element `r`, whose content is updated at time `t1`, has an attribute named `id` of type ID for which there is a reference from the element `q` (through its attribute `ref` of type IDREF). In order to represent the effect of this update, a multidimensional element `@r` is created and the two `r` elements (the old one and the new one) become facets of that multidimensional element. Notice however that the attribute `id` of the old element `r` has been removed from the facet `r` and becomes now an attribute of the multidimensional element `@r`. When we apply the reduction process to get the temporal snapshot of the document holding at a time point `t`, this attribute is copied to the element facet holding at `t`.

```
<p>                        | <p>
  <q ref="id1"> ... </q>   |   <q ref="id1"> ... </q>
    ...                    |     ...
    <r id="id1"> v1 </r>   |     <@r id="id1">
    ...                    |       [d in {start..t1-1}] <r> v1 </r> [/]
</p>                       |       [d in {t1..now}] <r> v2 </r> [/]
                           |     </@r>
                           |     ...
                           | </p>
```

Fig. 6. Updating an element which has an attribute of type ID.

The deletion of an element which is referred from another element through an attribute of type IDREF, is problematic since it results in a reference not showing to an existing element at every time point. However, this is a problem related to the deletion operation itself and not to its representation in MXML.

5 Querying the History of an XML Document

Supporting effective query answering on the history of an XML document is vital in the evaluation of a versioning model. In [4], a taxonomy of the most important queries on versions is given. This taxonomy consists of four categories, namely: 1) *Temporal selection*, which refer to the retrieval of either a particular version or a sequence of successive versions of the document. 2) *Document evolution & historical queries*, which focus on querying the changes between successive versions. 3)*Structural projection*, in which the user requests certain parts (elements) of the document. 4) *Content-based selection*, which refers in retrieving all versions (of the whole document or part of it) that fulfill some conditions.

Temporal selection queries are answered in our approach through the processes of reduction. Reduction to XML is used to retrieve a particular version for a given time instance. One can easily verify that each version of the XML document in Figure 4 can be obtained by applying the reduction to XML on the MXML in Figure 5. Partial reduction is used to retrieve an MXML subdocument encompassing a set of XML versions that correspond to a set of time instances.

We are currently working in incorporating context in XPath and XQuery, in order to come up with a context-aware query language for MXML. Previous work on a context-aware query language for MOEM led to *Multidimensional Query Language* (*MQL*) [18], which extends Lorel [1] and incorporates context specifiers. In [17], we demonstrate how MQL can be used to formulate queries on the history of OEM databases that cover all the above categories. Our current work follows a similar direction, and aims to extend XPath with context qualifiers that express conditions on the contexts of the corresponding path elements. Moreover, we plan to extend XQuery with additional clauses that use *context variables* to pose complex conditions on contexts.

6 Representing the History of the Schema of an XML Document

Changes in an XML document often require corresponding changes in the document's schema, as the preservation of the same schema might be too restrictive concerning the set of changes that we want to permit. The representation of the history of the document's schema in case that we allow its evolution is as important as the representation of the document's history. In this section we show that our model is powerful enough to represent also the history of the document's schema encoded as XML Schema.

Consider for example the XML document in the left side of Figure 2. The schema for this document may be encoded in XML Schema as follows:

```
<xs:element name="p">
  <xs:complexType>
    <xs:sequence>
        <xs:element name="q" type="xs:string"/>
        <xs:element name="r" type="xs:string"/>
        <xs:element name="s" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

It is easy to see that by deleting the element `<r>v2</r>` (as described in Figure 2), we get an XML document which is not valid with respect to the above schema (in which the existence of the element `r` between the elements `q` and `s` is mandatory). For this reason, it is necessary to modify the document's schema if we want the XML document resulting by applying the deletion to become valid. The new schema might declare that the content of the element p is now an element q followed by an element s. This change can be represented in our formalism by turning the element sequence of the above XML Schema into a multidimensional element with two facets:

```
<xs:element name="p">
  <xs:complexType>
    <@xs:sequence>
      [d in {start..t1-1}]
         <xs:sequence>
           <xs:element name="q" type="xs:string"/>
           <xs:element name="r" type="xs:string"/>
           <xs:element name="s" type="xs:string"/>
         </xs:sequence>
      [/]
      [d in {t1..now}]
         <xs:sequence>
           <xs:element name="q" type="xs:string"/>
           <xs:element name="s" type="xs:string"/>
         </xs:sequence>
      [/]
    </@xs:sequence>
  </xs:complexType>
</xs:element>
```

It is easy to see that the resulting multidimensional schema records the history of the document's schema. Moreover, the snapshots of this schema that refer to specific versions of the XML document (at specific time points), can again be obtained by applying the reduction process.

Another schema that could also validate the document might retain the element r as optional. This could also be encoded in our multidimensional formalism by simply replacing the line:

```
<xs:element name="r" type="xs:string"/>
```

in the initial XML schema description by:

```
<xs:element name="r" type="xs:string"
            minOccurs= [d in {t1..now}]"0"[/]/>
```

where a multidimensional version of the attribute minOccurs has been added, through which it is declared that the attribute minOccurs with value 0 is present in the elements declaration during the time interval t1..now.

7 Conclusions and Future Work

In this paper we proposed the use of MXML as a model for representing the history of XML documents as well as the evolution of their schema. We demonstrated how the effect of the basic change operations on XML documents can be represented in MXML and showed that we can easily obtain temporal snapshots of the XML document from its MXML representation.

The method presented in this paper is based on the ideas proposed by the same authors in [16, 17]. Other works closely related to ours include [20, 19], where the authors represent the interval through which an element holds by adding two extra attributes, namely `vstart` and `vend` whose values are the start and the end points of this interval respectively. However, the approach in [20, 19]: a) does not address the problem of IDREF(S), and b) versions of an element are not explicitly associated as being facets of the same (multidimensional) element. As shown in [17], grouping facets together allows the formulation of cross-world queries, that relate facets that hold under different worlds.

We believe that our approach is more general than other approaches as we allow the treatment of multiple dimensions in a uniform manner. Consequently, our approach may be proved powerful enough to represent multiple versioning not only with respect to time but also to other context parameters such as language, degree of detail, geographic region etc. For instance, incorporating dimensions in semistructured data and XML gives new possibilities for designing Web based applications that deal with context-dependent data. In [11, 12] a web-publishing platform is presented that supports context-dependent delivery, through a model for context which is based on application-specific sets of *characteristics* which are in fact the same as the dimensions of our model. Moreover, a method of constructing multidimensional web pages is presented in [8].

The investigation of efficient methods for storing and retrieving MXML documents and the incorporation of contexts in XPath and XQuery is in our immediate plans for future work.

References

1. S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The Lorel Query Language for Semistructured Data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.
2. T. Amagasa, M. Yoshikawa, and S. Uemura. A Data Model for Temporal XML Documents. In M. T. Ibrahim, J. Kung, and N. Revell, editors, *Database and Expert Systems Applications, 11th International Conference (DEXA 2000), London, UK, Sept. 4-8, Proceedings*, Lecture Notes in Computer Science (LNCS) 1873, pages 334–344. Springer-Verlag, Sept. 2000.
3. S. S. Chawathe, S. Abiteboul, and J. Widom. Managing Historical Semistructured Data. *Theory and Practice of Object Systems*, 24(4):1–20, 1999.
4. S.-Y. Chien, V. Tsotras, and C. Zaniolo. Efficient Schemes for Managing Multiversion XML Documents. *The VLDB Journal*, 11(4):332–353, 2002.
5. S.-Y. Chien, V. J. Tsotras, C. Zaniolo, and D. Zhang. Efficient Complex Query Support for Multiversion XML Documents. In *Advances in Database Technology - EDBT 2002, Proceedings of the 8th Conference on Extending Database Technology*, Lecture Notes in Computer Science (LNCS) Vol 2287, pages 161–178. Springer-Verlag, 2002.
6. Curtis E. Dyreson. Observing Transaction-time Semantics with TTXPath. In *Proceedings of the 2nd International Conference on Web Information Systems En-*

gineering (WISE 2001), Kyoto, Japan, Dec. 2001, pages 193–202. IEEE Computer Press, 2001.

7. M. Gergatsoulis, Y. Stavrakas, and D. Karteris. Incorporating Dimensions to XML and DTD. In H. C. Mayr, J. Lanzanski, G. Quirchmayr, and P. Vogel, editors, Database and Expert Systems Applications (DEXA' 01), Munich, Germany, September 2001, Proceedings, Lecture Notes in Computer Science (LNCS), Vol. 2113, pages 646–656. Springer-Verlag, 2001.
8. M. Gergatsoulis, Y. Stavrakas, D. Karteris, A. Mouzaki, and D. Sterpis. A Web-based System for Handling Multidimensional Information through MXML. In A. Kaplinskas and J. Eder, editors, Advances in Databases and Information Systems (ADBIS' 01), Proceedings, Lecture Notes in Computer Science (LNCS), Vol. 2151, pages 352–365. Springer-Verlag, 2001.
9. F. Grandi and F. Mandreoli. The Valid Web: an XML/XSL Infrastructure for Temporal Management of Web Documents. In T. M. Yakhno, editor, *Advances in Information Systems. First International Conference (ADVIS'02), Izmir, Turkey, 25-27 October*, pages 294–303, 2000.
10. Fabio Grandi. XML Representation and Management of Temporal Information for the Web-Based Cultural Heritage Applications. *Data Science Journal*, 1(1):68–83, 2002.
11. M. C. Norrie and A. Palinginis. Empowering Databases for Context-Dependent Information Delivery. In *Proc. of UMICS' 03*, 2003. (to appear).
12. M. C. Norrie and A. Palinginis. From State to Structure: An XML Web Publishing Framework. In *Proc. of the 15th Conference on Advanced Information Systems Engineering (CAiSE' 03)*, June 2003. (to appear).
13. B. Oliboni, E. Quintarelli, and L. Tanca. Temporal Aspects of Semistructured Data. In *Proc. of the 8th International Symposium on Temporal Representation and Reasoning (TIME-01)*, pages 119–127, 2001.
14. S. Shang and C. E. Dyreson. Adding Valid Time to XPath. In *Database and Network Information Systems, Proceedings of DNIS 2002*, Lecture Notes in Computer Science (LNCS), Vol. 2544, pages 29–42. Springer-Verlag, 2002.
15. Y. Stavrakas and M. Gergatsoulis. Multidimensional Semistructured Data: Representing Context-Dependent Information on the Web. In A. B. Pidduck, J. Mylopoulos, C. Woo, and T. Oszu, editors, *Advanced Information Systems Engineering, 14th International Conference (CAiSE'02), Toronto, Ontario, Canada, May 2002. Proceedings.*, Lecture Notes in Computer Science (LNCS), Vol. 2348, pages 183–199. Springer-Verlag, 2002.
16. Y. Stavrakas, M. Gergatsoulis, C. Doulkeridis, and V. Zafeiris. Accomodating Changes in Semistructured Databases Using Multidimensional OEM. In Y. Manolopoulos and P. Navat, editors, Advances in Databases and Information Systems (ADBIS' 02), Proceedings, Lecture Notes in Computer Science (LNCS), Vol. 2435, pages 360–373. Springer-Verlag, 2002.
17. Y. Stavrakas, M. Gergatsoulis, C. Doulkeridis, and V. Zafeiris. Representing and Querying Histories of Semistructured Databases Using Multidimensional OEM. *Information Systems*, ??(??), 2003. (to appear).
18. Y. Stavrakas, K. Pristouris, A. Efantis, and T. Sellis. Implementing a Query Language for Context-dependent Semistructured Data. Submitted for publication, 2003.
19. F. Wang and C. Zaniolo. Representing and Quering the Evolution of Databases and their Schemas in XML. In *Proc. of SEKE'03*, July 2003. (to appear).
20. F. Wang and C. Zaniolo. Temporal Queries in XML Document Archives and Web Warehouses. In *Proc. of TIME-ICTL' 03*, July 2003. (to appear).