A Web-based System for Handling Multidimensional Information through MXML[⋆]

Manolis Gergatsoulis[1], **Yannis Stavrakas**[1,2], **Dimitris Karteris**[1], **Athina Mouzaki**[1], and **Dimitris Sterpis**[1]

[1] Institute of Informatics & Telecommunications,
National Centre for Scientific Research (N.C.S.R.) 'Demokritos',
153 10 Aghia Paraskevi Attikis, Greece.
[2] Knowledge & Database Systems Laboratory
National Technical University of Athens (NTUA), 157 73, Athens, Greece.
`{manolis,ystavr,mouzaki}@iit.demokritos.gr`
`dkart@tee.gr`

Abstract. In this paper we address an issue common in the frame of WWW, namely information entities that present different facets under different *contexts* (or *worlds*). Handling such multifacet or *multidimensional* entities requires a multidimensional paradigm for Web data, which consists of representation, manipulation and presentation issues. For representing multidimensional data we employ Multidimensional XML, a markup language that incorporates *dimensions* in XML. We discuss the presentation of multidimensional data through multidimensional XSL stylesheets. We describe the design of a system that implements the basic functionality of the multidimensional paradigm, and demonstrates how a user can interact with a multidimensional document and view different variants of the document under different worlds.

Keywords: Multidimensional XML/XSL, Web Databases, Multidimensional Languages.

1 Introduction

The wide acceptance of WWW was due, to a great extend, to the simplicity of its protocol (HTTP) and language (HTML). However, the need for more sophisticated functionality led to secure protocols (HTTPS), flexible languages (XML) [5, 7] and ambitious visions for the future of the Web (Semantic Web [4]).

Viewing the Web as a large database, a number of query languages and data models, such as semistructured data [1, 15], have been proposed. However, those models fall short when it comes to representing *multidimensional information*; that is, information that presents different facets under different contexts. Actually, there are many cases where variants of the same entity do exist. As a

⋆ 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.

simple example imagine a report that needs to be represented at various degrees of detail and in various languages. A solution would be to create a different document for every possible combination. Such an approach is certainly not practical, since it involves excessive duplication of information. What is more, the different variants are not associated as being parts of the same entity. The problem of varying entities is very common in the frame of the WWW, where information providers cannot assume too much about the background *context* of the information consumers. For those reasons, models and languages suitable to represent and exchange multidimensional data over the Web are needed.

Ideas on how this problem can be tackled are given in [14, 13], where a formalism called Multidimensional XML (MXML) is presented. MXML was influenced by *Intensional HTML* (IHTML) [17, 6, 16]. IHTML is a Web authoring language, based on and extending ideas proposed for a software versioning system in [12], that allows a single Web page to have different variants and to dynamically adapt itself to a given context. The main difference between IHTML and MXML is a projection of the difference between HTML and XML, that is, focusing on encoding structure rather than on presentation.

In this paper, we propose a multidimensional paradigm for representing and viewing context-dependent Web data. We present a comprehensive example that demonstrates the syntax and relationships of the various multidimensional components in the paradigm. We describe the architecture and design of a system that implements the basic functionality of the proposed paradigm, and discuss some implementation issues. Finally, we conclude the paper with directions for future work.

2 A Multidimensional Paradigm

The problem of Web information entities that may assume different facets under different *contexts* (or *worlds*) leads to a new paradigm for representing and viewing such entities. We refer to the new paradigm as *multidimensional paradigm*, and in this section we examine its various aspects.

2.1 The Multidimensional Approach

A widely adopted way to encode, represent, and exchange information in the frame of WWW is through eXtensible Markup Language (XML in short) [5, 7]. XML is flexible enough to adapt to different domains, and capable of handling the irregularities often exhibited by Web data. XML does not address the issue of how to present information; a solution to this is offered by eXtensible Stylesheet Language (XSL in short) [2, 8].

XSL can be seen as a document containing instructions on how to present information in XML documents. It is important to note that XSL stylesheets are also XML documents. An XSL stylesheet can be applied to a specific XML document, and the result can be displayed by a Web browser. Actually, a number

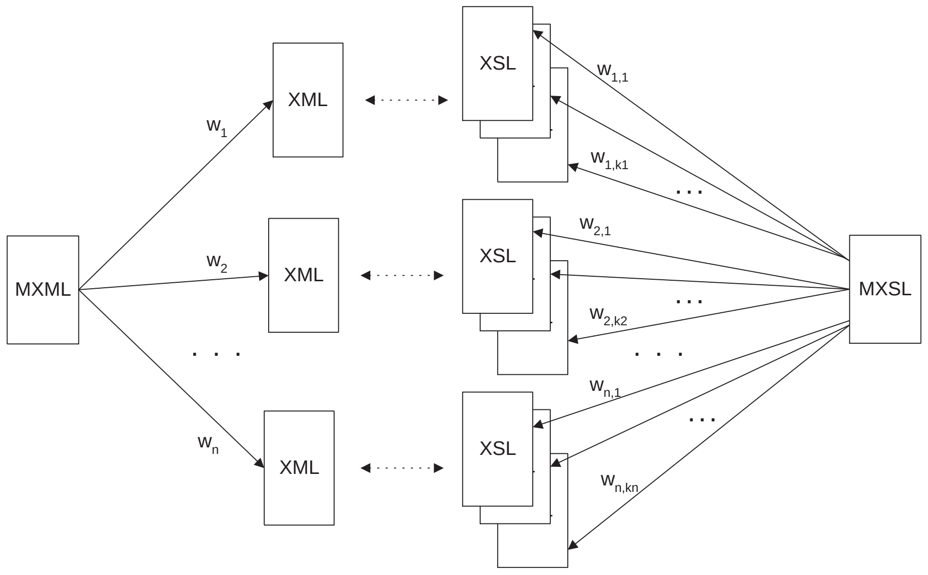


Fig. 1. Multidimensional XML, multidimensional XSL, and possible worlds.

of XSL documents can be applied to the same XML document, resulting in different ways to view that XML document, or parts of it.

A multidimensional approach would allow a single document to have a number of variants, each holding under a specific *world*. A world is defined by giving values to a number of parameters called *dimensions*. We assume that information in such a document is encoded in a suitable markup language called *multidimensional XML* (*MXML* in short). Once a world is specified, such an MXML document can be reduced to a conventional XML document that constitutes the holding facet under that world. This world-associated decomposition of an MXML document to a number of conventional XML variants is depicted in Figure 1, where the possible worlds for the MXML document are denoted as $w_1, w_2 \ldots w_n$.

Since XSL stylesheets are XML documents, the same principles hold for XSL stylesheets as well. A multidimensional XSL stylesheet (*MXSL* in short) encodes a set of conventional XSL stylesheet, each being the facet of the MXSL under a specific world. Like conventional XSLs, an MXSL must be associated with an XML or an MXML document. For each possible world, the holding XSL is applied to the holding XML to give the view of the document under that world. The relation between MXML and MXSL is given pictorially in Figure 1. Note that the possible worlds for an MXSL may not be identical with those of the corresponding MXML. A number of dimensions in an MXSL may relate to the definition of alternative presentations for the same XML document. Besides, some dimensions of an MXML may also be used in the corresponding MXSL to establish a correspondence between the holding variants of MXML and MXSL.

2.2 Multidimensional XML

For supporting the multidimensional approach described above, we propose *Multidimensional XML* (MXML for short) [14, 13, 10], a markup language that incorporates in an elegant way multidimensional capabilities in XML. The notion of *world* is fundamental in MXML. A world represents an environment under which data in a multidimensional document obtain a substance. A world is determined by assigning values to a set $\mathcal{S}$ of *dimensions*. For each dimension d there exists a domain $\mathcal{D}_d$ over which d ranges. A **world** W is defined to be a set which, for each dimension $d \in \mathcal{S}$, contains a pair (d, u), where $u \in \mathcal{D}_d$.

In an MXML document, dimensions may be applied to elements and attributes. An element whose content depends on one or more dimensions is called *multidimensional element*, while an attribute whose value depends on one or more dimensions is called *multidimensional attribute*.

An MXML document uses *context specifiers* which are syntactic constructs that specify sets of worlds. Context specifiers qualify the variants of multidimensional elements or attributes, relating each variant to the set of worlds under which the variant becomes the holding one for the corresponding multidimensional entity.

In MXML syntax, 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>
```

A multidimensional element is denoted by preceding the element name with the special symbol "@". A multidimensional element encloses one or more *context elements* that constitute facets of that multidimensional element, holding under specific worlds specified by the corresponding context specifier. Context elements have the same form as conventional XML elements. All context elements belonging to a multidimensional element have the same name which is the name of the multidimensional element. In MXML, context elements of the same multidimensional element may have different value or structure.

To declare a multidimensional attribute we use the following syntax:

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

A multidimensional attribute is expressed as an attribute whose value is a set of context - value pairs. Each one of those context-associated values becomes the holding value of the attribute under the corresponding context. Therefore, under different worlds, an attribute may evaluate to different values depending on its context specifiers.

A *context specifier* is of the form:

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

where `dimension_i_specifier`, for `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 $!=$, the *dimension_value_expression* consists of a single dimension value. Otherwise, if the *specifier_operator* is either `in` or `not in`, the *dimension value expression* is a set of values of the form $\{\texttt{value}_1, \ldots, \texttt{value}_k\}$, with $k \geq 1$.

In both multidimensional attributes and elements, a context specifier may also be the reserved word "`default`". The `default` context specifier represents all worlds not covered by other context specifiers of the same multidimensional entity. The empty context specifier [ ] is called *universal context specifier*, and represents the set of all possible worlds.

A multidimensional DTD (MDTD in short) has been proposed in [10] for defining constraints on the structure of MXML documents. An MDTD allows to specify dimensions and their respective domains. Moreover, it is possible to impose different constraints under different contexts to an element structure. In this way, an MDTD can specify that an element has different structure under different sets of worlds. A graph data model for MXML has also been proposed in [10] that takes into acount contexts when representing MXML data.

2.3 Reducing MXML to XML

An important point concerning the context specifiers of a multidimensional entity is that they must be mutually exclusive, in other words, they must specify disjoint sets of worlds. This property of multidimensional entities 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.

Notice that the above process can be generalized for producing a facet that holds under a set of more than one worlds. In the general case, that facet is an MXML document which is a pruned version of the original MXML document.

2.4 A Comprehensive Example

As an example of the multidimensional paradigm, 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 multidimensional element whose value depends on the dimensions `edition` and `customer_type`. Note that the element `translator` has substance only under the worlds where `edition` has the value `greek`.

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>
     [edition=english,customer_type=individual]<price>13.000</price>[/]
     [edition=english,customer_type=library]<price>10.000</price>[/]
     [edition=english,customer_type=student]<price>11.700</price>[/]
     [edition=greek,customer_type=individual]<price>5.000</price>[/]
     [edition=greek,customer_type=library]<price>3.000</price>[/]
     [edition=greek,customer_type=student]<price>4.500</price>[/]
  </@price>
</book>
```

The MXML in Example 2 is an MXSL stylesheet for the MXML document in Example 1. This MXSL specifies how the various facets of the corresponding MXML in Example 1 are to be presented.

The `ISBN` is shown only if the request has been made by a library, while `title` and `authors` are shown if the potential client is an individual or a student. Note how the `wrapper` element, which is defined in XSL, can also be used elegantly in MXSL to exclude the `translator` in case the request concerns the original language edition of the book. Finally, `publisher` and `price` are displayed in any case.

Notice that an MXSL may contain multidimensional versions of the elements and attributes defined within the frame of conventional XSL. There is no constraint on which dimensions participate in the MXSL context specifiers; they may also occur in the corresponding MXML document, as is the case of dimensions `customer_type` and `edition` in Example 2, or they can be different, as is the case of dimension `size` in the same example.

Example 2. An MXSL multidimensional stylesheet for Example 1.

```
<xsl:template match="/">
  <DIV STYLE=[size=large]"font-size:22pt"[/][size=normal]"font-size:18pt"[/]>
    Book
  </DIV>
  <@SPAN>
    [customer_type = library]
      <SPAN STYLE="font-size:15pt">
        ISBN: <xsl:value-of select = "book/isbn"/>,
      </SPAN>
    [/]
    [customer_type in {individual, student}]
      <SPAN STYLE="font-size:15pt">
        Title: <xsl:value-of select="book/title"/>,
        Authors: <xsl:value-of select="book/authors"/>,
        <@wrapper>
          [edition=greek]
            <wrapper>
              Translator: <xsl:value-of select="book/translator"/>,
            </wrapper>
          [/]
        </@wrapper>
      </SPAN>
    [/]
  </@SPAN>
  <SPAN STYLE="font-size:15pt">
    Publisher: <xsl:value-of select="book/publisher"/>,
  </SPAN>
  <SPAN STYLE="font-size:15pt">
    Price: <xsl:value-of select="book/price"/>
  </SPAN>
</xsl:template>
```

For the world `w = {(edition, greek), (customer_type, student)}`, the MXML document in Example 1 is reduced to the conventional XML document in Example 3.

Example 3.

```
<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>4.500</price>
</book>
```

For the world w' = $\{$`(edition, greek), (customer_type, student), (size, large)`$\}$, the MXSL stylesheet in Example 2 is reduced to the XSL stylesheet in Example 4.

Example 4.

```
  <xsl:template match="/">
    <DIV STYLE="font-size:22pt"> Book </DIV>
    <SPAN STYLE="font-size:15pt">
       Title: <xsl:value-of select="book/title"/>,
       Authors: <xsl:value-of select="book/authors"/>,
       <wrapper>
         Translator: <xsl:value-of select="book/translator"/>,
       </wrapper>
    </SPAN>
    <SPAN STYLE="font-size:15pt">
      Publisher: <xsl:value-of select="book/publisher"/>,
    </SPAN>
    <SPAN STYLE="font-size:15pt">
      Price: <xsl:value-of select="book/price"/>
    </SPAN>
  </xsl:template>
```

The result of applying the XSL stylesheet of Example 4 to the XML document of Example 3 looks like the following:

> Book
>
> Title: The C programming language, Authors: Brian W. Kernighan, Dennis M. Ritchie, Translator: Thomas Moraitis, Publisher: Klidarithmos, Price: 4.500

Finally, the analogous result for the world w' = $\{$`(edition, english), (customer_type, library), (size, normal)`$\}$ would look like this:

> Book
>
> ISBN: 0-13-110362-8, Publisher: Prentice Hall, Price: 10.000

3 System Architecture

In this section we describe a prototype system that demonstrates the basic principles of the multidimensional paradigm we introduced in the previous sections. The implemented system is called MXML Web Server and is a Web server capable of handling multidimensional data encoded in MXML/MXSL.

In a typical scenario, the user requests an MXML document through a conventional Web browser, and is prompted to select values for each of the dimensions associated with a requested document. After a world has been specified by the user, the server sends the corresponding XML/XSL facet to be displayed by the browser. The user can change the values of dimensions and observe how different worlds are associated to different variants of the same multidimensional document.

An important point is that, in the multidimensional paradigm, the management of dimensions and the reduction of MXML/MXSL can take place at the server, the client, or both. Our system implements this functionality at the server side mainly for reasons of compatibility with existing Web browsers. In the general case, however, some of the dimensions can be considered at server side, to eliminate irrelevant data and reduce the size of the response, while the rest of the dimensions could be handled at client side, for reasons of privacy or for minimizing the number of subsequent requests. This flexibility could be useful and further explored in the frame of applying MXML to domains such as electronic commerce and user modeling.

3.1 Extending URLs

URL (Uniform Resource Locator) [3] is the standard way to specify a resource available on the Internet. We extend the syntax of URL in order to enable it to handle multidimensional information. The extended URL has the form:

```
http://<host>:<port>/<path><context>?<search>
```

The only difference from conventional URL is that the token `<context>` has been added, in order to incorporate dimensions. In the following example we show an extended URL that represents a request for the variant of the document named `books.mxml`, where `edition` is `english` and `customer_type` is `student`.

```
http://myserver/books.mxml[edition=english,customer_type=student]
```

3.2 Design and Operation

The system comprises the software modules illustrated in Figure 2. These modules are: Request Analyzer, MXML Request Decomposer, View Extractor, MXML Response Composer, and Conventional Web Server. In the following paragraphs we shall briefly describe the functionality of each of those modules, as well as their submodules, and discuss their role in the system operation.

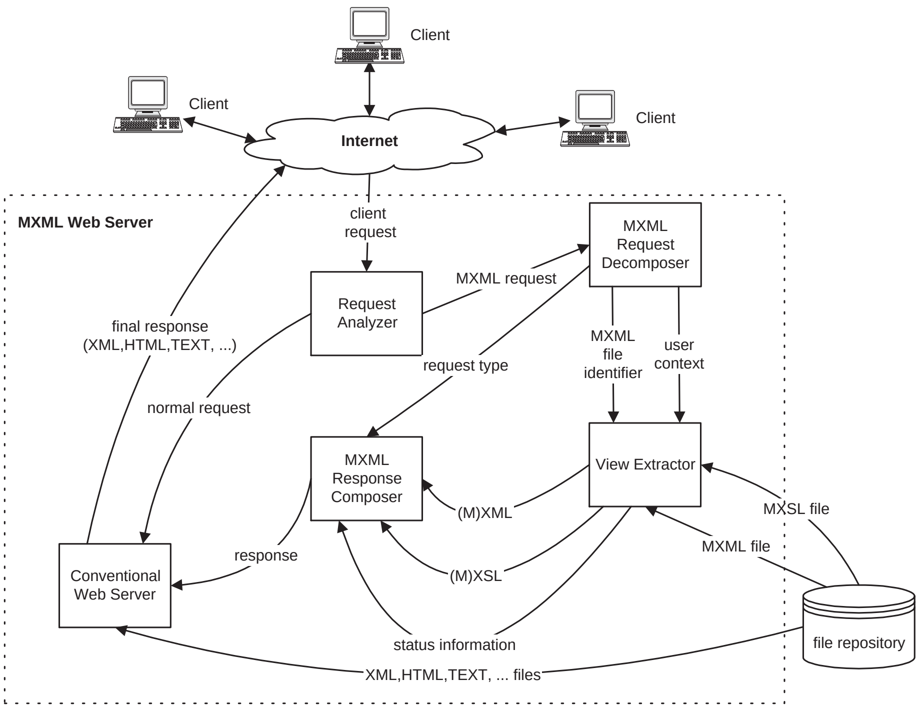


Fig. 2. MXML Web Server architecture.

Request Analyzer: The Request Analyzer is responsible for determining the type of client requests. There are two types of requests; *normal* requests, which refer to HTML, XML, text files etc., and *MXML-requests*, which refer to MXML files. If the request is of normal type, then the Conventional Web Server handles it. Otherwise, if the request involves an MXML document, then the MXML Request Decomposer is invoked to serve the request.

MXML Request Decomposer: This module decomposes the MXML request into sections. The three sections that comprise the MXML request are: the MXML File Identifier, the User Context and the Request Type. The first two sections are sent to the View Extractor module, while the third is sent directly to the MXML Response Composer.

View Extractor: This module combines the information provided by the MXML Request Decomposer with the corresponding MXML or MXSL files in order to serve the request. The View Extractor consists of two sub-modules called MXML Parser and MXML Specializer, as it is depicted in Figure 3(a).

– **MXML Parser:** The MXML Parser uses the requested MXML File Identifier, which is provided by the MXML Request Decomposer, to access the

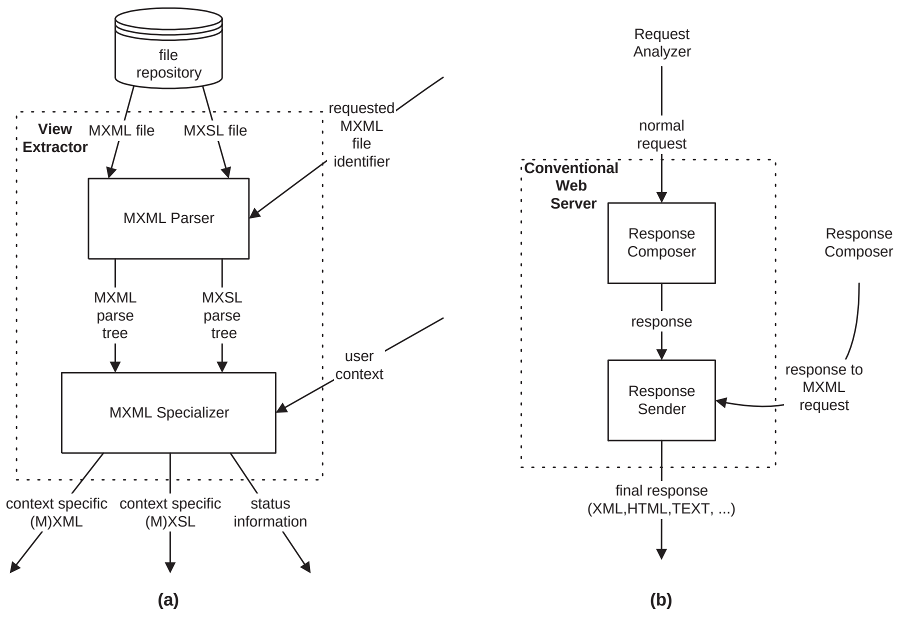


Fig. 3. (a) View Extractor's sub-modules, (b) Conventional Web Server's sub-modules.

corresponding MXML or MXSL file from the File Repository. Its role is to parse the file and generate the corresponding MXML or MXSL parse tree.

– **MXML Specializer:** The MXML Specializer uses the parse tree that is generated by the MXML Parser, and the User Context, which is provided by the MXML Request Decomposer, to generate a context specific (M)XML or (M)XSL file. This is done by reducing the original MXML/MXSL file to an (M)XML/(M)XSL file that holds under the specified context. The type of the produced file is passed to MXML Response Composer through the status information.

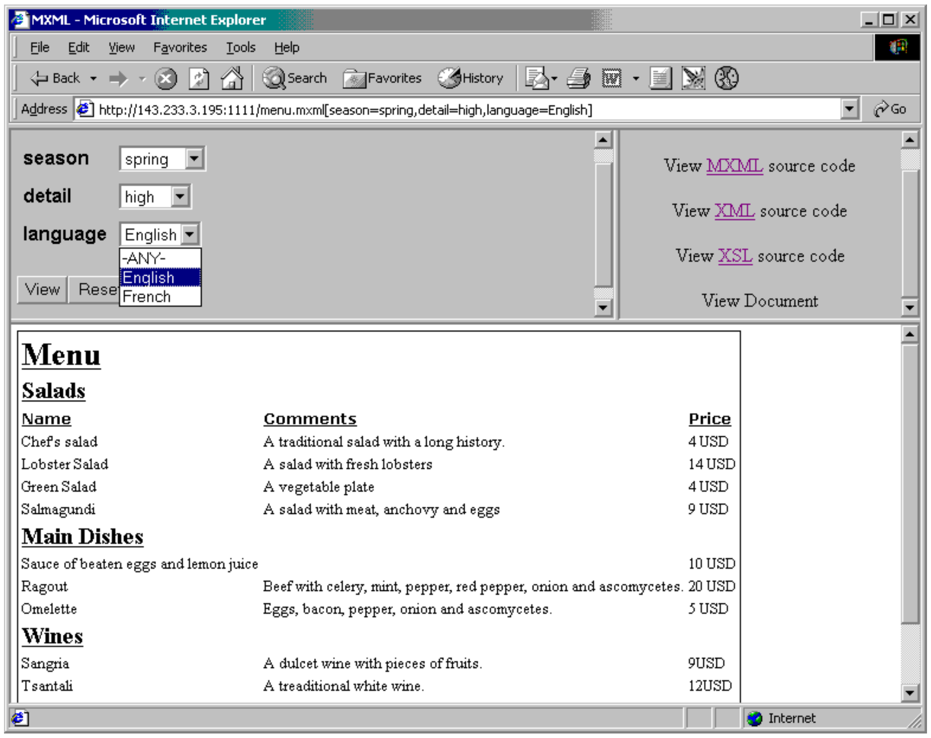
MXML Response Composer: The MXML Response Composer constructs the response and sends it to the Response Sender, which is responsible for dispatching the response to the client.

Conventional Web Server: The Conventional Web Server implements some of the essential features of a conventional Web server. It consists of two submodules, shown in Figure 3(b). In case the type of request is *normal*, the response is constructed by the Response Composer submodule that is part of the Conventional Web Server, whereas in case the type of request is *MXML-request* the response is produced in MXML Response Composer module and is passed to the Response Sender submodule of the Conventional Web Server.

- **Response Composer:** The Response Composer analyzes the normal request, and constructs the response that is sent to the client. As it is depicted in Figure 3(b), this module is engaged in a client request service procedure only if this request is of type *normal*.
- **Response Sender:** The Response Sender is responsible for sending the response to the client. As stated above, the response originates either from the Response Composer, or from the MXML Response Composer.

4 Implementation

The system components described in the previous section are in fact implemented as two separate programs. The MXML Web Server (except View Extractor) was developed in Java in order to take advantage of the powerful features that this language offers for network programming. The View Extractor has been implemented in ANSI C.



http://143.233.3.195:1111/menu.mxml[season=spring,detail=high,language=English]

season: spring
detail: high
language: English

Menu

Salads

Name	Comments	Price
Chef's salad	A traditional salad with a long history.	4 USD
Lobster Salad	A salad with fresh lobsters	14 USD
Green Salad	A vegetable plate	4 USD
Salmagundi	A salad with meat, anchovy and eggs	9 USD

Main Dishes

Sauce of beaten eggs and lemon juice		10 USD
Ragout	Beef with celery, mint, pepper, red pepper, onion and ascomycetes.	20 USD
Omelette	Eggs, bacon, pepper, onion and ascomycetes.	5 USD

Wines

Sangria	A dulcet wine with pieces of fruits.	9USD
Tsantali	A treaditional white wine.	12USD

Fig. 4. A screenshot showing a reply from MXML Web Server.

Figure 4 is a screenshot of the reply of MXML Web Server to the request:

```
http://143.233.3.195:1111/menu.mxml[season=spring,
      detail=high,language=English].
```

The context specifier in this extended URL is composed dynamically by JavaScript code, based on the values selected by the user for the dimension drop-down lists, in the upper left *dimension frame*. Note that for every dimension there exists a value "ANY" in the corresponding drop-down list. The meaning of "ANY" is that no value is specified for that dimension. Consequently, if a context specifier contains one or more "ANY", it specifies more than one worlds. If it does not contain "ANY" at all, it specifies exactly one world.

The upper-right *action frame* allows the user to view the MXML source code, and the MXSL source code. If the selected context specifies exactly one world (does not contain "ANY") then the user may also see the XML and XSL sources of the corresponding facets under that world, plus the final output document. The source code and the final document are displayed in the lower *output frame*, which in the case of Figure 4 shows a variant of a multidimensional menu of a restaurant corresponding to the world `w = {(season, spring), (detail, high), (language, English)}`.

The implemented system can be reached at the URL:

```
http://www.iit.demokritos.gr/~mxml
```

5 Conclusions and Future Work

In this paper we proposed a paradigm for handling multidimensional data in the frame of the Web. We proposed the use of MXML to represent multidimensional information, and showed how multidimensional stylesheets can be expressed in MXSL. Through a comprehensive example we demonstrated the reduction of MXML/MXSL to conventional XML/XSL under a specific world. Finally, we presented a system that implements the basic functionality of the proposed paradigm, and discussed some implementation issues.

Our future plans include the investigation on possible applications of MXML in diverse fields, such as electronic commerce and digital libraries. We also consider the representation of time dependent data through MXML [11], the representation of cartographic and GIS information where a possible dimension is *scale*, and applications where user profiling information plays an important role for delivering the right data.

Since MXML is primarily considered as a data model and data exchange format, an interesting direction is how to efficiently store and retrieve MXML data. Approaches used for XML, that employs RDBMS for storing XML fragments [9], seem promising for adaptation to MXML.

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. S. Adler, A. Berglund, J. Caruso, S. Deach, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman, and S. Zilles. Extensible Stylesheet Language (XSL) Version 1.0. http://www.w3.org/TR/xsl, 2000.
3. T. Berners-Lee, L. Masinter, and M. McCahill. Uniform Resource Locators (URL). http://www.ietf.org/rfc/rfc1738.txt, 1994.
4. Tim Berners-Lee. Semantic web road map. http://www.w3.org/DesignIssues/Semantic.html, 1998.
5. T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible markup language (XML) 1.0 (second edition). http://www.w3.org/TR/REC-xml, October 2000.
6. G. D. Brown. IHTML 2: Design and Implementation. In W. W. Wadge, editor, *Proceedings of the 11th International Symposium on Languages for Intensional Programming*, pages 1–13, 1998.
7. 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.
8. James Clark. XSL Transformations (XSLT). http://www.w3.org/TR/xslt, 1999.
9. M. Fernández, W.-C. Tan, and D. Suciu. SilkRoute: Trading between Relations and XML. *Computer Networks*, 33(1–6):723–745, 2000.
10. M. Gergatsoulis, Y. Stavrakas, and D. Karteris. Incorporating dimensions to XML and DTD. *It will be presented at International Conference on Database and Expert Systems Applications (DEXA' 01), Munich, Germany September, 2001.*
11. T. Mitakos, M. Gergatsoulis, Y. Stavrakas, and E. V. Ioannidis. Representing time-dependent information in multidimensional XML. *Proc. of the 23rd Int. Conf. "Information Technology Interfaces" (ITI'01), Pula, Croatia, June 2001.*
12. J. Plaice and W. W. Wadge. A New Approach to Version Control. *IEEE Transactions on Software Engineering*, 19(3):268–276, 1993.
13. Y. Stavrakas, M. Gergatsoulis, and T. Mitakos. Representing context-dependent information using Multidimensional XML. In J. Borbinha and T. Baker, editors, *Research and Advanced Technology for Digital Libraries, 4th European Conference ECDL'2000*, Lecture Notes in Computer Science (LNCS) 1923, pages 368–371. Springer-Verlag, 2000.
14. Y. Stavrakas, M. Gergatsoulis, and P. Rondogiannis. Multidimensional XML. In P. Kropf, G. Babin, J. PLaice, and H. Unger, editors, *Distributed Communities on the Web, Third International Workshop (DCW'2000)*, Lecture Notes in Computer Science (LNCS) 1830, pages 100–109. Springer-Verlag, 2000.
15. D. Suciu. An overview of semistructured data. *SIGACT News*, 29(4):28–38, December 1998.
16. 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.
17. T. Yildirim. Intensional HTML. Master's thesis, Department of Computer Science, University of Victoria, 1997.