From Database to Web multimedia documents

Philippe Mulhem, Hervé Martin

From Database to Web multimedia documents

Philippe Mulhem
IPAL-CNRS, KRDL, Singapore
mulhem@krdl.org.sg

Hervé Martin
LSR-IMAG, Grenoble, France
Herve.Martin@imag.fr

Abstract: *This paper deals with the integration of multimedia and database technologies in order to describe web multimedia documents. We present a middleware to seamlessly handle database accesses as well as compositional, spatial and temporal constraints related to data presentation. Our approach is based on the concept of Templates. A template is a logical presentation unit that merge database queries with layout specifications. We choose an XML and SMIL approach to implement template. Template definition and invocation are mapped into a XML DTD. Each template is then translated into a SMIL document. In this paper, we give an example to show the advantages of our approach.*

Keywords: Database, Multimedia Document, XML, SMIL.

1. Introduction

Multimedia softwares allow both manipulation of individual data[1] and multimedia documents[2]. Most of authorwares also provide gateways to databases, mainly for storage purposes. An example of such a trend is given by the partnership between Informix and MEDIAstra Inc. in order to propose solutions for storage and manipulation of video data.

[1] Adobe Systems Inc. proposes Photoshop 5.5 for dealing with photos and Premiere 5.1 for video data

[2] Director 8.0 from Macromedia Inc. is an example of a such software

Recent work on multimedia databases [8, 24] show that databases can fulfill some multimedia data requirements both from a storage point of view and a modeling point of view. On the other hand, the link between the web and databases allows to solve some problems of data presentation. Nevertheless, in a such environment, it is difficult to keep all advantages of Web, multimedia and databases.

Yet, the integration of database languages and *Web* document formats is a key ingredient in applications such as electronic commerce and virtual enterprise. Typical tasks in such contexts are for instance: on the fly generation of item lists presentation, depending on user's needs and profiles and stocks. DBMSs allow maintaining different contents. A standard like on the eXtented Markup Language (XML) [32] has some capabilities for presenting data in different ways.

Currently, web presentations of multimedia data is not easily done because databases do not usually handle the meaning of stored data, and because multimedia presentation languages neglect the database aspect related to collections of data. Our approach consists of taking the advantage of the numerous advances existing on XML, database query languages and multimedia document presentation languages like Synchronized Multimedia Integration Language (SMIL) [29], in a way to unify them. More specifically, this paper introduces solutions:

1. To define document parts as presentations of database query results
2. To integrate spatio-temporal constraint specifications and processing on a collection-based environment

3. To deal with current or ongoing normalization work related to multimedia document presentations.

To tackle these three points, we introduce a middle layer between usual database management systems and application layers. This middle layer implements a component model based on a *template* paradigm. Templates act as an interface between database query languages and XML. We choose XML for improving portability and reusability in various contexts. Templates define a three-dimension space:

1. Database embedding. In the definition of templates, the part in which database elements can be accessed is bounded, in a way to avoid errors or inconsistencies during database accesses. Each database query result is embedded into templates.
2. Temporal constraints. The ways templates are presented are constrained according to collection constraints. We use declarative interval-based constraints inspired from Allen relations [4], but applied on the presentation time interval of templates.
3. Spatial constraints. It is important to be able to handle where data should be presented (at least for visual or text data). This definition has also to support collections of data.

The main contribution of this paper is to reduce the gap between authoring tools and database management systems. Another contribution is to propose extends to conventional query languages instead of use graphical interface for specifying presentation. The advantage of our approach is to point out several levels in multimedia

presentations. Several attempts in this area exist but to the best of our knowledge there is no work proposing such a complete integration.

The remaining of the paper is organized as follows. The next section introduces in details related works. The section 3 is dedicated to the presentation model and the definition of the templates. Section 4 deals with the links between SMIL documents and templates. Section 6 introduces an example to show some advantages of our approach. Finally, we give some concluding remarks and propose some perspectives.

2. Related works

2.1. Web and Database Products

All major database products propose an interface with the Web. In this section, we present some representative products: Versant, Objectstore and O2 for object technology; Oracle, Informix and Microsoft SQL Server / Access (referred later as MSQLS/A) for relational technology. We study these systems from two points of views:

i) The way the query results are related to real Web pages (i.e. by a programming language or by an HTML template) and

ii) The way presentation definitions are stored. This point is important because when the definition material is stored in the database, the consistency between the presentation and the data is maintained easily.

Informix and Microsoft SQL Server / Access products use HTML templates to link the database and the Web. Such templates use HTML tags as well as additional syntactic elements that indicate how to integrate query results into generated HTML pages. The

HTML and the additional tags handle the spatial presentation features; so the expressive power of how things are spatially presented is low. However, Informix allows the definition of reusable user-defined template tags that are translated by procedures into real HTML tags when presenting results. Informix stores the templates in the database while Microsoft MSQLS/A manages the templates in files outside the database.

Oracle uses a programming language (PL/SQL) to express how Web pages are generated. The programs are not stored in the database itself. The use of a programming language allows complex operations. When a Web page is generated, the program translates procedure calls into HTML tags. This approach allows the definition, and the reuse, of procedures that define complex visual presentations. If the HTML language evolves, the procedure called can be modified without modifications to the calling programs. Compared to the Informix approach, Oracle does not allow the use HTML tags directly because an HTML tag is put in a result page by a call to a procedure, however the user-defined tags in Informix are somewhat similar to the procedure calls in Oracle.

For each object oriented DBMS studied, the approaches are roughly the same. **Objectstore** manages templates using the ObjectsForms template processor. The query processor can call functions for complex use. The templates are however not stored in the database. The **Versant** product does split template documents (outside the database) and views (in fact sub-templates dedicated to the presentation of one object). One object can have several possible views. The interesting point is that the views are associated with classes (and therefore stored in the database schema), and can then be inherited. The **O2** system proposes to define the presentation of objects using methods. Default

presentations are proposed. Because conventional class methods are used, inheritance applies. The idea of putting HTML tags directly in methods is less robust: all the methods related to the presentation of objects have to be modified as HTML evolves.

To sum up, none of these systems supports temporal expressions for query result presentations, and spatial features of result presentations are mostly imperative (by templates or programs). Almost all systems provide a way to define complex presentations using programs, but no language definition specifically dedicated to the query result presentation is provided. In this paper, we introduce such a language.

2.2. Presentation of query results

Some approaches consider structured (or semi-structured) documents as input-only for the database. Works used this approach on relational databases [9] or object-based databases like POQL [2]. The query language of the database is extended to retrieve documents and documents parts. Foundations of such works can be found in ORION [16], O2FDL [23] or the OQL language defined in [3].

Other works consider structured or semi-structured documents as input of the database, but they are also capable to generate structured documents as query results.

We focus first on works done for HTML documents. WebSQL [24] uses a relational database, and queries are based on a "Select-From-Such That-Where" pattern to allow complex *From* expressions. Path expressions are supported. Results are defined in HTML tables. WebOQL [5] allows to define the format of query results and to reuse the results.

The structure used in this system are hypertrees (i.e., ordered arc-labelled trees with two types of arcs: internal arcs for compositional links, and external arcs for references among objects). Sets of related hypertrees are grouped into webs. Path expressions in hypertrees and webs can be expressed, and hypertrees can be translated into HTML documents. WebOQL is a complex language, that is why a kind of template is used as a front-end.

Query languages for XML documents have also been proposed. XML-QL [13] is a simple language that queries and builds XML documents. XML-QL allows the extraction of parts of XML documents and has the ability to perform joins and agregates. XML-QL provides path expressions that use stars "*" to retrieve nested parts elements to an abitrary depth, "|" for alternation and "." for concatenation in such path expressions. The OQL-S query language of the Ozone system [18] is based on a representation of XML documents that mixes OEM and objects as defined in ODMG [10]. It allows method calls of ODMG objects in query expressions as well as path expressions along OEM objects.

In all the languages above, no consideration is made for the presentation of documents. One can argue that the generated of XML documents may express such presentation constraints, but in this case we are not able to handle the semantics of the presentation constraints soon enough, and the complexity of the query expression should increase.

That is why some researches focus on presentation of database query results. The SQL+D [8] proposals deal with presentation of relational data. A SQL+D query uses the usual "SELECT FROM WHERE" clause, and adds a "DISPLAY" clause in which we express the presentation features of the results. SQL+D distinguishes between multimedia

documents (without a temporal presentation schedule) and multimedia presentation that put a temporal schedule on multimedia documents. The writing (and the reading) of the query presentations using layers is delicate.

In accordance with SQL+D, we should be able to define the presentation of multimedia information from a database. The use of sub-queries can facilitate the definition of presentations and their reuse, based on a Cartesian decomposition of the problems and the algorithms.

3. The presentation model

This section presents the model that supports the logical presentation of database objects. First, we introduce the *Template* concept. A *Template* is a logical presentation unit. Any *Template* has a unique identifier, and can be composed of several components. A component is either a template or a query expression specified using the OQL query language [3]. *Template* definitions specify spatio-temporal constraints between components. In this section, we define *Template* structure, and then we introduce spatial and temporal features.

3.1. Template Definition

To define a template, we have to specify, the name of the template, its structure and various synchronization constraints. For instance, the template t1 of figure 1 illustrates the presentation of a collection of laboratories (sp1). Each laboratory p1 is selected using

a query expression and is presented according to the definition of another template, termed t2. This template is composed of a single component t1.c1. The different synchronization constraints are presented below.

```
Template : t1(spl : set(laboratory));
Components : tuple(t1.c1 = select t2(pl) from pl in spl);
Sp_Synchro: link*(t1.c1);
Te_Synchro : seq*(t1.c1);
Te_Duration : 400
```

Figure 1. A template definition example

A Template T is formally defined as a quadruplet {Id, IP, C, P} :

$$T = Id \times IP \times C \times P$$

Where:

- Id is the set of *Template* logical Identifiers. As we show below, a *Template* is implemented as a class, and its identifier will be the class name. At presentation time, a Template class may have several instances.
- IP is the set of Template Inputs. An input is responsible of the communication between templates. From an object perspective, it corresponds to message parameters.
- C is the set of *Template* components. A component identifies several logical elements linked by spatio-temporal relationships. An element is a query expression, that may reference other templates. For instance, to present a set of employees working in a project, we can define two templates: T_EMPLOYEES and T_EMPLOYEE. The content of T_EMPLOYEE is a tuple composed of data to present for an employee (e.g. his photo and his C.V.). The content of T_EMPLOYEES is a query that returns the templates of the employee objects to be presented according to T_EMPLOYEE

specification. T_EMPLOYEES is then called a *collection component* and T_EMPLOYEE is an *atomic component*.

- P is the set of spatio-temporal constraints related to the template and to its components. In our previous example, employees can be presented either sequentially or in parallel, and for each employee, it is allowed to specify presentation duration.

3.2. Spatial Description

The aim of the spatial description is to specify presentation layout. This task is performed in three steps.

First, we split the space according to the components that must be presented simultaneously. For example, to present one employee we assign one region to display his photograph, and another for his identity. Consequently, we split the presentation space into two regions.

Second, we define how the presentation space split is achieved. We postulate that the spatial presentation space of a template is a rectangle. It is allowed to divide this rectangle both horizontally and/or vertically. Figure 2 shows some possibilities for splitting a space into four rectangular regions.

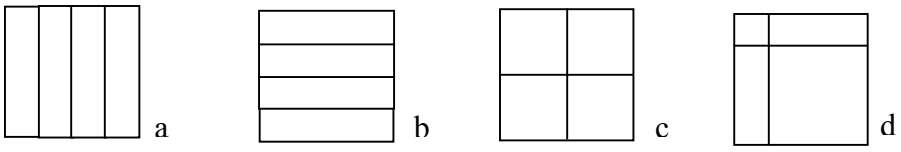


Figure 2. Example of divisions of a display workspace into 4 regions.

Third, each region can be assigned to a *Template* component, which can itself be assigned to several regions. Suppose we have to display two components C1 and C2 using the split of figure 2.c. The figure 3 shows valid distribution for regions, without considering vertical/horizontal symmetries or rotations between c1 and c2, neither allowing overlapping between regions.

Such distribution is defined in the clause *Sp_Synchro*. The number of horizontal and vertical partitions is defined using Sp_H^n for horizontal split and Sp_V^n for vertical split. The superscript number denotes to the number of slices of the presentation space. For instance an "Sp_H^2" defines two horizontal slices.

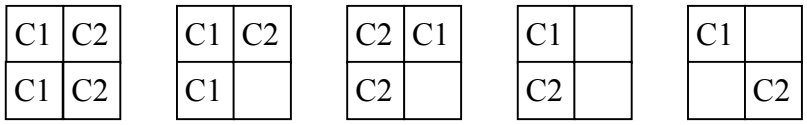


Figure 3. Valid assignments for two components in a 4-regions space

The width of the slices or the height of the slices in the case of horizontal split is expressed in percentage. For instance, *$Sp\_H^2(40, 60)$* assigns 40% to the first region and 60% to the second.

It is allowed to associate a vertical split with a horizontal split. In this case, each horizontal slice is split into vertical sub-slices. After such splits, regions are numbered from (1,1) to (nv, nh) where nh is the number of horizontal slices and nv the number of vertical slices. If it is omitted, we assume that the value of horizontal (resp. vertical) splits is equal to 1.

The second step of the spatial definition assigns components to regions. This assignment is done in clause *Sp_Synchro* using two primitives: *Sp_link* and *Sp_link**. The *Sp_link* primitive associates a single component with a region, and *Sp_link** associates each component belonging to a collection with a region. When using *Sp_link*, the part of the region associated with components is defined using the abscissa and ordinate of the related regions. The choice to associate a collection component with each presentation region helps to the specification of hierarchic components, and thus fosters the reuse of such components.

Consider for instance the following template :

```
Template : T2(l : laboratory);
Components: tuple(t2.c1, t2.c2);
Sp_Synchro : Sp_H2(40,60);
             Sp_link (t2.c1, 1,1); Sp_link (t2.c2,2,1);
```

The template T2 is composed of two components t2.c1 and t2.c2. The space associated with T2 is split in two regions. The coordinates of the first region is (1,1) and the coordinate of the second region is (2, 1). Thus, *Sp_link (t2.c1, 1,1)* specifies that the component t2.c1 is associated with the region (1, 1).

In the context of a collection component CC, and a split *Sp_V3(25,50,25).* Then, *Sp_link*(CC*) indicates that the first element of CC is associated with the region (1,1), the second one to (1,2), the third one to (1,3), the fourth one to (1,1) and so on.

This definition does not manipulate the syntactic elements of the target language like HTML for instance, and this approach is thus more flexible according to the target language evolution.

As suggested by [14], spatial relations can split into three categories: (1) topological relations such as inside, outside and overlap which are invariant under topological transformations of the reference objects; (2) metric relations in term of distances and directions such as north, south, west and east; (3) 3-D relations concerning the partial order of spatial objects as described by prepositions such as in front of, behind and below. Such possible relations between three objects are more than 200. Our spatial model considers that all objects do not intersect but may touch each other. We also handle directions and distances metric relations by the definition of the split of the space and the mapping proposed. 3D relations are not under our consideration, but the temporal model (see part 3.3.) handles stacking. Our goal is not to provide a sound reasoning about spatial relations. Thus, we do not consider works such as [19, 26]. The spatial model proposed is close to the one of SMIL, and we additionally exploit templates compositions by allowing multi-scale spatial split relative to the available space for a template.

To summarize, we allow first a simple presentation specification of a square 2D space (topological and metric relations), and second a link between a part of the space and a database object or another template.

3.3. Temporal Description

The goal of such a description is to temporally constrain component presentations, and to specify a temporal duration constraint on the template itself. When a template is

composed of several components, it is possible to specify whether they are presented in sequence or in parallel. It is specified using *seq* and *par* constraints. We propose the following synchronization constraints inspired from [22] where C1 and C2 are atomic or collection components, and C is a collection component:

seq(Ci, Cj): Ci and Cj must be presented in sequence.

par(Ci, Cj): Ci and Cj must be presented in parallel.

seq-meet(Ci, Cj): Ci and Cj must be presented in sequence with no delay between presentations.

par-equal (Ci, Cj): Ci and Cj must be presented in parallel. Moreover, they must begin and finish simultaneously.

par-start (Ci, Cj): Ci and Cj must be presented in parallel. The two presentations must begin simultaneously. Duration of components Ci and Cj can differ.

par-finish (Ci, Cj): Ci and Cj must be presented in parallel. Moreover, the two presentations must finish simultaneously. Duration of Ci and Cj can differ. The presentation must be stopped when either Ci or Cj terminates its presentation.

par-during(Ci, Cj): Ci and Cj are presented in parallel and can have different durations. The presentation of Ci must begin after Cj starts and must finish before Cj stops.

seq*(C): all components belonging to C must be presented in sequence.

par*(C): all components belonging to C must be presented in parallel. This constraint is equal to a conjunction between par-start*(C) and par-finish*(C).

par-start*(C): all components belonging to C must be presented in parallel. Moreover, all the presentations must begin at the same time. Each component has its own duration.

par-finish*(C): specifies that all components belonging to the C must be presented in parallel. Moreover, all the presentations must finish at the same time. The components belonging to C can have different durations. The presentation must be stopped as soon as one component has terminated its presentation.

A problem may occur when considering collection components: if the number of regions in which the collection is presented is smaller than the number of elements in the collection, the system has to decide how to present these elements. In this case, we propose to use stacks of components presentations. A stack contains elements to display one by one at the same location. For a collection of components, several stacks can be presented simultaneously. For example, to present the photo of each employee belonging to a set, we can use four stacks to constrain employee photos to be presented four by four. We define two spatio-temporal primitives:

stk_par*(C): changes synchronously the top elements of any related stack,

stk_ind*(C): considers each stack as independent, but when the presentation of one stack ends, it stops related stack presentations.

The stack mechanism differs from usual multimedia presentation systems. It allows us to specify a synchronization inside a component presentation. Such a synchronization is crucial for dealing with large collections of objects.

For instance, consider a collection component CC1={C1, C2, C3, C4}, presented using two stacks. To illustrate the explanation, consider that the components C1, C2, C3 and C4 are the images of figure 4. The figure 5 explains the behavior of the stack presentation of CC1 when considering both *stk_par** and *stk_ind** primitives. In figure 5, the initial state

is a two-stacked presentation, where the left stack contains the presentations of C1 and C2, and the second the presentations of C3 and C4. If the presentation of CC1 is described by *stk_par*,* then C2 is presented to C4 after C1 or C3 ends. If the presentation of CC1 is described by *stk_ind**, C2 can be presented without an effect on the stack to the right, and C4 can be presented without effect on the stack to the left.

Figure 4. the templates C1, C2, C3 and C4 of CC1

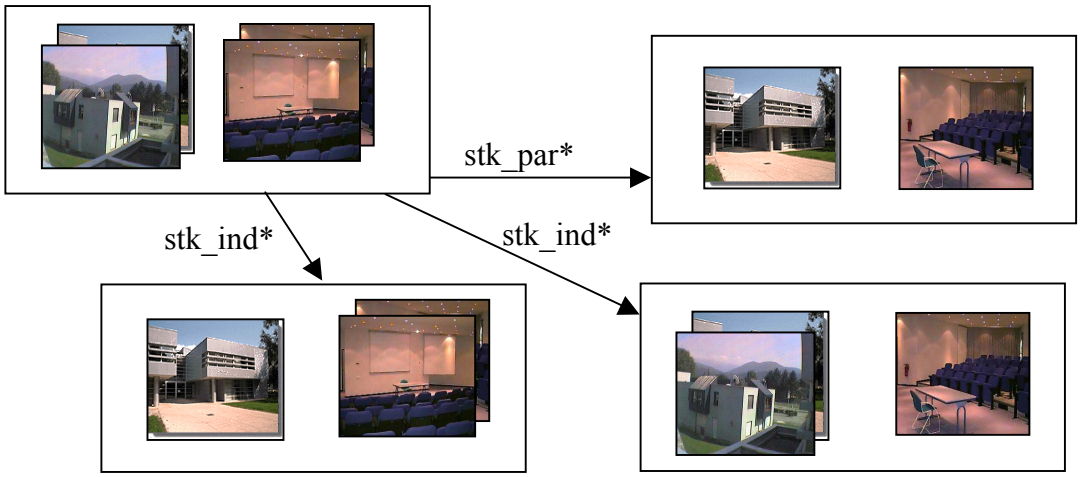


Figure 5. The temporal synchronization of CC1 using two stacks

It is possible to set a template time duration. This constraint can be either an integer greater than zero to specify the maximum duration in seconds, or equal to -1 to indicate that the duration constraint depends only on the context of the template presentation (i.e. the temporal duration of a composed template, or a user driven duration). When conflicts

between several duration constraints occur, the priority is given to the duration of the composed template. For sequential constraints, the duration is equally split into the different components. This priority also applies with stack-based presentations.

Syntactically, the temporal description is then composed of two parts that are the temporal synchronization and the duration :

```
<Te_declaration> ::= "Te_Synchro :" <Te_synchronization>
                     "Te_Duration:" <Te_duration>
```

This approach is also used when objects to be present cannot respect their own duration constraint. A simple way to validate the constraints is to use a resolution process that, as previously explained, sets priority to the composed elements over the composing ones.

Considering the two classes of temporal models defined in [31], our approach is clearly interval-based. Time-based temporal models, used by most authoring tools, lack of flexibility but are widely used because they are easy to implement. Such models are limited to point-to-point relations like *before*, *simultaneously* and *after*. Interval-based existing models (like the well known Allen temporal algebra [4], or the work of Shi and Chang [28]), are able to express more complex relationships. [21] uses Petri nets where places sets the intervals and transitions synchronize end points of intervals. In [17], causality links are introduced between temporal intervals. An interval-based model is more suitable in our case because we do not know a priori which object will be presented. We extended existing temporal relations to handle n-ary temporal relations. The temporal model allows specifying (1) the duration of the complete presentation, (2) the duration of the presentation of each element of the list, (3) the delay between object presentations and (4) the synchronization among the presented objects.

4. Implementation

In this section, we present how templates have been implemented using XML and SMIL standards. The general architecture of the system is presented in Figure 6 below. A clent send a request to the web server. Then, the Web server performs the *cgi* program that interacts with Templates manager in order to create an XML document. This document is returned to the client. The first part of this section concerns the translation of template definition into an XML format. The second part is related to the translation from templates to SMIL.

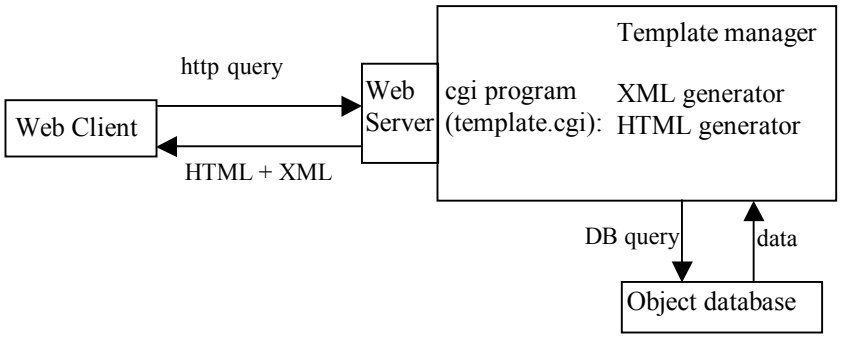


Figure 6. General architecture of the implementation

4.1. XML DTD for Template Definition

The translation of template definition into an XML format is realized using a XML Document Type Definition presented in Figure 7.

```
<!DOCTYPE template PUBLIC "-//TEMPLATE 1.0//EN" "TEMPLATE_DEF1.0.dtd">
```

```
<!ELEMENT template (parameters, components, sp_synchro, te_synchro, duration) >
<!ATTLIST template id ID #IMPLIED>

<!ELEMENT parameters (parameters)*>
<!ELEMENT parameter (#PCDATA) >

<!ELEMENT components (components)* >
<!ELEMENT component ((component_name, sql_with_ref_template))* >
<!ELEMENT component_name (#PCDATA) >
<!ELEMENT sql_with_ref_template (select, from, where?) >
<!ELEMENT select (#PCDATA | ref_template)* >
<!ELEMENT ref_template (call_parameters)* >
<!ATTLIST ref_template ref IDREF #IMPLIED >
<!ELEMENT call_parameters (#PCDATA) >
<!ELEMENT from (#PCDATA) >
<!ELEMENT where (#PCDATA) >

<!ELEMENT sp_synchro (#PCDATA) >
<!ELEMENT te_synchro (#PCDATA) >
<!ELEMENT duration #NDATA>
<!ATTLIST duration unit (ms|s|m|h) "s" >
```

Figure 7. The DTD for template definition

Such template definition is sent to the Template Manager for processing (as presented in figure 8). The verifications of the syntax for the inner elements are done by the template manager and not by the DTD syntax. However, we choose to put explicitly reference call to templates, in a way to check the existence of such templates in the XML tool.

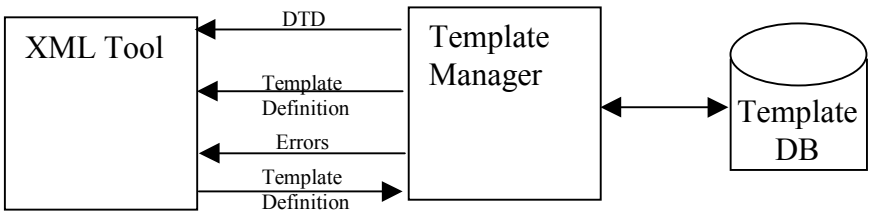


Figure 8. Template manager interactions.

Consider the template of figure 1, dedicated to apply the same presentation constraints to each of the elements of a set of Laboratories from a database. The laboratories are presented in sequence and occupy all the display area. According to the definitions of figures 8 and 9, the corresponding XML template definition is given in figure 10. We notice the simplicity of the translation of a template definition into an XML document.

```
<?xml version="1.0" PUBLIC "-//TEMPLATE 1.0//EN" "TEMPLATE_DEF1.0.dtd" ?>
< template id="t1">
<parameters>
        <parameter>set(Laboratory)</parameter>
</parameters>
<components>
    <component>
        <component_name>t1.c1</component_name>
        < sql_with_ref_template>
          <select><ref_template ref="t2">pl</ref_template></select>
          <from>pl in spl</from>
        </sql_with_ref_template>
    </component>
</components>
<sp_synchro>sp_link*(t1.c1)</sp_synchro>
<te_synchro>seq*(t1.c1)</te_synchro>
<duration>400</duration>
</template>
```

Figure 9. A template definition using XML syntax.

.

4.2. From Templates to SMIL

SMIL [29] is dedicated to multimedia presentations. Nevertheless, SMIL does not intend to consider database accesses. The aim of SMIL is hopefully to manage the temporal and spatial constraints related to multimedia presentations. A component position in SMIL

structure tells “when it appears” [27]. The idea is here to translate a template instance into an SMIL document and to present this document. The architecture of the use of SMIL from Templates is described in figure 10.

There are numerous implementations of SMIL viewers [27]: Soja, Grins and G2. The SMIL standard allows the definition of spatial aspects of presentation using the definition of regions. A region is a square described by an identifier, a width, a height, and the co-ordinates of its top-left corner relatively to the top-left corner of the display area. SMIL also allows temporal schedule definition using tags that indicates sequence or parallel presentations, or any composition of them. According to template needs, the generation of the spatial aspects into SMIL is straightforward: we translate the relative coordinates of the templates into absolute co-ordinates in SMIL. The synchronization constraints that manage a given number of elements used in templates have equivalents in SMIL. For constraints that are defined on set during the definition of templates, the instantiation of a template fills the set, and then we know the number of elements to schedule. For the spatio-temporal primitives related to presentations stacks, namely *stk_par** and *stk_ind**, one schedule can be provided using SMIL temporal primitives. The definition of the temporal schedule has then to take into account these parameters. The main problem that remains is that there is no way to handle the sizes of the texts displayed in regions: the text fonts are not handled, so if a text it too large to fit on the space assigned to it, it is cropped (in Grins and Soja at least). Using the “fit” attribute, it is possible to display images at a given size. The compositional aspects of the Templates cannot be translated directly into composition of SMIL documents, because even if SMIL handles such composition (using the “ref” tag), the root-layout needs absolute sizes definitions, which

is not satisfactory in our case. We must then generate one SMIL document that describes the presentation.

The table 1 summarizes the SMIL features according to the templates needs.

Document type	spatial aspect support	temporal aspect support	Composition aspect support	Collections aspect support
SMIL	YES	YES	YES	NO

Table 1: SMIL for template support.

As described in part 4, the definition of Templates is done using a simple XML DTD, and the Template manager stores the template.

The Template Manager then checks consistencies related to template composition, spatial and temporal points of view. It acts both as a repository of metadata about templates and a consistency checker in order to insure multimedia presentation, as shown in figure 2.

Figure 10 describes the different elements of the generation of a presentation based on a template call. The presentation calls (i.e. the template calls) are made in a SMIL+ document (i.e., a SMIL document containing template calls) as defined in the next section. The document is sent in a HTML form. A template call expresses the fact that we use one template on a set of data coming from a database. The Template Manager checks that the template can be used on the data of the database and then process generation of the SMIL document related. In fact, the manager uses the part dedicated to the template call and the features of the region where the template have to be displayed. Any call to templates is achieved using the O2web interface. At the object database level, we define

methods to generate objects presentations, and the template manager fuses all these presentations into a valid SMIL document. The document is then sent to the web client.

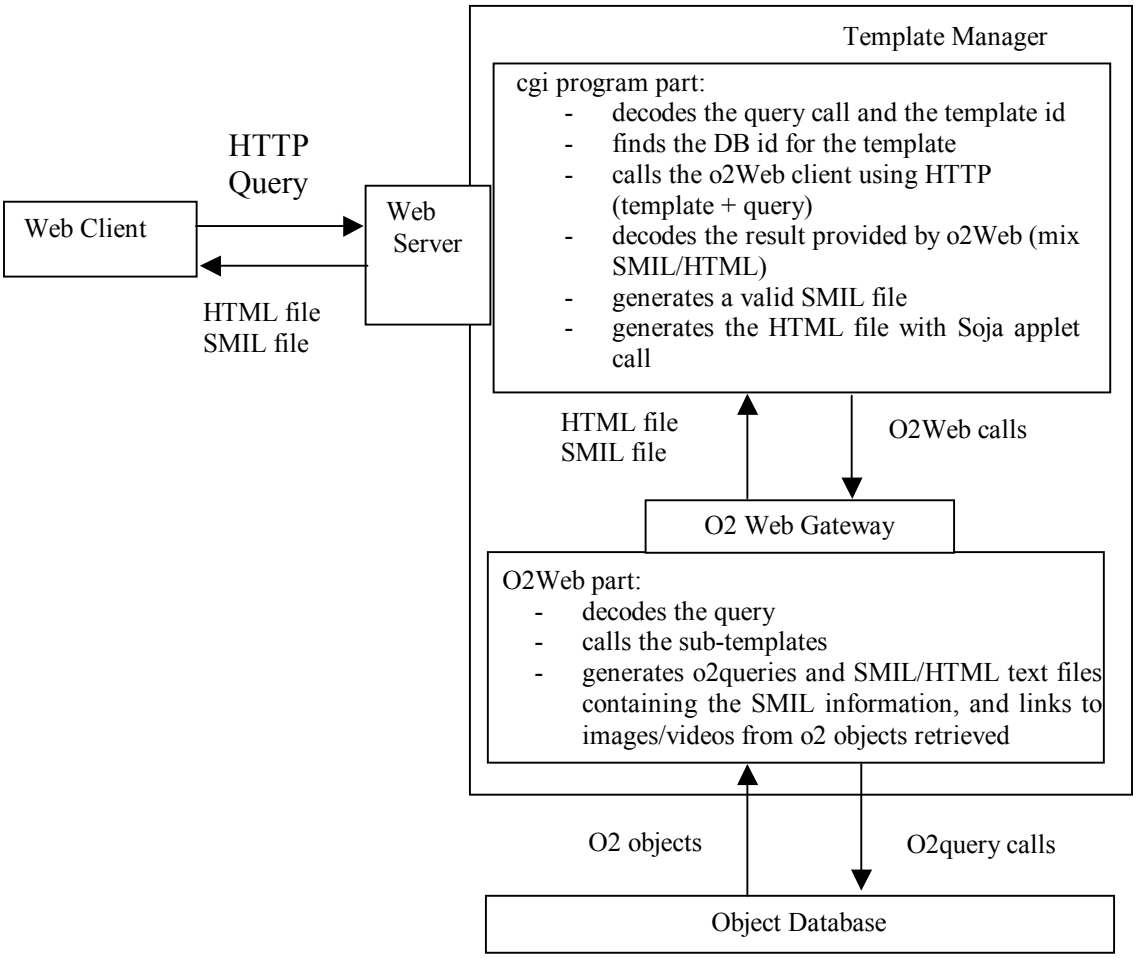


Figure 10. Template manager for SMIL documents

4.3. Template calls embedded in SMIL documents

As we explained previously, the call of templates is the event that triggers the links with a database. What we express here is a way to embed template calls into existing SMIL documents. SMIL allows the creation of multimedia documents by constraining data to spatial regions and to time regions. Our idea here is then to extend the SMIL syntax to

define new data elements corresponding to template calls. Such template calls are then related to spatial and temporal regions, as other multimedia data. The spatial constraints define the display area for a template, and the temporal constraint coming from the SMIL description may override the duration provided during the template definition. The temporal duration is then "extern" by default, meaning that the context is responsible for the definition of the duration of the template.

We present in figure 11 the part of the DTD that should be added to a SMIL DTD (to avoid confusion, we define these extended SMIL document using the tag "e_smil") in a way to integrate the template calls, and we show in figure 12 one example of call in an extended SMIL document.

```
<!ELEMENT template_call >
<!ATTLIST template_call
        id_template ID #IMPLIED
        id_smil ID #IMPLIED
        database CDATA #IMPLIED
        parameters CDATA #IMPLIED
        duration CDATA "extern">
```

Figure 11. The DTD of a SMIL embedded template call.

The call of one template, for instance the one described above on laboratories, is shown in figure 12. In figure 12, we use an OQL query to focus only on computer science laboratories. The duration of the template presentation is 300 seconds, because of the *par* tag, and the lab region defined as usual in the head of the SMIL document defines the spatial region presentation.

```
<e_smil>
 <head>
  <layout>
   <root-layout width="500" height="500" background-color="#FFFFFF" />
   <region id="lab" width="400" height="400" left="0" top="0" />
   …
  </layout>
 </head>
 <body>
  <par dur=300>
   <template_call id_template="t1"
                  id_smil="lab"
                  database="http://db.edu/database"
                   parameters="(select lab from lab in LABS
                            where domain='Computer Science')"/>
   …
  </par>
 </body>
</e_smil>
```

Figure 12. A template call example in a template extended SMIL document

5. Examples of template definitions

In this section, we introduce a practical example to show some benefits of our model. Suppose we want to present all the computer science research laboratories of a database and the people working in these labs. All the people working in a lab have to be displayed during the presentation of the information on the lab. The overall presentation must be at the most 400 seconds. Each laboratory is displayed during 30 seconds in the upper 1/3 part of the presentation, the employees are displayed four by four in the remaining 2/3 of the display with stacks synchronized by an *stk_par**, and without temporal duration constraint. For an employee, we display her/his first and last names, and her/his photography. For a laboratory, we present its name, its logo, its address and an

information field. The descriptions above are made using the original syntax and not the XML one, because the original is more compact for the purpose of the explanation.

The definition of the different templates is defined in a top-down way as:

- the main template for the presentation is then :

```
Template : t1(spl : set(laboratory));
Components : tuple(t1.c1 = select t2(pl) from pl in spl);
Sp_Synchro: Sp_link*(t1.c1);
Te_Synchro : seq*(t1.c1);
Te_Duration : 400
```

The component t1.c1 is a collection component where each element is a template presentation of a laboratory. Due to the spatial synchronization, each component element occupies the whole template presentation space. The temporal synchronization constraint indicates that the laboratories are presented in sequence. The duration 400 of template t1 has priority over the duration 30 for the template t3 (see below).

```
Template : t2(l : laboratory);
Components: tuple(t2.c1=t3(l), t2.c2=t4(l.employees));
Sp_Synchro : Sp_H2(40,60);
             Sp_Link(t2.c1, 1,1); Sp_Link(t2.c2,2,1);
Te_Synchro : par-equal2(t2.c1, t2.c2);
Te_Duration : -1
```

The template t2 is composed of a template that presents information about the laboratory (t2.c1), and information about employees (t2.c2). We assign 40% of the space area to t2.c1, and 60% to t2.c2.

```
Template : t3(l : laboratory);
Components : tuple(t3.c1 = IMAGE(l.logo),
                   t3.c2 = STRING(l.name);
                   t3.c3 = STRING(l.address),
                   t3.c4 = TEXT(l.info));
Sp_synchro :  Sp_H4();
              Sp_Link(t3.c1,1,1);
              Sp_Link(t3.c2,2,1);
              Sp_Link(t3.c3,3,1);
              Sp_Link(t3.c4,4,1);
Te_Synchro : par-equal(t3.c1, t3.c2, t3.c3, t3.c4);
Te_Duration : 30
```

The four *Link* sentences associate each piece of data with its corresponding region. The presentation of each datum must start and finish simultaneously. The duration (30 seconds) can be changed according to t1 duration.

```
Template : t4(sp : set (employee));
Components : tuple(t4.c1 = select t5(p) from p in sp);
Sp_Synchro: Sp_H2();
            Sp_V2();
            Sp_link*(t4.c1);
Te_Synchro: stk_par*(t4.c1);
Te_Duration: -1
```

t4 is an example of stack use. The employees are presented four by four in parallel (*stk_par*(t4.c1)).* Note that it is possible to query a parameter.

```
Template : t5 (p : employee);
Components : tuple( t5.c1 = IMAGE(p.photo),
                    t5.c2 = STRING(p.firstname),
                    t5.c3 = STRING(p.lastname));
Sp_synchro :  Sp_V100() ;
              Sp_H100() ;
              Sp_link(t5.c1,3..97, 5..30);
              Sp_link(t5.c2,30..70, 35..45);
              Sp_link(t5.c3, 70..95, 55..65);
Te_Synchro : par-equal(t5.c1, t5.c2, t5.c3);
Te_Duration : -1
```

To show that accurate layout can be specified, we define a “stair” layout that contains the photograph, the first and last names of the employees of each of the selected laboratories.

6. Conclusion

In this paper, we proposed an environment for a better utilization of database capabilities in the context of Web presentations. Some studies like [1] show the necessity of providing different views of database data. Not many studies focus on multimedia features [7]. In existing proposals, HTML pages are created either by directly adding text and images or by querying a database. In the latter, the presentation system is very basic and temporal and spatial properties are generally ignored. We proposed in this paper a generic spatio-temporal model for creating Web presentations. A Web presentation is defined according to different query expressions and according to spatio-temporal properties. Temporal and spatial constraints determine the behavior and the layout of each object belonging to the presentation in the presentation space and on the time line. Recent efforts such as the SMIL standard show that it is important to consider the spatio-temporal dimension in the context of Internet applications.

We used the OQL query language that is almost the standard for object database products, to create multimedia Web presentations automatically. We showed that the basic functionalities of a DBMS could provide powerful functionalities to support such presentations. Note that our proposal could be implemented using any object DBMS

supporting ODMG standard. The generic temporal model has been implemented on top of the O2 DBMS.

Some other advantages of our approach come directly from the DBMS utilization: the data is stored in the database, so updates are immediately taken into account by presentations. Our environment preserves a logical independence between data and presentations. Temporal and spatial behaviors are not imposed by the query. It is possible to define different presentations for the same object collections. Any object belonging to the presentation is just referred to by its object identifier.

Compared to related works, this paper is a seamless integration of different technologies in order to improve the construction of Web presentations. Our proposal can be considered as a contribution to the future SMIL standard by fulfilling the gap between SMIL presentations and database technology. Moreover, from a temporal point of view, we showed that it is possible to force a temporal interpretation of a multimedia database query. We also studied the related features of spatial descriptions. We propose a very efficient and expressive approach. We consider that this work is a suitable approach for many applications. We argue that the DBMS can offer more than storage capabilities.

The database part of this work (except for the spatial aspects) has been prototyped on top of the O2 object DBMS. We plan to use the XML Web tool to incorporate the HTML or SMIL pages construction in our context. Recent works on XML query languages such as

[12] and XML document storage such as [20] are related to our work. From a theoretical point of view, we continue our investigations in several directions.

Firstly, we will study the possibility of incorporating heterogeneous and distributed sources of data. In this paper, we assumed that all the data is stored in one database server. For dealing with heterogeneous support, we plan to extend MHEG specifications for dealing with generic presentations. We also plan to study advantages of an algebraic approach such as [6] in order to specify templates.

Secondly, as a perspective to this work, we aim to add user interactions in generated presentations. Another future research direction addresses the specification of user profiles and the generation of dynamic presentations according to a specific profile. The approach of templates offers also a great advantage tailoring the presentations according to network or hardware client features, by adding these features in the profile parameters. For instance, templates should be used in emerging contexts like the Wireless Application Protocol (WAP), so that it provides on the fly presentations that fit on small displays with a narrow network bandwidth. Definitively, XML is a sound basis to support more semantic multimedia models. We are in the process investigating some researches in this direction by considering XML as a low level representation instead of a data model.

7. References

1. S. Abiteboul, *On Views and XML*, Symposium on Principles of Databases Systems (PODS), Philadelphia, PA, USA 1999

2. S. Abiteboul, S. Cluet, V. Christophides, T. Milo, G. Moerkotte and J. Simeon, Querying structured documents in object databases, International Journal of Digital Libraries, Vol. 1, n. 1, April 1997, pp. 1-19.

3. A. Alasqur, OQL: A Query Language for Manipulating Object Oriented Databases, 15th VLDB Conference, Amsterdam, The Netherlands, September 1990.

4. J.F. Allen « Maintaining knowledge about temporal intervals » - CACM, vol 26, n° 11, 1983.

5. G. Arocena and A. Mendelzon, WebOQL: Restructuring Documents, Databases and Webs, Proc. of the ICDE Conference, Orlando, Florida, USA, February 1998, pp. 24-33.

6. S. Adah, M.L. Sapino, V.S. Subrahmanian, *A Multimedia Presentation Algebra*, ACM SIGMOD 1999, Philadelphia.

7. E. Bertino, E. Ferrari, M. Stolf, *A System for the Specification and Generation of Multimedia Presentations*, IEEE Transactions on Multimedia Systems, 1999.

8. C. Baral, G. Gonzalez and A. Nandigam, SQL+D: extended display capabilities for multimedia database queries, Proc. of the ACM Multimedia'98 Conference, Bristol, UK, pp.109-114.

9. D. C. Blair, An extended relational document retrieval model, Information Processing & Management, Vol. 24, n. 3, 1998.

10. R.G.G Cattell, editor, The Object Database Standard: ODMG-93, Morgan-Kaufmann, San Francisco, California, USA, 1994.

11. "Cascading Style Sheets, level 1", W3C, REC-CSS1-19990111, January 1999

12. V. Christophides, S. Cluet, J. Siméon, *On Wrapping Query Languages and Efficient XML Integration*,SIGMOD Conference, Dallas, Texas, USA, 2000.

13. A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Suciu, XML-QL: A Query Language for XML, W3C, NOTE-xml-ql-19980819, August 1998.

14. M. Egenhofer and R. Franzosa, Point_set topological spatial relations, Int. Journal on Geographical Information Systems, vol. 5, n°. 2, 1991, pp. 161-174.

15. Nael Hirzalla, Benjamin Falchuk, Ahmed Karmouch: A Temporal Model for Interactive Multimedia Scenarios. Revue IEEE MultiMedia Vol. 2, n. 1, Spring 1995.

16. W. Kim, E. Bertino and F. Rabitti, A Query Language for Object Oriented Databases, Technical Report n. ACT-OODS-337-89, Austin, USA, September 1989.

17. C. Keramane and A. Duda, « Interval Expressions - a Functional Model for Interactive Dynamic Multimedia Presentations », Proc. of IEEE International Conference on Multimedia Computing and Systems (ICMCS'96), Hiroshima, Japan, June 1996.

18. T. Lahiri, S. Abiteboul, J. Window, Ozone: Integrating Structured and SemiStructured Data, International Journal of Digital Libraries, Vol. 1, n. 1, April 1997, pp. 68-88.

19. J. Li, T. Özsu and D. Szafron, «Spatial Reasoning rules in Multimedia Management Systems», Proc. International Conference on Multimedia Modeling, Toulouse, France, November 1996, pp. 119-133.

20. H. Liefke, D. Suciu, *XMILL: An Efficient Compressor for XML Data*, SIGMOD Conference, Dallas, Texas, USA 2000.

21. M. Lino, Y.F. Day and A. Ghafoor, « An Object-Oriented Model for Spatio-Temporal Synchronization of Multimedia Information ». Int. Conf. On Multimedia Computing and Systems, Boston, USA, May 1994.

22. H. Martin, «Specification of Intentional Multimedia Presentations using an Object-Oriented Database » Proc. of the International Symposium on Digital Media Information Base, Nara - Japan , November, 1997.

23. M. Mannino, J. Choi and B. Batory, The Object Oriented Functional Data Language, IEEE Transactions on Software Engineering, Vol. 16, n. 11, 1990.

24. A. Mendelson, G. Mihaila, T. Milo, Querying the World Wide Web, Journal on Digital Libraries, Vol. 1, n. 1, pp. 54-67.

25. H. Martin and P. Mulhem, Generic Multimedia Document Presentation for the WWW using an object DBMS, 9th International Database Conference, Hong Kong, China, July 1999, pp.57-71.

26. D. Papadias and D. Sellis, «Qualitative Representation of Spatial Knowledge in Two-Dimensional Space», VLDB Journal, Special Issue on Spatial Databases, Vol. 3, n°. 4, 1994, pp. 479-516.

27. L. Rutledge, L. Hardman, J. Ossenbruggen, The use of Smil: Multimedia research currently applied on a global scale, Modeling Multimedia Information and Systems Conference, Ottawa, October 1999, pp.1-17.

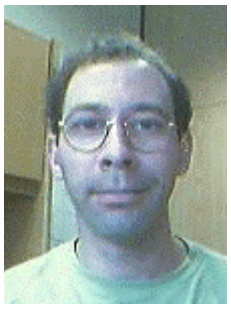
28. T. Shih and A. Chang, «Toward Generic Spatial/Temporal Computation Model for Multimedia Presentations», IEEE Intl. Conf. On Multimedia Computing and Systems; Château Laurier, Canada, June 1997.

29. Synchronized Multimedia Integration Language (SMIL) 1.0 Specification, W3C, REC-smil-19989615, June 1998.

30. Thomas D. C. Little, Arif Ghafoor, Interval Based Conceptual Models for Time-Dependent Multimedia Data, IEEE Transaction on Knowledge and Data Engineering, Vol. 5, n. 4, August 1993, pp. 551-563,

31. T. Wahl, K. Rothermel, « Representing Time in Multimedia Systems », Int. Conf. On Multimedia Computing and Systems, Boston, USA, May 1994.

32. Extensible Markup Language (XML) 1.0, W3C, REC-xml-19980210, February 1998.



Philippe Mulhem is currently Director of the Image Processing and Applications Laboratory (IPAL) located in Singapore, joint laboratory between the French Centre National de la Recherche Scientifique, the National University of Singapore and the Kent Ridge Digital Laboratories. He is also a member of CLIPS-IMAG laboratory of Grenoble. His works focus on formalization and experimentation of image and video indexing and retrieval, as well as on multimedia documents indexing and retrieval. He obtained his PhD from the Joseph Fourier University, Grenoble, France, in 1993.

Hervé Martin is an assistant professor of computer science at the University of Grenoble and a senior researcher at LSR-IMAG laboratory in the SIGMA group. His main topic of current research is Multimedia Web Information Systems. In this context he is leading a project in which experimentation platform for generating Web-Information System is being developed.