XML-Based Automatic Web Presentation Generation

Flavius Frasincar, Geert-Jan Houben
Department of Computer Science
Eindhoven University of Technology
Eindhoven, the Netherlands
{flaviusf, houben}@win.tue.nl

Abstract: This paper presents a method that automates hypermedia presentation generation on the Web. The method is based on RMM (Relationship Management Methodology) for aspects of hypermedia design. It distinguishes between the logical representation of the data and its actual presentation. Using new emerging Web technologies like XML (Extensible Markup Language) and XSLT (XML Stylesheet Language Transformation), we have implemented a prototype to experiment with the proposed method. Data filters written in XSLT prove to facilitate the multi-phase hypermedia presentation generation method.

Introduction

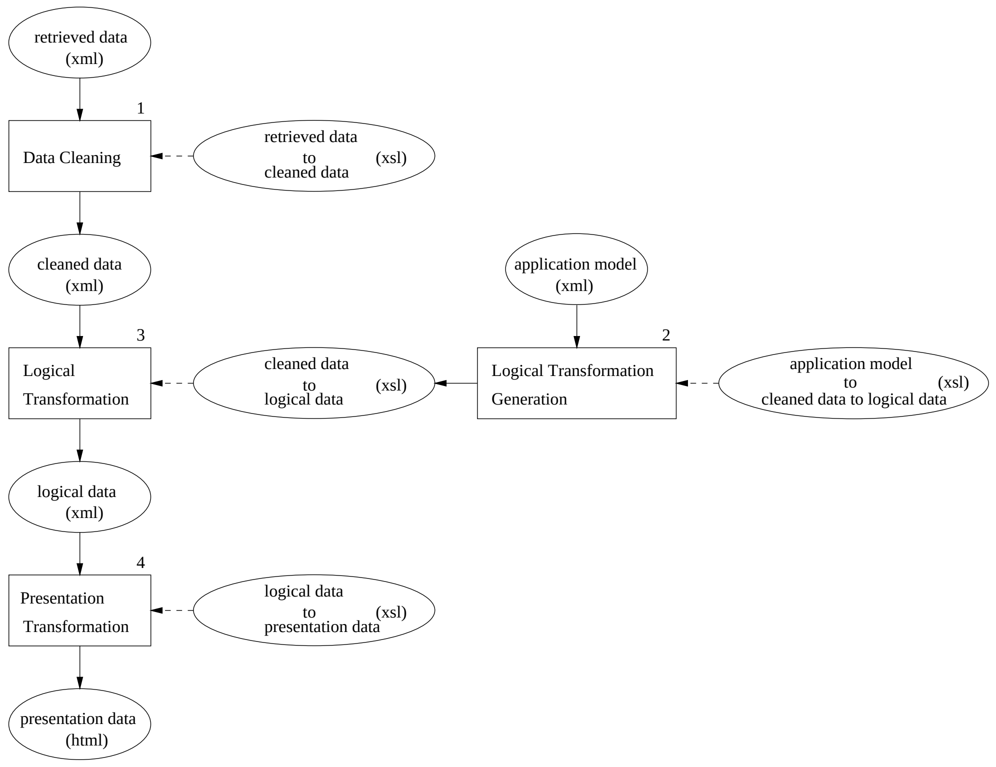
World Wide Web (WWW) is the most important platform for information exchange between Internet users. As a result there is an increasing need for methodologies that support the design of Web-based Information Systems (WIS) (Isakowitz et al., 1998a). We consider WIS that integrate heterogeneous data sources, such as XML repositories, relational and object-oriented data bases etc. A user asks a query to such a system and as a result a presentation is generated (De Bra & Houben, 2000). The main focus of this paper is the question how to automate the process of generating such hypermedia presentations.

Several hypermedia design methodologies are available, like RMM (Relationship Management Methodology) (Isakowitz et al., 1995) and OOHDM (Object Oriented Hypermedia Design Methodology) (Schwabe et al., 1996). The design method that we use here applies the core of RMM for reasons of its simplicity and its E-R (Entity-Relationship) foundation. Extending the well-accepted E-R model to model information domains, and subsequently adding navigation structures to it, proves to be a solid ground on which to base the automatic generation of hypermedia presentations.

Before we discuss the prototype implementation, we shortly mention the key concept (both in RMM and our specific design method) of 'slice', which is used to denote meaningful presentation units. A slice groups together attributes and possibly other slices. In order to have a uniform approach to slice hierarchies (Isakowitz et al., 1998b) we consider primitive slices to be attributes. Each slice belongs to an entity but it can contain also slices belonging to different entities: in this case, the relationships between entities which make such an embedding possible are indicated. If the relationships are one-many an access structure (index, tour, and indexed guided tour) is associated to the slice nesting. Many-many relationships (from the E-R model) are decomposed in two one-many relationships. There are two types of slice relationships: aggregation (presented above) and reference (hyperlinks between slices). For a particular application, the application model describes all slices and their relationships (Isakowitz et al., 1998b), thus specifying the hypermedia aspects of the application. Due to the fact that each slice is owned by an entity and encapsulates some of the entity's attributes we can view the application model as an extension of the E-R model, and thus as an important step in the design process of the hypermedia presentation.

Method and Tools

The presentation generation method is based on four steps which are depicted in (Fig. 1). In the first step, Data Cleaning, the data retrieved from the Data Retrieval module is adjusted to the specific format for E-R model instances. For each application, there is an application model which describes the slice relationship model on top of the considered domain model. In the next step, Logical Transformation Generation, a transformation engine is produced based on the specific application model. This transformation engine is used by the following step, Logical Transformation, to package the retrieved data instances into slices. The final step, Presentation Transformation, generates a presentation, e.g. in HTML, from the slice packaged data.



retrieved data (xml)
1
Data Cleaning
retrieved data to cleaned data (xsl)
cleaned data (xml)
application model (xml)
3
2
Logical Transformation
cleaned data to logical data (xsl)
Logical Transformation Generation
application model to cleaned data to logical data (xsl)
logical data (xml)
4
Presentation Transformation
logical data to presentation data (xsl)
presentation data (html)

Figure 1: Method

The artifact of each step is a valid XML (Extensible Markup Language) (W3C XML Working Group, 2000, Bradley, 2000) file, that is a file that complies to a prescribed DTD (Document Type Definition) file. For the Logical Transformation Generation step an XSL (XML Stylesheet Language) (W3C XSL Working Group, 2000, Bradley, 2000) file is produced which is also an XML document. At the core of each step there is an XSL file which describes to an XSLT (XSL Transformation) processor how to convert the input XML file to the desired output XML file.

In order to experiment with the proposed method a demo was developed. The domain model of this application is based on the data provided by the Web site of the Rijksmuseum in Amsterdam. The prototype uses annotated Rembrandt paintings to exemplify the clair-obscur painting technique. As a software tool, the XSLT processor Xalan, provided by the Apache XML Project (Xalan Apache XML Project, 2000), is used. The next sections illustrate the different steps of the method for this demo application. Due to space limitations we can only show small excerpts of the software.

Data Retrieval

The demo considers one particular query, a restriction that doesn't affect the purpose of the presented method to show how to dynamically generate hypermedia presentations on the Web. The user asks a query to the system, a query which is expressed (in the current implementation) in SQL. The results of the query are encapsulated in an XML file which has three components: entity instances, attribute instances, and relationship instances. (Example 1) presents a small excerpt from the DTD used to describe the retrieved data and (Example 2) provides a piece of the retrieved data as an XML fragment.

Example 1 (Data.dtd)

```
<!ELEMENT attribute-instance (#PCDATA)>
<!ATTLIST attribute-instance attribute-id CDATA #REQUIRED>
```

Example 2 (Data.xml)

```
<attribute-instance attribute-id="attribute.technique.name">
<![CDATA[clair-obscur]]>
</attribute-instance>
```

Data Cleaning

The Data Cleaning step bridges the gap from the XML data representing the SQL output to XML data representing an E-R model instance. The transformation stylesheet of this step captures the domain knowledge to fill the missing data. Relationship names are added to the retrieved data and the inverse of the relationship instances is built (since they were not originally included in the data retrieved). (Example 3) gives a flavor of the XSL file that specifies the above transformations.

Example 3 (DataCleaning.xsl)

```
<xsl:when test="@relationship-id='painting-technique'">
    <xsl:attribute name="relationship-id">relationship.exemplifies</xsl:attribute>
</xsl:when>
```

Application Model

The application model is used to describe at the logical level the hypermedia aspects of the application (Isakowitz et al., 1998b). At this logical level, the slice types are identified. By slice type we mean a full specification of the elements (slices and/or attributes) contained in the slice, and reference relationships to other slices. In the introduction section we explained that the application model is an extension of an E-R model. (Fig. 2) presents the E-R model of the demo application. Note that we model pictures as URLs (Uniform Resource Locator), so all the attributes are of type text (string).

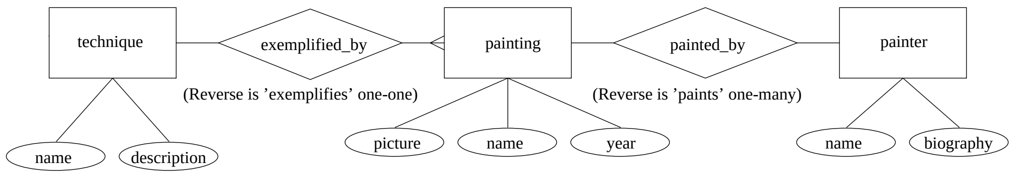


Figure 2: E-R model

(Fig. 3) gives the application model design for the application. There are two (non-primitive) slices, both of them called *main* that belong to two different entities, *technique* and *painting* respectively. The starting point of the presentation, that is *technique.main* (the '.' denotes the ownership relation between slices and entities), is indicated in bold font. *technique.main* encapsulates the *name* and *description* attributes from *technique* and an index structure of hyperlinks. Each of these hyperlinks has as an anchor the picture's *name* attribute and as target the slice *painting.main*. The slice *painting.main* is composed of three attributes of painting: *picture*, *name*, and *year*, and the attribute *name* of the associated painter. The entity relationships *exemplified_by* and *painted_by* are used to specify (at instance level) which attributes (instances) belonging to different entities are (logically) grouped together in the presentation.

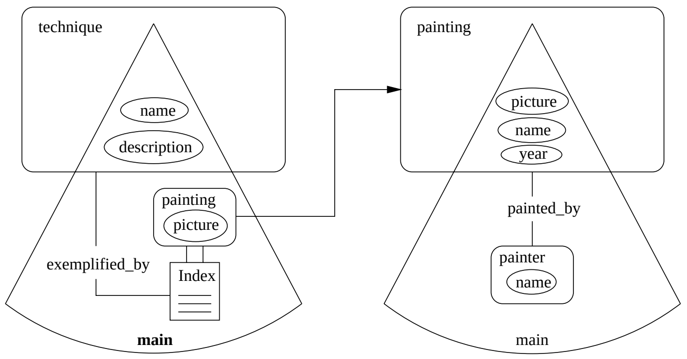


Figure 3: Application model

We encode slices as XML elements that can contain text (the so-called empty slice) (Isakowitz et al., 1998b), slice references, hyperlinks, index, and guided tours (Example 4). (Example 5) presents a small piece of the application model which illustrates the use of slice references in slice definitions.

Example 4 (Application.dtd)

```
<!ELEMENT slice (text | (slice-ref|hyperlink|index|guided-tour)*)>
<!ATTLIST slice id ID #REQUIRED>
```

Example 5 (Application.xml)

```
<slice id="slice.painting.main">
    <slice-ref idref="attribute.painter.name"
               relationship-ref="relationship.painted_by"/>
...
</slice>
```

Logical Transformation Generation

The Logical Transformation Generation step is responsible for building the main presentation transformation engine, that is the engine that packages the retrieved data instances into slices. It is implemented as an XSL stylesheet that generates another stylesheet (Lemmens & Houben, 2001). The input of this step is the application model encoded as explained in the previous section in an XML file. Knowing the type description of each slice, a stylesheet is generated that will transform data at instance level. An excerpt of the stylesheet implemented for this step is presented in (Example 6).

Example 6 (LogicalTransformationGeneration.xsl)

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:axsl="http://www.w3.org/1999/XSL/TransformAlias"
                version="1.0">
<xsl:template match="/">
    <axsl:stylesheet version="1.0">
    <axsl:template match="/">
    <xsl:element name="results">
        ...
        <xsl:apply-templates select="//slice-ref" mode="ROOT"/>
    </xsl:element>
    </axsl:template>
    </axsl:stylesheet>
</xsl:template>
...
</xsl:stylesheet>
```

One can observe that there are two namespaces defined for stylesheets, one for the current one and one for the output. For each slice reference a template is called in the mode *ROOT*, which generates an element of type *slice-instance* at the top level. Once a slice instance is encoded each reference to another slice (in its body) is replaced by a *slice-instance-ref* which points to one of the top level slice instances. The main slice type (the starting point of the presentation) is populated by all the instances of the entity that owns that slice. In case that a relationship is involved, only those entity instances that are related to the current entity instance via a relationship are considered. For attributes or slices that belong to the current entity instance the population is obvious, as the current entity instance is used.

Logical Transformation

The Logical Transformation step packages the data instances into slices (it provides instances of slice types based on the retrieved data). The stylesheet that performs this transformation was automatically generated by the previous step. The input of this step is an XML file containing the cleaned data, and the output is another XML file that describes the grouping of the input data into slices. (Example 7) illustrates how to retrieve attribute values from the input data.

Example 7 (LogicalTransformation.xsl)

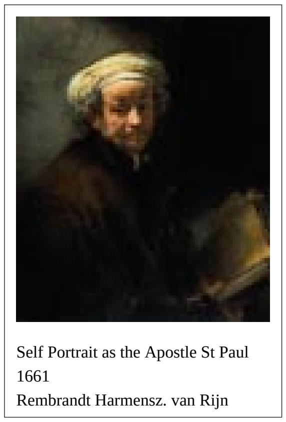
```
<text>
    <axsl:value-of select="attribute-instance[@attribute-id='attribute.painting.name']"/>
</text>
```

Presentation Transformation

The Presentation Transformation step builds a Web presentation file. We used HTML for the code generator, but it can be any other hypermedia format supported by the Web. The presentation code generator is based on a stylesheet that takes as input the slice packaged data (the output of the previous step) and converts it to a HTML file, ready to be presented on a Web browser (like Internet Explorer or Netscape Navigator). (Example 8) indicates that for each *slice-instance* an HTML table is generated. For each *slice-instance-ref* a row is built in the table. (Fig. 4) illustrates the HTML representation of the *painting.main* slice instantiated with a painting by Rembrandt.

Example 8 (PresentationTransformation.xsl)

```
<xsl:template match="slice-instance">
    <TABLE>
        <xsl:apply-templates select="*"/>
    </TABLE>
</xsl:template>
```



Self Portrait as the Apostle St Paul
1661
Rembrandt Harmensz. van Rijn

Figure 4: HTML presentation

In collaboration with CWI, Amsterdam, an interface between the output of Logical Transformation and Cuypers presentation system (van Ossenbruggen et al., 2001) will enable the generation of a different, more customized presentation in SMIL (Synchronized Multimedia Integration Language) format.

Hera Architecture

The experiences with this specific prototype are exploited in the Hera project (Houben, 2000), a research project that investigates hypermedia systems able to automatically generate presentations from ad hoc queries on heterogeneous data sources. Based on the principle of separation of concerns we distinguish two components in the Hera system: data retrieval and data presentation. Data retrieval is responsible for the integration (wrapping and mediation) of the different data input sources. It processes the input query and gathers the retrieved data. Data presentation builds a logical view of the data to be presented (it includes all the steps presented in this paper except for the last one) and outputs the (final) presentation (the last step of the proposed method).

Conclusions

In this paper we have considered an automatic presentation generation method based on four steps, and we have illustrated experiences from a prototype based on this method. Throughout the entire process implemented in the prototype the data is encoded in XML which proves to be an ideal format to store this structured data. We have shown how this XML data is managed in the generation process: for each step a data filter is written in XSLT, a standard supported by W3C for XML transformations. These data filters prove to fit well in our multi-phase presentation generation.

In the future we will extend this prototype (and apply it to different applications) in order to investigate: the extension of the system to multiple queries (based on slices), the impact of having different attributes types (besides text) on the presentation generation, different back-ends for the final presentation, and slices-on-demand (slices that are provided on demand, from a query, by a servlet).

References

Bradley, N. (2000). *The XML companion*. Addison Wesley.

De Bra, P. & Houben, G.J. (2000). Automatic Hypermedia Generation for Ad Hoc Queries on Semi-Structured Data. *ACM Digital Libraries*, 240-241.

Houben, G.J. (2000). Hera: Automatically Generating Hypermedia Front-Ends for Ad Hoc Data from Heterogeneous and Legacy Information Systems. *Engineering Federated Information Systems*, Aka and IOS Press, 81-88.

Isakowitz, T., Stohr, E., & Balasubramanian, P. (1995). RMM: A Methodology for Structured Hypermedia Design. *Communications of the ACM*, 38(8), 34-44.

Isakowitz, T., Bieber, M., & Vitali, F. (1998a). Web Information Systems. *Communications of the ACM*, 41(7), 78-80.

Isakowitz, T., Kamis, A., & Koufaris, M. (1998b). The Extended RMM Methodology for Web Publishing. Working paper IS-98-18. Available online at `http://rmm-java.stern.nyu.edu/rmm/papers/RMM-Extended.pdf`.

Lemmens, P. & Houben, G.J. (2001). XML to XML through XML. *AACE WebNet*.

Schwabe, D., Rossi G., & Barbosa S.D.J. (1996). Systematic Hypermedia Application Design with OOHDM. *ACM Hypertext*, 116-128.

van Ossenbruggen, J., Geurts, J., Cornelissen, F., Rutledge, L., & Hardman, L. (2001). Towards Second and Third Generation Web-Based Multimedia. *WWW10*, 479-488.

Xalan Apache XML Project (2000). Xalan-Java version 1.2.2. Available online at `http://xml.apache.org/xalan`.

W3C XML Working Group (2000). Extensible Markup Language (XML) 1.0 (Second Edition). W3C. Available online at `http://www.w3.org/TR/2000/REC-xml-20001006`.

W3C XSL Working Group (2000). XSL Transformations (XSLT) Version 1.1. W3C. Available online at `http://www.w3.org/TR/xslt11`.