A FRAMEWORK FOR THE DESIGN AND IMPLEMENTATION OF WEB SITES

Carlos Castillo
University of Chile / Newtenberg Digital Publishing Ltd.
2120 Blanco Encalada / 10 Estado 3rd floor
ccastill@dcc.uchile.cl

ABSTRACT

We present a framework for modeling and generating web sites that effectively separates content from representation. This model includes generic documents, called "eidox" and a multiverse of views over the documents that can be used to generate a consistent navigation in a web site. We also introduce a content management and digital publishing software suite that currently implements this model.

KEYWORDS:

Content management, XML.

1.INTRODUCTION

At the beginning, most web sites were created manually, page by page, starting with a single home page and then handcrafting links between the home page and the section page, then between all pages and, for instance, a feedback form, and so on.

Enthusiastic people that were interested in “getting their organizations into the net” created these first websites. These efforts were well intentioned but, in general, authoring hypertext is a difficult and time-consuming task, and most sites could not support adding more information very well. At that time, most pages contained some sort of “under construction” icon; but at some point, organizations began to pour more effort (and money) into their websites, with hopes of bringing more customers, keeping the current ones, or improving their internal processes as advocates of intranets proposed.

This process of professionalization of website development started with phrases like “make web pages dynamic” and later “separate content from presentation”.

These two ways of presenting the problem of developing large-scale web sites have some important differences. While the first one emphasizes the use of some programming language to encode an application logic, the second one is naturally expressed as page templates that merge with data, in a similar way to that of document templates used for mail merging against a database of addresses.

Nowadays, most websites that are not hand-crafted are made with programming languages and attempt to ``make web pages dynamic''' (on the early days of the web, shellscript was used to generate server-side includes; now typical programming languages used for that are PHP [php], ASP [asp], Perl [perl], Python [python], ColdFusion [coldfusion], W3-MSQL, etc.). But most of the time, presentation is not separated effectively from application logic. It is usual to use these mixes of SQL [Date 1993] plus HTML [html]. The programmer is thinking of database schemes, hypertext layout and application logic at the same time. In other cases, there is some degree of separation of concerns, with the creation of objects that reside in the server and are accessed by the server scripts.

In this paper, we introduce a way of describing a website content, navigation, layout and presentation in a high-level way, based on XML [xml] and XSLT [xslt]; this description can be made based on smaller independent units, and can be mapped to an implementation very easily. We also explain how content management and digital publishing can be done with the help of general properties and how this can lead to web sites in which users browse according to their own distinctions.

This model has been developed by **Newtenberg Digital Publishing Ltd**. [newtenberg] and implemented in a content management and digital publishing software suite called **Newtenberg Engine**. This has been used for about two years in the production of large web sites.

The rest of this article is organized as follows: Section 2 discuss previous work on this subject. Sections 3,4,5 and 6 present the main elements of the model: Eidox, Properties, Boxes and Layouts. Section 7 outlines the work methodology for developing a website based on this model. Some important aspects of our implementation are shown in Section 8. Section 9 presents the summary and main conclusions.

2.PREVIOUS WORK

In relation to enriching documents with information for different contexts, the idea that the next step is to build a Semantic Web [Berners 2001] of relationships has been gaining wide acceptation, and has pushed the birth of (meta)languages to express these relationships as RDF[rdf], SHOE [Hefling 1999] and Ontobroker [Decker 1998]. An interesting work about how this semantic web can be used to build hypermedia documents can be found in [Carr 2001].

Our work is related with that of “placeless documents” [Dourish 1999], in which the user desktop, with its files and folders, is replaced with a view of collections induced by the properties of the documents.

The idea that hypermedia patterns [Garrido 1997] should be used in the same way of object oriented design patterns [Gamma 1995], has been gaining wide acceptation, although the hypermedia pattern catalog [hdpr] has been growing in an rather inorganic way because it is not so clear what indeed constitutes a pattern and what does not, or what types of patterns exist. A complete survey about hypermedia design patterns can be found in [German 2000]. Most patterns (such as "set based navigation" and others) can be expressed as combinations of selection, representation and style rules in our framework.

We use the concept of Multiverse envisioned by Humberto Maturana [Maturana 1988] (that multiple views over the world can and should coexist) and the idea of multiple views or multiform visualization [Roberts 2000] used commonly in the area of visualization. The idea of multiple views arises initially to be applied to the user interface and is later extended to collaborative applications [Wood 1997] in which several users interact simultaneously.

A series of works by the team of Gustavo Rossi (specially [Schwabe 1998]), show a methodology called Object-Oriented Hypertext Design Model (OOHDM) that comprises three stages: conceptual model, navigational model and interface model. OOHDM is based on object-oriented design, that is used at all stages, specially the conceptual modeling stage; then, relationships between objects are used to create links in the navigational model. Navigational objects are views over conceptual objects; in a similar way, in our model a “box” is a way of describing a view over one or more “eidox”.

Another language that describes how a Web site can be designed using object orientation is the Web Composition Markup Language [Gaetke 2000]; in WCML, there is a good separation between the content (the information component, what we call an "eidox"), the representation of the content (the decorator component) and the selection logic (the data source and the query of the factory component). In WCML, references between components are explicit, while we try to exploit the ontologies to create links.

Also WebML [Ceri 2000] describes an approach to the process of website creation, that emphasizes the aspects related with formally defining every page and zone into the page. WebML defines several types of zone as: data unit, index, filter, scroll, etc; WebML also defines relationship between these zones and other pages, and how “entities” are presented in each of the zones. We think that our framework is more general, because we assemble a page with a homogeneous set of filters, and their functionality is not intrinsic, but corresponds to the distinction that an observer makes at watching the page as a whole and the relationships between its zones. Also, in our case the navigational model arises naturally from the conceptual model and it is closer to the attribute-based object than a classical object model.

There are similarities between these works and the present one, and there are also novel ideas in each of them that contribute to enhance modularity, improve the consistency and accelerate the development of Web sites.

3.EIDOX

An **Eidox** is the basic unit of content. The name comes from the Latin **Éidos** that means idea or shape, and X for XML. An eidox is an abstraction of a document that includes the content, its properties and the generative history of the content.

The **content** of the eidox can be anything that complies with a known DTD (Document Type Definition). This content is not written for a specific medium or representation, but instead it is written for all the possible representations. For instance, if the eidox represents a newspaper article, it can contain a title, epigraph, lead and text, but also a single-line content for sending to mobile phones, an abstract for using on PDAs, etc.

The generative **history** of the document is its history of changes: who edited the document and what parts they changed. It is common in collaborative editing to have some sort of versioning control, as in WebDAV [webdav].

Finally, the properties are pairs *(name,value)* attached to the eidox. They will be explained in the next section.

4.PROPERTIES AND MULTIVERSE

We see web sites as technological artifacts that communicate humans; they are a substratum, an infrastructure, for asynchronous and distant interaction. This implies that a web site must account for differences among its users, specially those that arise in the main form of interaction - browsing. The web site should allow the user to navigate according to its own distinctions, and this is hard to achieve, because the information architect usually chooses one main classification scheme that may be the most natural for some users but usually is not the most natural for some group of users.

We have observed that most of the application logic of current websites deals with what we call the selection logic. This is: how we can express which eidox we want to retrieve to show in a particular web page. This fact explains why SQL queries are so widely used in web development.

The selection logic is very important for all actors involved in the web site development: for the information architect that defines content units and navigational patterns, for the programmer that implements the logic, and specially for the user that will browse the web site. We intend to carry selection logic to something more abstract, more high-level, through the use of general **properties**.

A property is a pair *(name,value)*. A property is an assertion that a person makes about some document [rdf] that tells something about the document (a piece of meta information).

Each property can take different values. These values can have a specific type, or can be chosen from a fixed set, from a hierarchical set or be completely free-form. This is shown in the sample eidox on Figure 1.

Figure 1: Properties attached to the document can be anything.

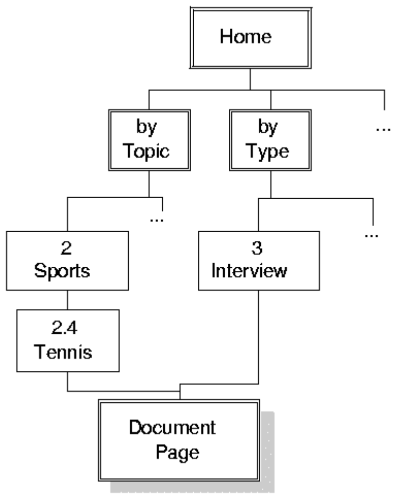
```
<eidox>
<properties>
 <property name="topic">
   <property-value value="2.4">Sports</property-value>
 </property>
<property name="type"><property-value>Interview</property-value></property>
 <property name="person"><property-value>John S.</property-value></property>
</properties>
<content>... content of the document ...</content>
<history>... history of changes of the content ...</history>
</eidox>
```

Properties are useful for a variety of reasons:

- They do not need a scheme to work (as a database scheme), but they can use it (i.e.: property values can be typed, or property values can have sub-values). As the classification of documents can have many changes, flexibility is needed.
- Each property value induces a collection and a navigation over the documents. If property values can have sub-values, the navigation can be hierarchical and multi-leveled. This generalizes the idea of folders.
- The information architect does not have to opt for a particular classification scheme, and can let the readers retrieve documents according to their own distinctions.

The later is a natural implementation of the notion of **Multiverse** [Maturana 88], that is, the information is not only classifiable or surfable from one single approach (universe), but it may coexist and complement with diverse approaches to the same content. This way, users can browse based on their own distinctions, as shown in Figure 2, where some users can browse by topic, other by type, etc.

Figure 2: A multiverse navigation can improve browsing, allowing them to browse based on different properties.



5.SELECTION, REPRESENTATION AND STYLE

In the following, when we say web page we will be refering to an XHTML document, but also to a WML card deck [wapforum], or to another XML document, or to a plain text document. In general, we mean every document that can be generated using a XSLt transformation engine.

For web page layout, we divide each web page into zones. Every zone in the page is a **box**. This is common in many hypertext generators.

A box does not always have static content: it can have dynamic content. This is generated in a series of steps, similar to the ones in the cocoon model [cocoon]. To create a box, the site designer must define its Selection Rule, Representation Rule and Style Rule as explained below:

The **Selection Rule** indicates which eidoxes will be in the box. The selection rule is expressed as a query over a set of property values. Optionally, another property that acts as a sort criterion and/or a number that tells how many documents to show at maximum can be used.

The selection rule language itself is not important for the model; the selection rule can be expressed in any language with, for instance, full text search capabilities or document content structure queries, or mapped to SQL.

The **Representation Rule** indicates which parts of the eidox will be shown, and how. The representation rule is an XSLT fragment that, when interpreted by an XSL transformation engine, chooses structural parts of the eidox and transforms them into a web page. As the eidox has the content, properties and history of a document, the representation rule has access to all of them.

Typical examples of representation rules for XHTML include itemized lists of titles, vertical and horizontal tabular menus, pull-down menus, among a large variety.

Finally, a **Style Rule** indicates what formatting rules apply. In the case of XHTML, the style rule is simply a set of CSS properties. In other cases, the style can be XSL formatting objects (XSL:fo), or any other abstraction of the visual properties of the rendered box.

These three parameters allow for a wide range of flexibility for the web designer.

6.LAYOUTS

The layout of a page is specified as an XSL document that specifies the relative position of boxes in a page.

A layout has the potential to generate many instances of pages. For example, a layout for a newspaper article in a news site will probably be the same for all the articles of today's edition. This is why layouts receive **arguments**. In the particular case of the layout for article pages, the argument received is the identifier of the article that will be shown.

In the case of section covers, as sections are modeled as property values, the argument received is the corresponding property value. Boxes can use the arguments that pages receive for the selection rule; for instance, the following selection rule selects all the documents that have the same type as the parameter given, and arrange them by section:

```
Properties: type = PARAMETER
Order: section
```

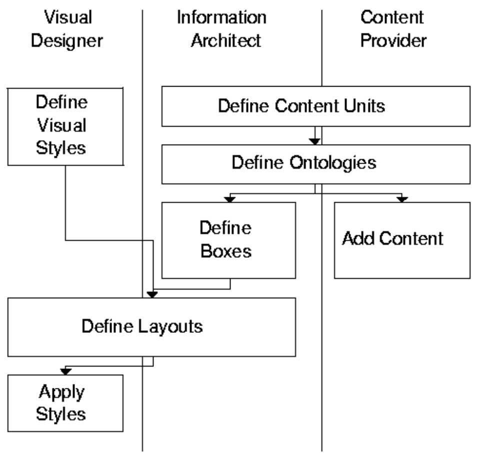
In this way, selection rules can be more flexible as the same box can generate multiple different instances depending on the context of the layout it is inserted in.

It is important to notice that a layout has neither a specific functionality into the web site navigation nor a fixed depth into the navigation hierarchy. We will say, for instance, that a layout represents a section cover, because as observers we can see that it has a dominant box that shows an index of articles that share the same topic. The functionality of a page arises from the boxes in it.

7.WORK METHODOLOGY

Figure 3 presents the main tasks of the proposed work methodology. One branch of work, the main responsibility of the information architect, is to define content units (eidox) and define the ontologies or properties. This is necessary to add, remove, and update content. The ontologies can be refined over time.

Figure 3: Main tasks for the design of a web site.



Defining ontologies is not an easy task, but it is simplified by the fact that the architect can create as many ontologies as he wants, and does not have to opt for a particular one or give it more importance than the others. The relative importance of ontologies is something that will naturally arise later from box construction.

Boxes are built once the main ontologies are defined. We have noticed that a typical web site has no more than 20 different boxes including content and navigational aids. Once defined, boxes are arranged in the layouts.

Visual styles (fonts, colors, sizes, alignment, etc.) can be developed independently, and are applied later to boxes. A typical web site has about six visual styles that are used by boxes, with only two or three really different ones and the others, variations of the main styles.

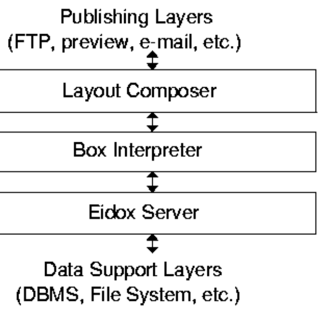
8.IMPLEMENTATION

We have developed a content management system called *Newtenberg Engine*. This falls into the category of "Model-Driven web generators" [Fraternalli 1999], in the sense that it provides a way of automating hypertext generation based on a specific model.

Generating and Publishing Web pages

The modules involved in generating a web page are the ones in Figure 4.

Figure 4: The main modules involved in going from the raw data to web pages



The bottom level is the data support layer, in our implementation, a database management system that stores all the properties and a file system for the content of the eidox. These are visible only to the **Eidox Server**, that receives a request (in the form of a selection rule) and dispatches the corresponding eidox(es) as XML to the Box Interpreter.

The **Box Interpreter** takes one box at a time, requests the eidoxes to the Eidox Server and receives a large sheet of all of the eidox concatenated. Then it uses the representation rule to transform this sheet into XHTML, WML, or plaintext, as needed. Finally, the Box Interpreter applies the style rules and passes the resulting fragment to the Layout Composer.

The **Layout Composer** just glues the resulting fragment to a complete web page or WML deck and sends it to the corresponding publishing layers. These can send the page via FTP, or e-mail, or WAP, as needed.

The publishing layer also can check if any representation rule generated a link to another page and self-invocate again to recursively build new pages.

Adding and Updating Content

The process of adding and updating content by the content providers is done via a web interface to the Eidox Server. The web interface can be changed by an e-mail interface or a direct XML interface (for composing pages offline using an XML editor).

The web interface has two main parts: the content editor and the properties editor.

In the **content editor**, some input boxes are presented to the user, according to the chosen DTD. The user can type commands in the boxes, that are later translated into XML; also we have embedded an HTML editing component into the Explorer browser for WYSIWYG editing in which XML structural elements are mapped into visual elements for the editor.

In the **properties editor**, some choices are presented based on the configuration of properties. The choices are property values, or input boxes in the case of free-form properties.

Defining Navigation and Boxes

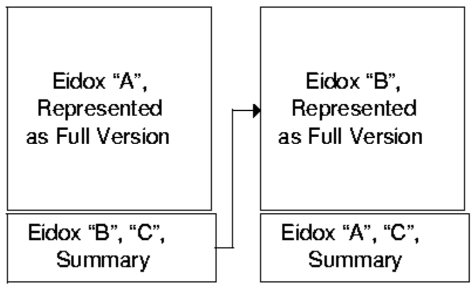
Linking appears when one layout has a box that links to another page, with possibly some arguments. For that to happen, the box representation rule must include a link in its output (for instance, an `href` in HTML). These links define the navigation on the resulting web site.

The links can be generated automatically or by hand.

Automatic linking is ontology-driven and can occur if and only if two eidoxes share at least one common property value. In that case, the representation rules for one box in the page in which the first eidox is being displayed can generate a direct link to a page in which the second eidox is displayed, or an indirect link to a page that has a box that lists all eidoxes with the same property value.

In Figure 5, the layouts of two simple pages are depicted. Each layout has two boxes, the top one has a representation rule that shows a full version of the content of an eidox, while the bottom one has a selection rule that selects documents with some property value in common with the one shown in the top box (such as the same date, or section) and a representation rule that shows a summary version of the content of an eidox with a link to the another page.

Figure 5: Two pages with an automatic link.


Eidox "A", Represented as Full Version
Eidox "B", "C", Summary
Eidox "B", Represented as Full Version
Eidox "A", "C", Summary

More complex layouts, with many boxes, can generate more complex links; also, links can be bi-directional if the destination page has a box that selects articles as the first one does.

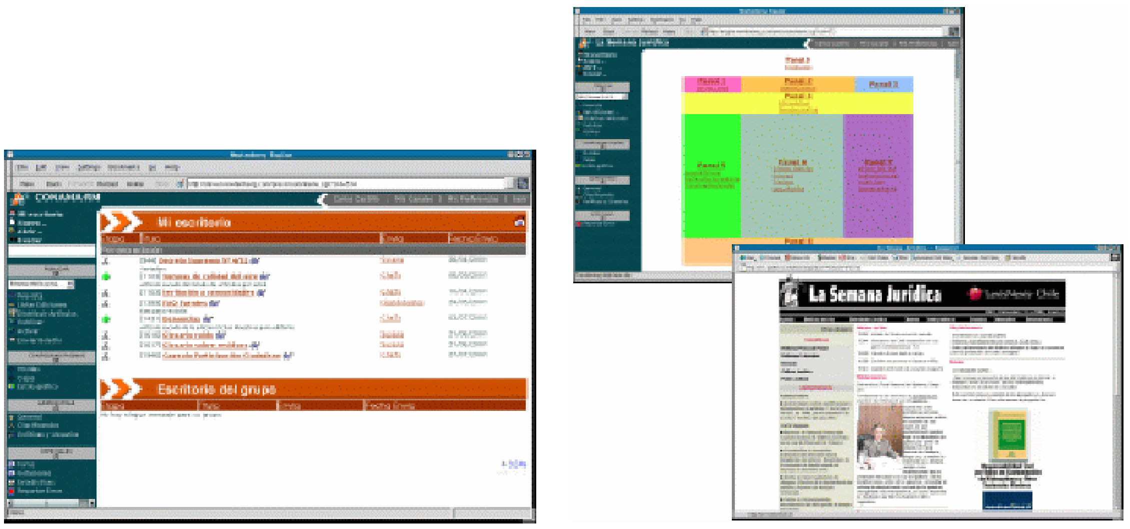
The anchor of the link can be chosen in the representation rule of the source of the link. Also, a table of contents (more generally, a set of tables of contents for this multiverse) can be generated applying a representation rule to the set of all property-values of the eidoxes; this means that the selection rule has to be able to express selection of eidoxes as well as of properties.

Manual linking can be done while editing. If the DTD for the content of the eidox supports linking, the interface can help the user write an arbitrary reference between one eidox and another.

Typical Usage

An editor or content provider logs in with a username and password and is presented with his desk with all the pending tasks (usually documents to edit, or proofread, or approve), as shown in Figure 6a.

Figure 6: a) The desktop presents all pending tasks; b) Layout editor and final result.



The editor can review, modify and approve or send a document to another person or to a group of persons (a role), as defined in the workflow. The software has a tool to define and edit the workflow.

On the other side, the administrative interface allows users to define the boxes, layout, styles and content though a graphical interface. For instance, the layout editor and the resulting page are shown in Figure 6b.

9.CONCLUSIONS

We have presented a framework that allows for quick modeling of a web site based on boxes. A box is defined by its selection, representation and style rules and it is the basic unit that the information architect will use to build pages in the web site. These boxes are easy to understand by the graphic designer and by the content providers.

The proposed model supports different viewpoints without forcing the architect to opt for a particular one. This allows the creation of web site that users can navigate based on their own views.

The web development process is modeled in a natural way. First content units (eidox) are determined, then the ontologies are created and the content providers can start adding content. Visual style is developed independently. This way, every actor can focus on his task. We have noticed that this leads to great improvements in the final result in terms of content organization and page usability.

For more information and an extended version of this article, visit [newtenberg].

ACKNOWLEDGEMENTS

We appreciate the contribution of the entire staff of engineers and journalists at Newtenberg Publishing Ltd, to Jose Flores from Levtraru, to Jorge Walters and many others that have provided valuable criticism and insight for the development of this model.

REFERENCES

Berners-Lee, T. 2002 The Semantic Web. *Scientific American*.

Carr, L., Hall, W. Bechhofer, S. And Goble, C. A. 2001 Conceptual linking: ontology-based open hypermedia. *In World Wide Web pp. 334-342*

Ceri, S. Fraternali, P., and Bongio, 2000 A. Web Modeling Language (webml): a modeling language for designing web sites. *WWW9/Computer Networks 33 1-6,137-157.*

Date, C.J. and Darwen, H. A 1998 Guide to the SQL standard. *Addison-Wesley, Reading, Mass., 199*3.

Decker, S., Erdmann, M., Fensel, D., and Studer, R. Research and Advanced Technologies for Digital Libraries. *Springer-Verlag, 1998, ch. Ontobroker in a Nutshell.*

Dourish, P., Edwards, K, LaMarca, A., and Salisbury, M. 1999 Presto: An Experimental Architecture for fluid interactive document space. *ACM Transactions on Computer-Human Interaction*.

Dourish, P., Edwards, K., LaMarca, A., and Salisbury,M. 1999.Using properties for uniform interaction in the presto document system. *In User Interface Software and Technology.*

Fraternali, P. 1999 Tools and approaches for developing data intensive web applications: a survey. *ACM Computing Surveys 31, 3, 227-263*

Gaedke, M., Segor, C.,Gellersen, H.-W. 2000 WCML: Paving the Way for Reuse in Object-Oriented Web Engineering, *ACM Symposium on Applied Computing, Italy*.

Gamma, E.,Helm, R., Johnson, R., and Vlissides, J. 1995 Design Patterns. *Addison-Wesley, Reading, MA.*.

Garrido, A., Rossi, G., and Schwabe, D. 1997 Pattern systems for hypermedia. *In Hypertext Conference.*

German, D. And Cowan, D. 2000 Towards a unified catalog of hypermedia design patterns. *In Hawaii International Conference on System Sciences.*

Heflin, J., Hendler, J. And Luke, S. 1999 Shoe: a knowledge representation language for internet applications. *Tech rep. University of Maryland.*

Maturana, H.1998 Ontology of observing. *In Text in Cybernetics.*

Roberts, J. 2000 Multiple-view and multiform visualization. *In SPIE.*

Schwabe, D. And Rossi, G. 1998 An object-oriented approach to web-based application design. *Theory and practice of object systems.*

Wood, J. Wright, H and Brodlie, K. 1997 Collaborative visualization. *In IEEE Visualization.*

Web Sites (verified 2002)

[coldfusion] Allaire's Coldfusion product page. <http://www.allaire.com/Products/coldfusion/index.cfm>

[cocoon] Apache XML integration project – cocoon. <http://xml.apache.org/>

[cpan] Comprehensive Perl archive network. <http://www.cpan.org/>

[hdpr] Hypermedia design pattern repository. <http://www.designpattern.lu.unisi.ch/>

[asp] Microsoft developer network – asp resources. <http://msdn.microsoft.com/asp>

[newtenberg] Newtenberg digital publishing ltd. <http://www.newtenberg.com/>

[php] PHP – The hypertext preprocessor. <http://www.php.net/>

[python] Python programming language. <http://www.python.org/>

[wapforum] Wap forum. <http://www.wapforum.org/>

[webdav] Webdav resources <http://www.webdav.org/>

[html] World Wide Web consortium- html <http://www.w3.org/MarkUp>

[rdf] World Wide Web consortium – rdf <http://www.w3.org/TR/PR-rdf-syntax>

[xml] World Wide Web consortium – xml <http://www.w3.org/XML>

[xsl] World Wide Web consortium – xsl <http://www.w3.org/Style/XSL>