*High Energy Physics Libraries Webzine*

**Issue 8 / October 2003**

# Experience in Computer-Assisted XML-Based Modelling in the Context of Libraries

**Marko Niinimäki, Vesa Sivunen** (*)

## Abstract:

In this paper, we introduce a software called Meta Data Visualisation (MDV) that (i) assists the user with a graphical user interface in the creation of his specific document types, (ii) creates a database according to these document types, (iii) allows the user to browse the database, and (iv) uses native XML presentation of the data in order to allow queries or data to be exported to other XML-based systems. We illustrate the use of MDV and XML modelling using library-related examples to build a bibliographic database.

In our opinion, creating document type descriptions corresponds to conceptual and logical database design in a database design process. We consider that this design can be supported with a suitable set of tools that help the designer concentrate on conceptual issues instead of implementation issues.

Our hypothesis is that using the methodology presented in this paper we can create XML databases that are useful and relevant, and with which MDV works as a user interface.

## 1 Introduction

The aims of this paper are (i) to survey modelling from the point of view of computer science and information studies, (ii) to study how XML can be used in this context and what its benefits are, and (iii) to outline an implementation of an XML-based modelling tool. The details of the implementation will be explained in a forthcoming paper.

According to Reingruber and Gregory [1], a model in general is 'a hypothetical or stylized representation' it attempts to capture in meaningful form some larger, or in some cases smaller, object that exists or will exist. Modelling, naturally, is the process of forming these models. We call the object of modelling a domain of interest or, briefly, domain.

There are possibly dozens (if not hundreds) of different accounts of modelling in different domains - engineering, economics, medicine, and social sciences, to name just a few. In the scope of this paper we limit our interest to modelling in computer science and information studies, and in order to do so, we make some simplified assumptions:

- Assumption 1: models and modelling are important tools for capturing information about the domain.
- Assumption 2: this information can be expressed in textual form, that we call documents.
- Assumption 3: the structure of the documents reflects the features of the domain.

- Assumption 4: these structured documents can be stored in a database and queries that take advantage of the structure of the documents can be supported by the database management system.

In information studies, the main emphasis has been to provide users with tools for finding the information they are looking for. Traditionally, this has been a two-phase process: first the information provider (say, a library or a publisher) creates a bibliographic description of the document. The contents of this description alone are then used by the user (the information requester). With the help of this information, the user locates the actual information source (e.g. a book) and retrieves it. An alternative for this two-phase process is one in which the user has the documents on-line and queries this repository of documents. Everyone who has searched documents using World Wide Web's popular search engines, knows that the query does not always yield relevant results; for instance the query "Donald Knuth" not only returns references (URLs) to actual documents written by Donald E. Knuth, but also lots of articles about Knuth. In order to make the difference between Knuth as an author and Knuth as a subject, a description of the content of the document is needed. XML offers a data definition language for creating a model of the document (see Ref. [2]). This model of the document structure can be used in describing the contents of the document. For example, we can extract the author elements and find the documents that have "Donald Knuth" as their author element.

In computer science, modelling in the form of conceptual modelling, information modelling, database modelling, and data modelling has been applied in database design, artificial intelligence, and software design (see Refs. [3], [4], [5]). Despite the differences in terminology, it seems that most of the approaches emphasise the use of abstraction mechanisms to capture knowledge about the domain. Abstraction mechanisms allow us to state that there are objects, attributes of objects, and different kinds of relations (of objects and attributes) in the domain of interest; briefly, they allow us to express the structure of the domain.

As Reingruber and Gregory [1] state, the model's representation relies on the adoption of a language. Using database terminology, a conceptual model is expressed by a conceptual schema using an appropriate language (see Ref. [6]). In information studies the bibliographic description rules are used to describe the origin and appearance of the document e.g. the document type. For content description, libraries have traditionally used natural language, usually in the form of keywords and abstracts, or controlled vocabularies (see Ref. [7]). In computer science, semantic data models, like the Entity-Relationship model and description logics ([8], [9]), are among the best known languages created for the purposes of the representation of the model. Analysis of different languages from the point of view of their applications and expressive power, for instance, has been a source of many studies (see for example [10], [3], [11], [12]).

Ultimately, the conceptual schema will be translated into a logical database schema that can be implemented using a database management system [13]. The result of the implementation is a database that can be populated with actual data. In the context of this paper, and according to our assumptions, we consider the database to be a collection of structured documents.

eXtensible Markup Language (XML) is a set of data representation conventions. A representation of something is called an XML document if the representation conforms with these conventions. These conventions do not state that a "real" document would always have an author or a title. In order to require such properties, we need to create a definition of our own, specific type of document. Naturally, this creation requires that we have knowledge about the domain of our documents and that we use this knowledge to impose a structure on our description of such documents - for instance that each one of them has at least one author. Once we impose this

requirement, the fact that each document has at least one author can be used in information retrieval, i.e. searching for documents based on the name(s) of the author(s).

The role of XML in this context can be summarised as follows.

- Constructing XML document types can be seen as modelling.
- It is possible to express abstraction mechanisms (stating that there are objects, attributes of objects and different kinds of relations) using XML.
- XML databases, their management software, and their query tools that take advantage of the structure of XML documents are readily available.

Meta Data Visualisation (MDV) software is a web application for managing XML-based data. The software supports both document type design and simple document management and query operations and contains its own database management system. Let us consider the database design process (as in Ref. [10]) as consisting of the collection of requirements, conceptual database design, the choice of database management system software, logical database design, physical database design, and implementation. We can say that MDV supports all these steps in a trivial way, and additionally provides the user with a graphical user interface by which he can browse his data.

In order to illustrate the use of XML and MDV, one could consider a domain of a wine farm. The user wants to model wines, their qualities and attributes, and their sales by bottles and barrels. The conceptual database design step, in this case, is the creation of document types corresponding to "wine", "wine bottle", "barrel", etc. This could be done by manually creating XML Document Type Definitions (DTDs), but most users would find this cumbersome. With MDV, the user creates these definitions interactively and then inserts the documents that are instances of these definitions.

## 1.1 Related work

Several XML-based web design frameworks like Expresso [14], Araneus [15] and WebML [16] (commercialized by WebRatio) are currently available. Many of them provide the user with sophisticated tools for web site building and work flow management. The scope of MDV is more limited but quite user friendly. The user of MDV does not need back-ground knowledge about Entity-Relationship models or other modeling languages. Moreover, the user is provided with a simple, intuitive interface with categories (that contain documents) and methods for designing and creating documents.

Obasanjo [17] divides XML databases into three different types (models). In the data-centric type, one may want to extract XML data, store XML data, or both, usually using a relational database. The document-centric type typically consists of a repository that stores XML documents; an editor and an engine with features like version, revision, and access control; the ability to re-use documents in different formats; web publishing facilities; and indexing and searching. In the hybrid type both document-centric and data-centric types of XML usage occur [17]. We consider the document-centric approach most relevant in the context of this paper. In Ref. [18], Graves presents database design methods for XML-based data. This paper follows the same guidelines, but expands the subject with modelling and implementation issues.

## 1.2 Contents

The rest of this paper is organised as follows. In Section 2 we define some terminology, in Section 3 we discuss document management in general and especially in libraries. Section 4 presents how abstraction mechanisms can be expressed with XML and, finally Section 5 describes, by an

example, how to use MDV in modelling and XML-based document management. Conclusions are presented in Section 6.

# 2 Terminology and conventions

XML (eXtensible Markup Language) is "the universal format for structured documents and data on the Web" [19]. As such, XML is a meta-language - a language for describing other languages - which lets one design customised markup languages for different types of documents, using a defined declaration syntax [20]. A particular XML language (like the one we use to describe documents in MDV) conforms to a grammar, that can be defined either using Document Type Definitions (DTDs) or XMLSchemas (see Ref. [19]).

According to the terminological conventions, an XML document consists of objects that are either entities or attributes (of entities). An XML processor (a computer program) can parse an XML document for as long as these conventions (and some further constraints) are not violated - even if there is no DTD for this document. The role of a DTD is to define valid element-type names and attribute-type names and in what order they must occur in the document.

A simple DTD and a document that conforms to it are shown in Figure 1 (adapted from Ref. [21]). Here, according to the DTD, in a valid "person" document, there can be one or more names and zero or more professions; a name can have attributes "last" and "first"; a profession can have attribute "value", and each of the attribute values may consist of arbitrary character strings. In addition to this, name and profession cannot have sub-elements, just attributes.

```
<?xml version="1.0"?>
 <!ELEMENT person (name+, profession*)>
 <!ATTLIST person id CDATA #IMPLIED>
 <!ELEMENT name EMPTY>
 <!ATTLIST name first CDATA #IMPLIED
                last CDATA #IMPLIED>
 <!ELEMENT profession EMPTY>
 <!ATTLIST profession value CDATA #IMPLIED>
]>
<person id="1">
   <name first="Alan" last="Turing"/>
   <profession value="computer engineer"/>
   <profession value="mathematician"/>
   <profession value="cryptographer"/>
</person>
```

**Figure 1:** A simple DTD and document.

Let us suppose we wanted to create meaningful links in our XML data, for instance that a person named Julius Mathison, a civil servant, is Alan Turing's father. For this purpose, the World Wide Web consortium (W3C) has defined XML linking language XLink that "specifies constructs that may be inserted into XML resources to describe links between objects" [22]. We do not include a DTD of Xlink here, but Figure 2 contains a document of another person (Julian Mathison) and an XLink presentation of the "father-of" relation between him and Alan Turing.

```
<person id="2">
     <name first="Julian" last="Mathison"/>
     <profession value="civil servant"/>
</person>
<father-of xmlns:xlink="http://www.w3.org/1999/xlink"
        xlink:type="extended">
<son xlink:type="locator" xlink:label="son"
             xlink:href='xpointer(//person[id = "1"])'/>
<father xlink:type="locator" xlink:label="father"
             xlink:href='xpointer(//person[id = "2"])'/>
<bind xlink:type="arc" xlink:from="father"
              xlink:to="son"/>
</father-of>
```

**Figure 2:** Another person and the father-of relationship.

"XML document" in this paper is a technical term and refers to a well-formed string of characters by which elements (entities or attributes) are coded. But most documents that we encounter are not XML documents. Salminen [23] presents a document as a part of the Electronic Document Management environment and regards the following characters as essential to a document:

- it is intended for human perception, to be understood as information pertaining to a topic;
- it has content and one, or more, external representations;
- the content consists of parts; the parts consist of symbols; the parts are structured to support human understanding;
- it is stored on media;
- it can be identified and handled as a unit.

Additional characteristics of a digital document according to Salminen are:

- the content is stored on digital media, and
- the document is associated with hardware and software which can identify from the digital content the symbols and the structure of the parts, and produce from them external representation to be perceived by humans.

# 3 Document description and management

In this section, we consider modelling in the context of information studies and libraries. In the case of libraries, the domain already contains "documents" in the sense of Salminen (see above). However, modelling, i.e. description of these documents, is needed in order to capture their characteristics and to facilitate searching. We maintain that practices adopted in libraries can be useful in document description and document management in general.

Libraries have a long tradition in describing documents. It is done for several purposes, yet the main task is to help users to find the information they are looking for. The following is a short explanation of this modelling in the library context. The work has been usually divided into two: (i) description of the origin and appearance of the document and (ii) description of the contents of the document [24].

## 3.1 The origin and appearance of the document

When describing the origin and appearance of a document, libraries use the term "bibliographic data". Since libraries nowadays maintain many kinds of documents, it is better to use the term "metadata".

This kind of metadata is used, for example, to

- tell of the existence of a document after it has been published;
- store documents systematically and organise them into collections;
- build catalogues;
- control the access points (information elements of the documents, e.g. the title and author).

Libraries normally use International Standard Bibliographic Description instructions (or national/local versions of it) for building the metadata. The MARC format (Machine Readable Cataloging) is the most-used format for implementing the description instructions of physical documents, like books. For describing web-based documents there is also a widely accepted standard called Dublin Core [25].

## 3.2 The contents of the document

Often the information seeker is looking for a particular subject. In these cases the information about the content of the document is more interesting than the appearance. This is obvious in traditional libraries. Also in databases where the content cannot easily be extracted into something helpful for the user (e.g. picture databases), the content description is still the main tool for searching. Full-text databases usually provide searching from the whole text of the documents. There, the content description is not absolutely necessary, but can offer good support for finding the relevant information. Content description in libraries and information services is done by using natural language, documentary language or both.

### 3.2.1 Natural language

A keyword is a word or a phrase which describes the document and is used in the document. An abstract is a short description of the content of the document. It is provided by the author or, for example, by a librarian. Instructions exist on how to write an abstract: generally speaking it should be a short, non-critical, and independent text. There are two types of abstract: indicative and informative. Indicative abstracts describe the content and the way of presentation more broadly, while informative abstracts try to represent the content as accurately as possible [26].

Natural language produces problems for information retrieval because of its richness and versatility. One solution for these problems is to use a documentary language.

### 3.2.2 Documentary languages

Documentary languages, or controlled vocabularies, allow information seekers and content-describers to use the same language and expressions. There are two types of documentary languages called classification, and indexing languages ([24]).

#### 3.2.2.1 Classification languages

Classification is an old method for organising information, usually used in libraries. Classifications are used for a variety of tasks and subjects and so they obviously differ from each other.

A classification can be either universal across all subjects, like the widely used Universal Decimal Classification, or it may cover just a certain subject, like the PACS (Physics and Astronomy Classification Scheme) [27]. A classification scheme is often hierarchical and is normally easily convertible to follow the XML syntax.

#### 3.2.2.2 Indexing languages

In indexing, the document is given a descriptor(s) from a controlled vocabulary. A thesaurus is a special form of controlled vocabulary: it contains terms which have been organised according to their semantic relations. As with classification, controlled vocabularies vary from universal to subject-specific vocabularies [24].

### 3.3 Document description and XML

The MARC format was developed long before XML, originally by Library of Congress in 1968. There are many versions of MARC, usually created for national purposes, like FINMARC or UKMARC. MARC is not XML, but as it is well structured, it is relatively easily converted to XML. Several programs, conversion tools and DTDs for translating MARC to XML-MARC exist [28].

In the following sections, we shall give examples of the use of XML as a format for expressing bibliographic data, and describe our computer program MDV that assists the user in creating and manipulating these XML representations.

Since MDV can be used for this purpose, it can be seen as a bibliographic software package like Endnote, Reference Manager, or Biblioscape. However, MDV is free of charge, provides the users with a web interface and uses a non proprietory XML data format that makes it easy to combine databases with external resources. For example, MDV has been used as an integrated tool for the searching, categorizing, and sharing of documents containing digital images.

# 4 Representing documents using XML

As defined in Section 2, "XML document" is a technical term, and modelling is needed for representing in XML the information contained in real documents: we need to decide the format of XML in order to express the information in a meaningful form.

It is our assumption that the structure of the documents reflects the features of the domain. In what follows, we shall study how to model documents and represent them in an XML format.

In the case of document description and management, the actual modelling process has already been done by librarians, and there exist schemes to translate, for example, MARC into XML.

In libraries, metadata management is well-defined, however, bibliographic data is condensed and often there is only a limited amount of information in the metadata of the documents (metadata fields). In many other contexts, there is a need for a variety of types of records by which structural information can be presented. Designing them can be supported by XML and XML-based modelling.

In electronic document management, we need to conceptualise the domain in order to find the relevant structures reflected by the documents. This kind of conceptualisation has been made in the field of Semantic Data Models (see for example Refs. [29], [30]). We adopt the following

abstraction mechanisms since they appear frequently in documents and domains of documents (examples below)[1]:

- IS-A, i.e. the relation between a subtype and its supertype,

  for example: `THEORY-PREPRINT` IS-A `PREPRINT`;

- Attribution, i.e. functional relations,

  for example, `PREPRINT` has, among others, attributes of `title` and `author`;

- Association, i.e. named relations between some types of objects,

  for example, a relation of `precedes` exists between `PREPRINT` and `PUBLISHED ARTICLE`: `PREPRINT precedes PUBLISHED ARTICLE`;

- Aggregation, i.e. constructing something out of independent parts,

  for example, `BOUND PREPRINT` might contain `COVER` and `PREPRINT`;

- Grouping, i.e. creating finite sets of objects of a given type.

  for example, a grouping of `ARTICLES` can form a `COLLECTION OF ARTICLES`.

'Association' can be seen as 'attribution', in a way that a `PUBLISHED ARTICLE` could have an attribute `is preceded by` (whose value would be the respective preprint). Therefore, we omit 'association' in further discussion.

'Attribution' can be represented trivially using XML attributes as in Figure 1. But since 'IS-A', 'aggregation', and 'grouping' are basically relations, we have to adopt XLink to represent them. Examples of an 'IS-A', an 'aggregation' and a 'grouping' relation definition are shown in Figure 3. The 'IS-A' relation is demonstrated using the relation: `THEORY-PREPRINTS IS-A PREPRINT`. The 'aggregation' relation is demonstrated using the relation: `BOUND THESIS contains THESIS and COVER`. The 'grouping' relation is demonstrated using the relation: `ARTICLE is a group member of COLLECTION.`

```
<is-a xmlns:xlink="http://www.w3.org/1999/xlink"
        xlink:type="extended">
<subtype xlink:type="locator" xlink:label="subtype"/>
             xlink:href='xpointer(//theory-preprint])'/>
<supertype xlink:type="locator" xlink:label="supertype"
             xlink:href='xpointer(//preprint)'/>
<bind xlink:type="arc" xlink:from="subtype"
              xlink:to="supertype"/>
</is-a>

<aggregation xmlns:xlink="http://www.w3.org/1999/xlink"
        xlink:type="extended">
<contains xlink:type="locator" xlink:label="contains"/>
             xlink:href='xpointer(//bound-thesis)'/>
<is-contained xlink:type="locator" xlink:label="is-contained"
             xlink:href='xpointer(//thesis)'/>
<is-contained xlink:type="locator" xlink:label="is-contained"
             xlink:href='xpointer(//cover)'/>
<bind xlink:type="arc" xlink:from="contains"
              xlink:to="is-contained"/>
</aggregation>

<grouping xmlns:xlink="http://www.w3.org/1999/xlink"
        xlink:type="extended">
<group xlink:type="locator" xlink:label="group"/>
             xlink:href='xpointer(//collection)'/>
<group-member xlink:type="locator" xlink:label="group-member"
             xlink:href='xpointer(//article)'/>
<bind xlink:type="arc" xlink:from="group-member"
              xlink:to="group"/>
</grouping>
```

**Figure 3:** Xlink representations of IS-A, aggregation, and grouping.

# 5 Using MDV in designing and using the database

Though the XML representation explained in section 4 constitutes a solid XML foundation for representing documents, a set of these document type definitions probably means very little to a normal user. A user interface is needed for the user to add his own document types and relations in the database. Moreover, a user interface also enables the user to enter, search, edit, and delete actual documents.

The modelling features of MDV have been greatly influenced by Hull and King's accounts of Semantic Data Models ([29], [30]). The abstraction mechanisms that Hull and King discuss are those of 'IS-A', 'attribution', 'aggregation', and 'grouping', as explained above. As indicated by Hull and King [30], 'attributes' and 'aggregation' can both be used in some situations[2]. This can confuse users, and therefore in MDV we impose a simplifying rule that seems to be quite intuitive to most users: an aggregate type is *just* an aggregate; it cannot have attribute fields.

In MDV, these abstraction methods are related to document types and can be edited by the administrator. Document types may contain 0...n field types. In the simple case of Figure 4, a field (string, enumeration, or picture) is printable; in this way, MDV implements simple attributes.

```
PREPRINT                    Document id 5456
Author: STRING              Author: Dado, S and Dar, Arnon and De Rujula, Alvaro
Title: STRING               Title: The supernova associated with GRB 020405
Language: STRING            Language: en
```

**Figure 4:** A simple document type and an example document.

Other abstraction methods are managed by MDV as follows.

- Functional relations: MDV fields that are not printable are called functional relations. In Figure 5, the user has created a document type `PUBLISHED ARTICLE`, that has a field `preceded-by-preprint`. This field can be filled with a document of type `PREPRINT`. This is a method of implementing attributes whose values are non-printable.

```
PUBLISHED ARTICLE                   Document id 5457
Preceded-by-preprint: PREPRINT      Preceded-by-preprint: 5456
```

**Figure 5:** A document type with a functional relation.

- IS-A relations: the administrator can derive a new document type from an existing document type; the new type will be a subtype of the existing one. In Figure 6, the user has derived `THEORY-PREPRINT` from `PREPRINT`.

```
THEORY-PREPRINT                 Document id 5458
IS-A: PREPRINT                  Language: en
                                Author: Ellis, G F R and Uzan, J P
                                Title: 'c' is the speed of light,
                                       isn't it?
                                Subject: General Relativity and
                                         Cosmology
```

**Figure 6:** `THEORY-PREPRINT` document type and an example document.

- Grouping relations: the administrator can create a grouping of an existing document type. In Figure 7 a grouping `PREPRINT-COLLECTION` has been created.

```
PREPRINT-COLLECTION             Document id 5459
GROUPING-OF: PREPRINT           GROUPING-OF values: 5456, 5458
```

**Figure 7:** `PREPRINT-COLLECTION` document type and an example document.

- Aggregate relations: shown in Figure 8 by the use of `has-part-1: document type`, etc. As mentioned earlier, MDV will not allow an aggregate to also have attributes. Therefore, if the field: `price` is to be included in the definition of a `BOUND PREPRINT`, then all the relationships must be described as attributes as in the second example in Figure 8.

```
BOUND PREPRINT                    Document id 5461
has-part-1: PREPRINT              has-part-1: 5460
has-part-2: COVER                 has-part-2: 5458

BOUND PREPRINT                    Document id 5461
Thesis: PREPRINT                  Thesis: 5460
Cover: COVER                      Cover: 5458
Price: STRING                     Price: 0 CHF
```

**Figure 8:** Two ways of implementing `BOUND PREPRINT`: the first using aggregation, the second using attribution.

As an example, we demonstrate how to create corresponding document types, relations and documents using MDV. The basic functionality of the software is explained in Ref. [31], but Figures 9 and 10 are shown here for completeness.

Figures 11 and 12 illustrate how to create a new basic document type `PREPRINT`. Figure 13 shows how to add fields to it to get a simple document type as in Figure 4.

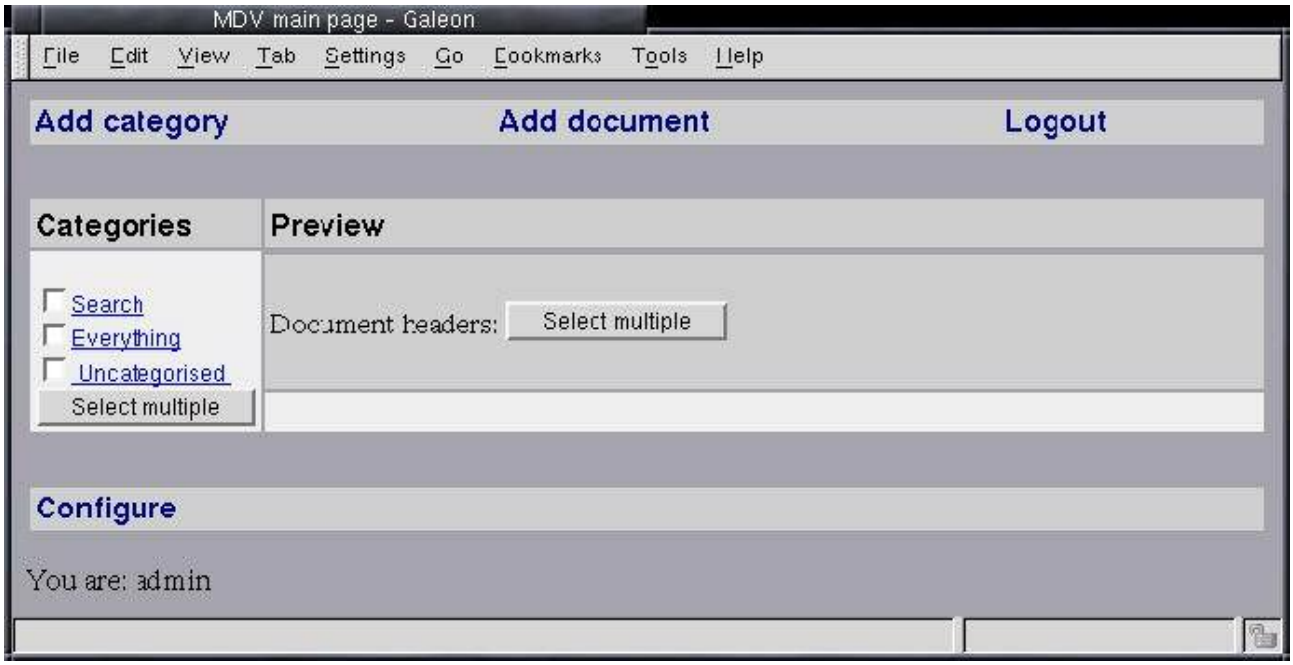


**Figure 9:** MDV main page.

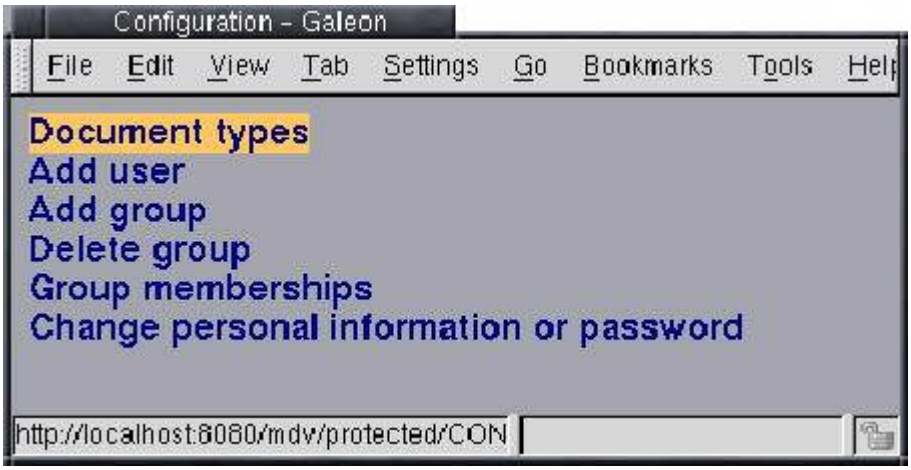


**Figure 10:** MDV configuration page. This page is accessed by clicking the "Configure" link on the main page.

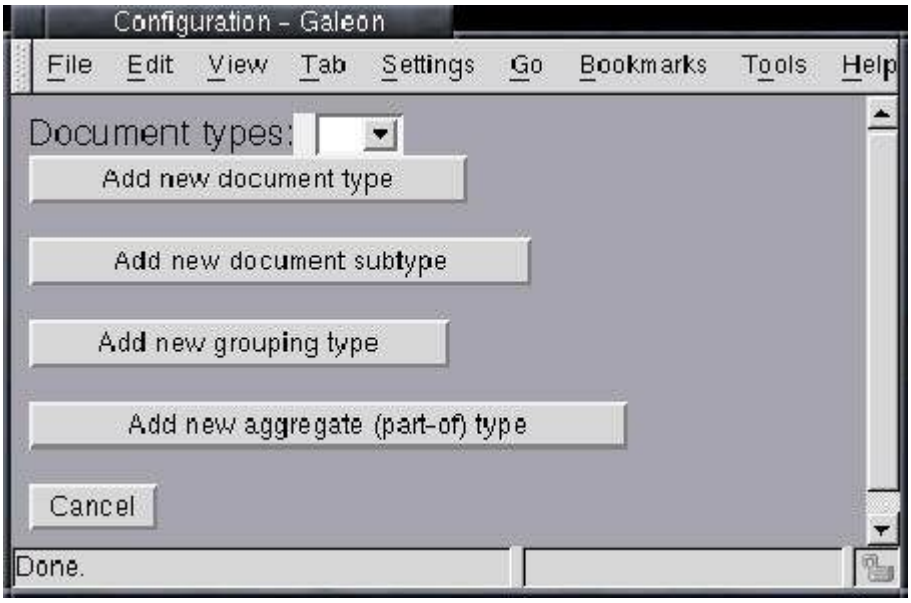


**Figure 11:** MDV document types configuration page. This page is accessed by clicking the "Document types" link on the previous page.

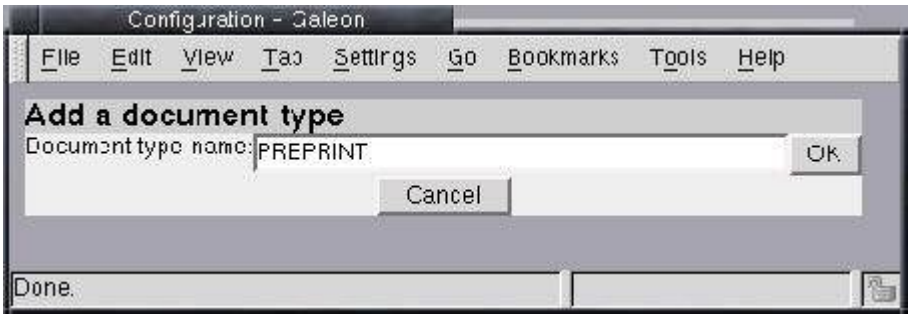


**Figure 12:** Adding new document type `PREPRINT`. This page is accessed by clicking the "Add new document type" link on the previous page.


**Figure 13:** Adding fields `Author`, `Title` and `Language` to `PREPRINT`.

Figure 14 illustrates the creation of a document type: `PUBLISHED-ARTICLE` and the creation of a functional relation from it to `PREPRINT`, as in Figure 5.


**Figure 14:** PUBLISHED-ARTICLE and its relation to PREPRINT.

In Figure 15 we see the user creating a new document type THEORY-PREPRINT as a subtype of PREPRINT, as in the IS-A relationship in Figure 6.

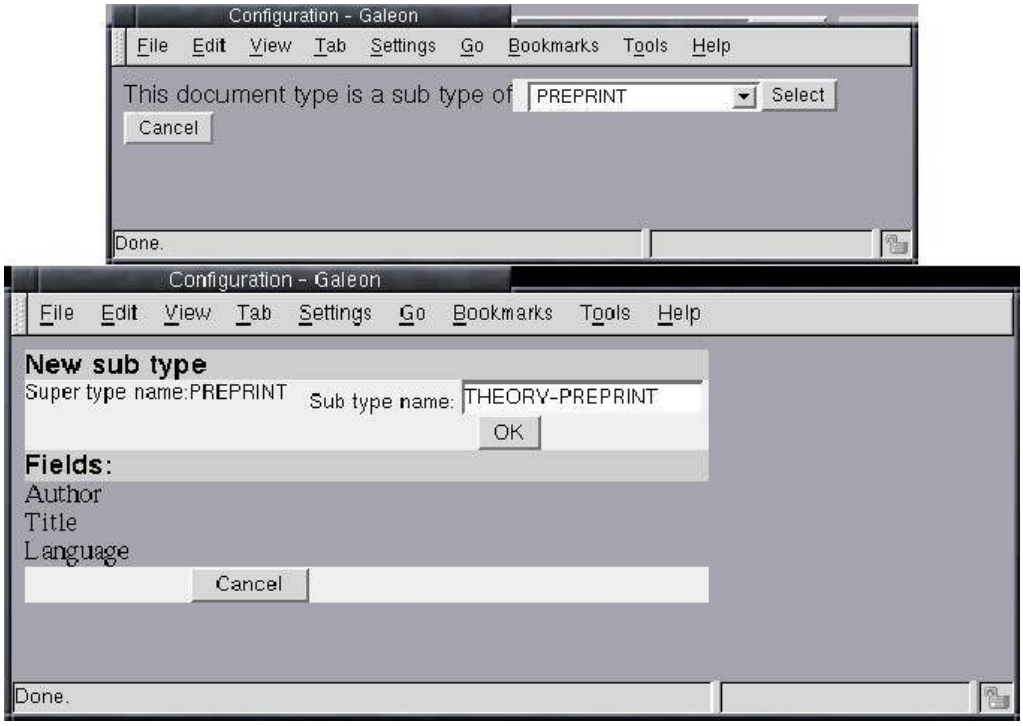


**Figure 15:** Deriving THEORY-PREPRINT from PREPRINT.

Figure 16 shows a grouping PREPRINT-COLLECTION, whose members are of type PREPRINT. The user creates a new grouping by clicking on the link "Add new grouping type" on the document type configuration page shown in Figure 11. The grouping relationship will create a link in the user interface of Figure 19. The user has to indicate (with the "Categorise" function of MDV) the documents that he wants to belong to a given PREPRINT-COLLECTION. This, naturally, corresponds to classification, discussed in Section 3.2.2.

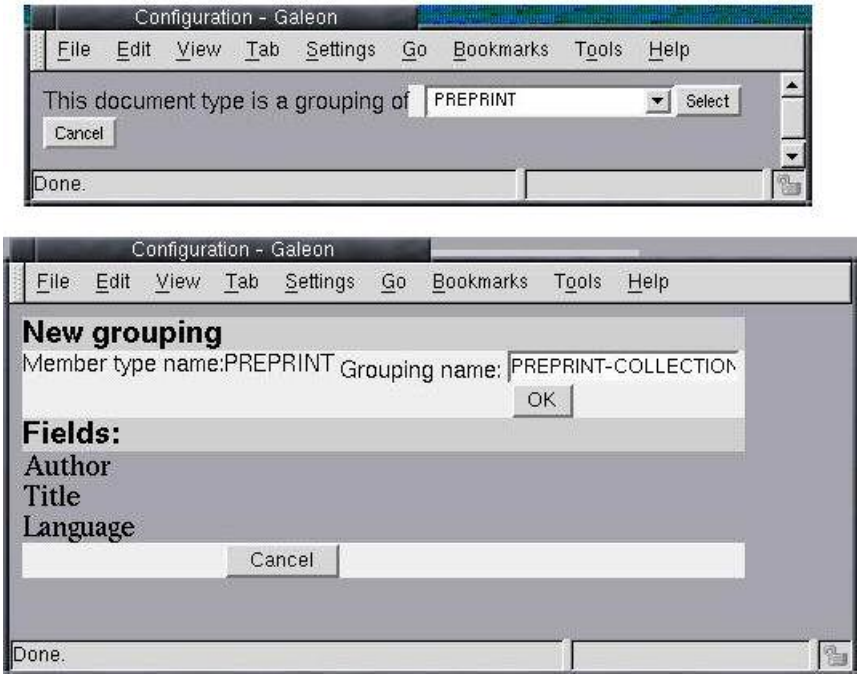


**Figure 16:** A grouping of `PREPRINTs`.

In an aggregation, only references to other (existing) document types are possible. The user creates a new aggregate by clicking on the link on the document type configuration page shown in Figure 11. Figure 17 shows how to then create the document type `BOUND PREPRINT` that has `Preprint` (of document type `PREPRINT`) as one of its parts. In the figure, the user creates a field (part) `Preprint` in `BOUND THESIS` and states that only `PREPRINT` document types can be in this field.

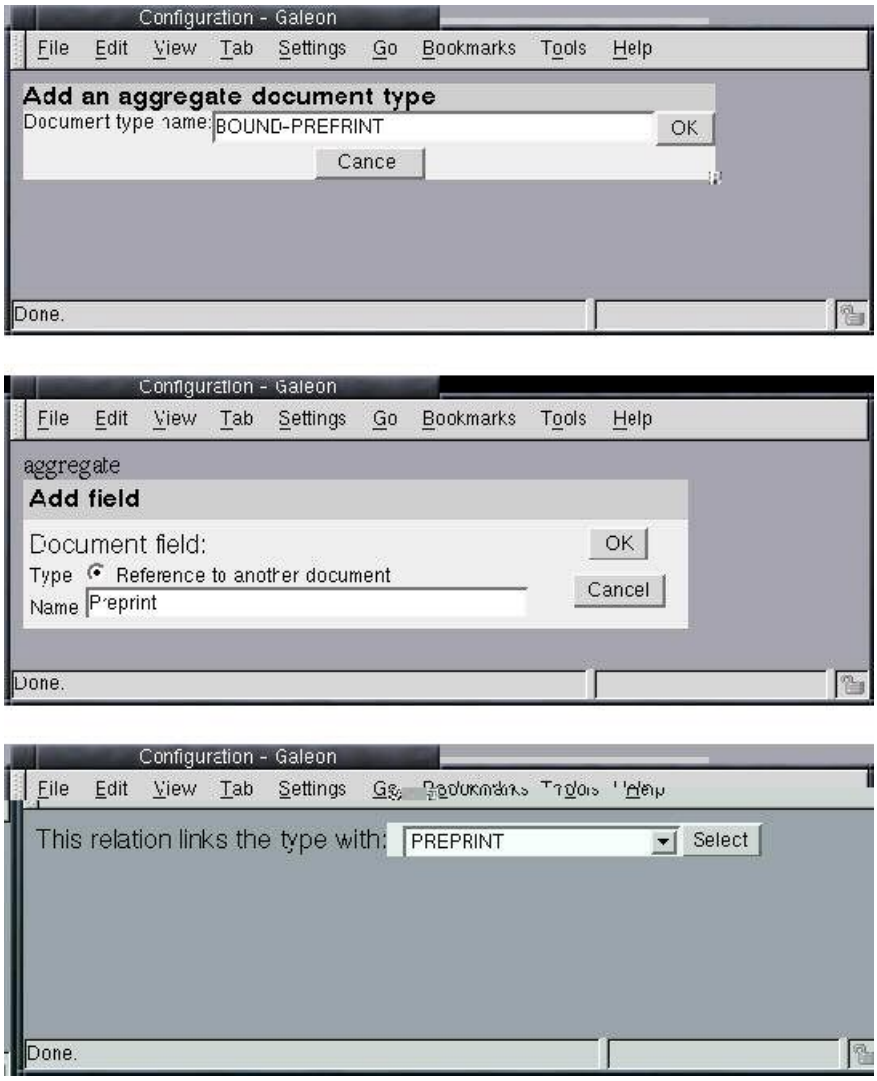


**Figure 17:** Creating an aggregate.

In Figure 18 we demonstrate the user interface for creating and editing documents using MDV. These functions are accessible through the "Add document" link from the MDV main page shown in Figure 9. In Figure 18, the user creates a `PREPRINT` document.

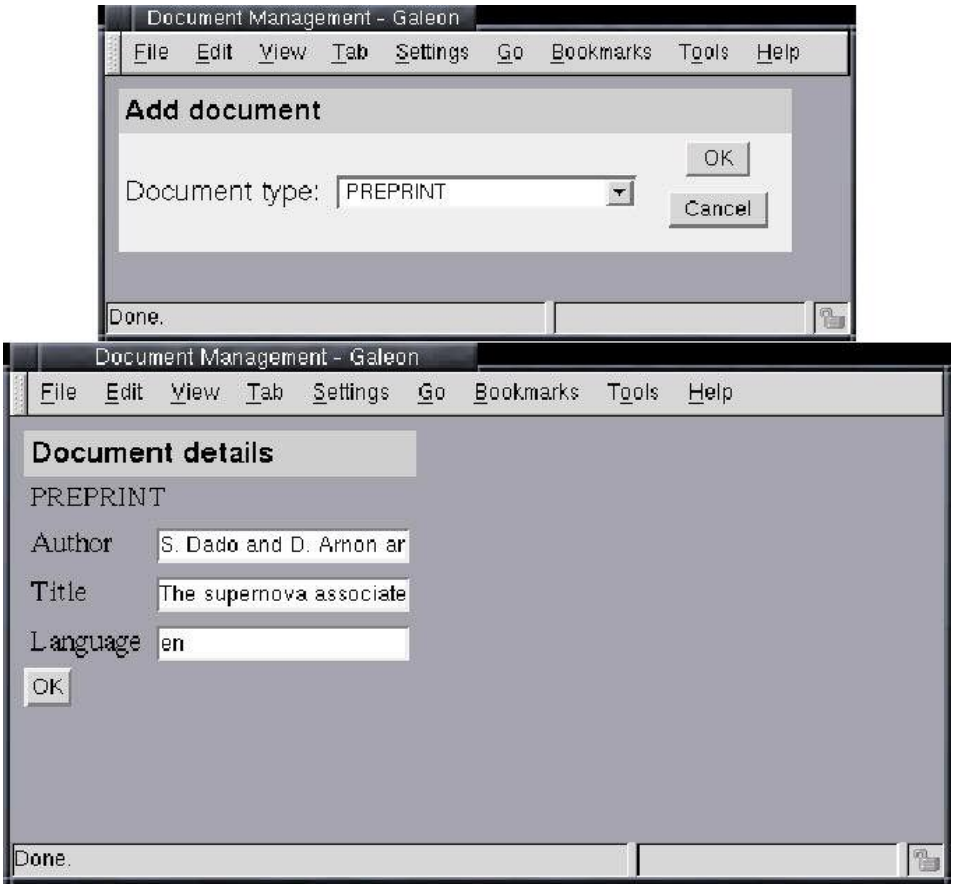


**Figure 18:** Creating and editing objects.

Our assumption 4 (detailed in the introduction) states that with structured documents, queries can take advantages of the structure. In Figure 19, we demonstrate the query interface, as follows:

- Refer back to Figure 13; the user checked in the check-boxes to enable searching by `Title` field;
- Figure 19: a new item, "Search by Title" is now available in the list of categories on the left of the window. To search in the title field, the user types a title keyword into the dialog box;
- A set of documents corresponding to the query is returned in the preview pane.

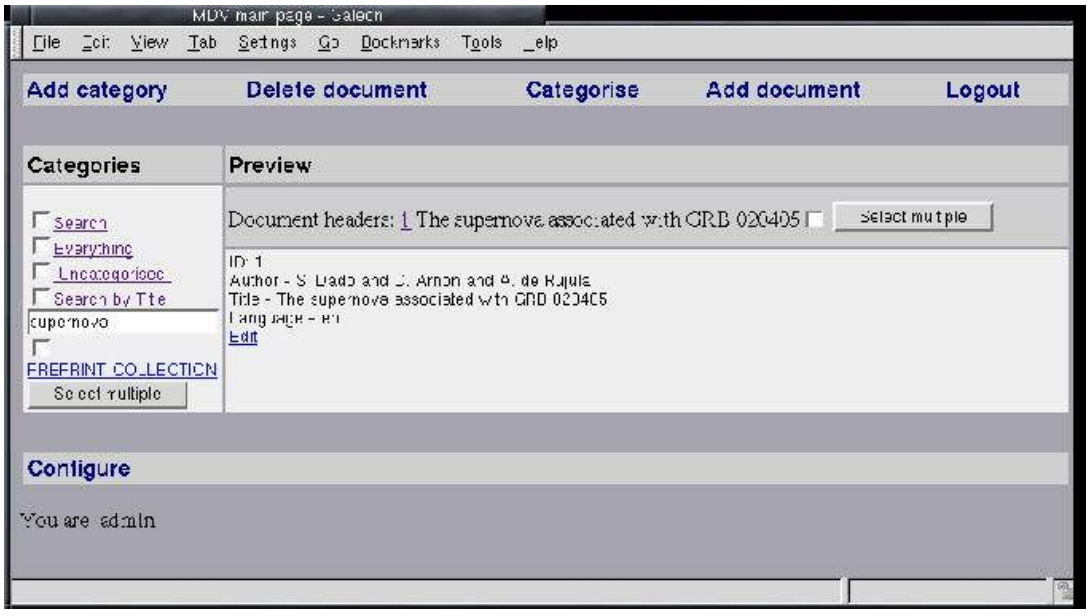

**Figure 19:** Searching by structure.

# 6 Discussion and conclusions

Models and modelling are used in many areas to capture knowledge of the domain. We have discussed how modelling is done in computer science and in information studies. The document management tradition in libraries has served as an example of how a document collection can be described to help the efficient searching and retrieving of the needed documents.

There is a lack of suitable tools which can facilitate the modelling process. Our solution is the XML framework called Meta Data Visualisation. The MDV software assists the user in the creation of the specific document types with a graphical user interface. Then documents are stored into MDV's database. This database can be browsed and queried or data can be exported to other XML systems. We have demonstrated the functionality of MDV using a real life example that uses the abstraction methods (IS-A, attribution, aggregation, ...) in a natural manner. The same methods can be used in creating a library catalogue system, described in Section 3, as we plan to do in the near future. Another item under development is a model browser that illustrates MDV's document types and their relations with each other. This will allow users to learn quickly about the information contained in the domain.

We believe that our methodology and the MDV software will help the modelling work and by using it one can create XML databases that are useful and relevant. MDV is open source software and is downloadable from http://mdv.sourceforge.org/.

# Bibliography


1

M. Reingruber and W.W. Gregory.
*The Data Modeling Handbook, A Best-Practice Approach to Building Quality Data Models*.
Wiley and Sons, 1994.

2

A. Salminen and F. Tompa.
Grammars++ for modelling information in text.
*Information Systems*, 24(1), 1999.

3

G. Engels, M. Gogolla, U. Hohenstein, K. Hulsmann, P Lohr-Richter, G. Saake, and Ehrich H.-D.
Conceptual modelling of database applications using an extended ER model.
*Data and Knowledge Engineering*, 9(4), 1992.

4

H. P. Winston.
Learning structural descriptions from examples.
In H. P. Winston, editor, *Psychology of Computer Vision*. McGraw-Hill, 1975.

5

M. Fowler and K. Scott.
*UML Distilled*.
Addison-Wesley, 2nd edition, 2000.

6

Information processing systems Technical Committee ISO/TC 97.
*Information processing systems - Concepts and Terminology for the Conceptual Schema*

*and the Information Base*.
ISO, TR 9007:1987 (E), 1987.

7

The International Federation of Library Associations and Institutions.
*Cataloguing and indexing of electronic resources.*
Available on: http://www.ifla.org/II/catalog.htm, 2003.

8

P. Chen.
The entity-relationship model - towards a unified view of data.
*ACM Transactions on Database Systems*, 1(1), 1976.

9

A. Borgida.
Description logics in data management.
*IEEE transactions on knowledge and data engineering*, 7(1), 1995.

10

P. Lambrix.
*Part-Whole Reasoning in Description Logics*.
PhD thesis, Linköping Studies in Science and Technology No. 448, 1996.

11

D. Calvanese, M. Lenzerini, and D. Nardi.
Description logics for conceptual data modeling.
In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*.
Kluwer Academic Publisher, 1998.

12

F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt.
The complexity of concept languages.
*Information and Computation*, 134(1), 1997.

13

R. Elmasri and S. Navathe.
*Fundamentals of Database Systems*.
Benjamin/Cummings, 2nd edition, 1994.

14

Jcorporate.
*Expresso framework project.*
Available on: http://www.jcorporate.com/, 2002.

15

G. Mecca, P. Merialdo and P. Atzeni.
Araneus in the era of XML.
*IEEE Data Engineering Bullettin*, Special Issue on XML, 1999.

16

S. Ceri, P. Fraternali, and A. Bongio.
Web modeling language (webml): a modeling language for designing web sites.
In *Proceedings of the Ninth International World Wide Web Conference*, Amsterdam, May 15-19, 2000.

17

D. Obasanjo.
*An exploration of xml in database management systems.*
Available on: http://www.25hoursaday.com/StoringAndQueryingXML.html, 2001.

18

M. Graves.
*Designing XML databases*.
Prentice Hall, 2001.

19
The World Wide Web Consortium.
*Extensible markup language (xml).*
Available on: http://www.w3.org/XML/, 2003.
20
The World Wide Web Consortium.
*Extensible markup language (xml) 1.0 (second edition), W3C recommendation 6 October.*
Available on: http://www.w3.org/TR/2000/REC-xml-20001006, 2000.
21
E. R. Harold and W. S. Means.
*XML in a Nutshell.*
O'Reilly, 2001.
22
The World Wide Web Consortium.
*Xml linking language (xlink) version 1.0.*
Available on: http://www.w3.org/TR/xlink/, 2001.
23
A. Salminen.
Methodology for document analysis.
In A. Kent, editor, *Encyclopedia of Library and Information Science*, volume 67. Marcel Dekker, 2000.
24
M. E. Soper and L. N. Osborne et al.
*The librarian's thesaurus : a concise guide to library and information terms.*
IL ALAA, 1990.
25
DCMI.
*Dublin core metadata initiative (dcmi).*
Available on: http://dublincore.org/, 2003.
26
National Information Standards Organization.
*Guidelines for Abstracts, Z39.14-1997.*
NISO, 1997.
27
American Institute of Physics.
*Physics and astronomy classification scheme.*
Available on: http://www.aip.org/pacs/, 2003.
28
The Library of Congress.
*Marcxml, marc 21 xml schema.*
Available on: http://www.loc.gov/standards/marcxml/, 2003.
29
S. Abiteboul and R. Hull.
IFO: A formal semantic database model.
*ACM Transactions on Database Systems*, 12(4), 1987.
30
R. Hull and R. King.
Semantic data modelling: Survey, applications and research issues.
*ACM Computing Surveys*, 19(3), 1987.
31
M. Niinimaki, M. Tuisku, and M. Heikkurinen.
*Patterns, XML and MDV platform, a case study.*

Technical report, Department of Computer and Information Sciences, University of Tampere, 2002.

# About this document ...

**Experience in computer assisted XML-based modelling in the context of libraries**

This document was generated using the **LaTeX**2HTML translator Version 2002-2 (1.70)



The command line arguments were:
**latex2html** `-nonavigation -split 0 xml-modelling-webzine.tex`

The translation was initiated by Marko NIINIMAKI on 2003-07-02

---

# Footnotes

... below)[1]
In what follows, the names of types are written in uppercase and names of fields, attributes, and relations in lowercase.

... situations[2]
Hull and King discuss an example where the relationship of people and their businesses can be represented (i) by `PERSON's` attribute `works-for` whose values are `BUSINESSes` or (ii) by the aggregate `EMPLOYMENT` that has `person` and `business` as its parts.

---

*Marko NIINIMAKI 2003-07-02*

# Author Details

**Marko Niinimäki**
Helsinki Institute of Physics at CERN, CH-1211 Geneva, Switzerland
email: Marko.Niinimaki@cern.ch

**Vesa Sivunen**
Helsinki Institute of Physics at CERN, CH-1211 Geneva, Switzerland
email: Vesa.Sivunen@cern.ch

# For citation purposes:

Marko Niinimäki and Vesa Sivunen "*Experience in computer-assisted XML-based modelling in the context of libraries*", High Energy Physics Libraries Webzine, issue 8, October 2003
URL: http://library.cern.ch/HEPLW/8/papers/2/