Mapping XML Instances

Sai Anand
ETH Zürich
anand@tik.ee.ethz.ch

Erik Wilde
ETH Zürich
net.dret@dret.net

ABSTRACT

For XML-based applications in general and B2B applications in particular, mapping between differently structured XML documents, to enable exchange of data, is a basic problem. A generic solution to the problem is of interest and desirable both in an academic and practical sense. We present a case study of the problem that arises in an XML based project, which involves mapping of different XML schemas to each other. We describe our approach to solving the problem, its advantages and limitations. We also compare and contrast our approach with previously known approaches and commercially available software solutions.

Categories and Subject Descriptors: D.2 [**Software**]: Software Engineering; H.4.m [**Information Systems**]: Miscellaneous

General Terms: Algorithms, Experimentation, Design

1. INTRODUCTION

The *Shared References (ShaRef)* project [3] develops software that helps maintain bibliographic information in individual and shared settings, so that bibliographic data can be shared between users. One of the central issues in the design of the tool is its ability to import and export bibliographies maintained by individual users in various formats. The system uses an internal data model defined by an XML schema. To provide flexibility to users that migrate from other formats and also for those that wish to use their bibliographies outside of ShaRef, import and export features are provided. Currently, BibTEX, Endnote 7/8, and MODS are supported. Figure 1 shows the import/export design. Note how native non-XML formats (like BibTEX and Endnote) are handled by first parsing them into XML.

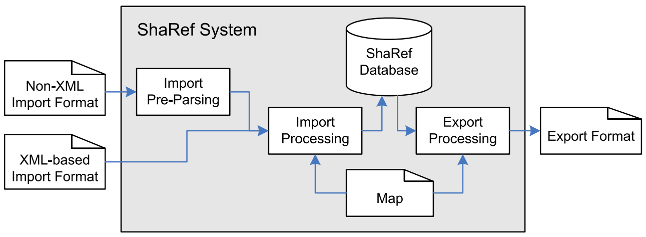

Figure 1: Sharef Import/Export Design

A principal problem that occurs in many XML-based applications is that of mapping XML schemas ("XML schema" is used throughout this paper to refer to the generic concept, and not only the W3C schema language) to each other. This problem is not limited or special to XML alone and is common in database applications as well. In the database domain, the mapping problem is well studied. A survey paper by RAHM and BERNSTEIN [2] characterizes the various automated approaches to schema matching, which are also relevant to mapping XML schemas.

There are two aspects to the general mapping problem: a) semantic interoperability and b) instance mapping. Semantic interoperability addresses how schemas are mapped to each other and if, in fact, they can be mapped at all. The instance mapping aspect addresses how instances of one schema are mapped to the other. Because semantic interoperability requires domain knowledge but no implementation, it can logically be completely separated from instance mapping. In practice, however, there often is an overlap that makes it harder to do the separation.

2. MAPPING XML INSTANCES

While mapping XML instances itself is essential for implementing XML interchange, it has to be combined with another mapping procedure, which is the mapping of XML structures to application structures. These application structures may be objects (in case of OO languages) or other runtime constructs. Only this mapping to application structures makes it possible to work with the XML data in the application environment. Figure 2 shows how these mappings relate. While the XML-XML mapping is located between the different application environments, each application environments has its own way of mapping XML to application data structures, a process that is often referred to as "XML Data Binding".

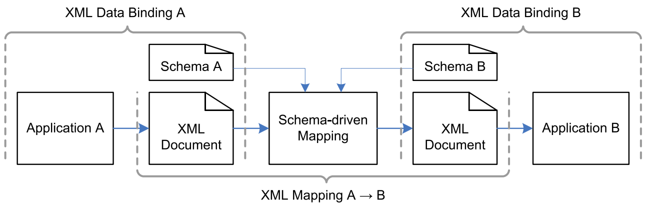

Figure 2: The XML Instance Mapping Process

For achieving semantic interoperability, it is necessary to specify how nodes (elements and attributes) of two schemas relate to each other. It is useful to distinguish the following four kinds of mappings of nodes:

- *One-to-One (1:1):* The value of a node in one schema is mapped to another node in the other schema, with little or no processing. This is the simplest mapping between nodes. For example, two schemas use different names to designate the same concept.
- *Many-to-One, One-to-Many (1:n):* Here, two or more nodes of one schema are mapped to the same node

in the other. For example, Endnote defines different elements for storing the ISBN/ISSN numbers of a reference and for URLs. MODS, on the other hand, stores everything in a single element called `identifier` and distinguishes them using an attribute of this element. Thus, mapping ISBNs/ISSNs and URLs to `identifier`s is a many-to-one mapping.

- *Unions:* With Union mappings, values of several nodes in the source schema are mapped onto a single node in the target schema. The difference between many-to-one and union mappings is that whereas in the former the individual nodes of the source schema are still separate nodes in the target schema, in the latter values of separate entities of the source schema are combined into one element or entity. For example, combining the values of `year` and `month` nodes in the source schema to a single value in the `date` node of the target schema is a union mapping.
- *Context Sensitive mappings:* In several cases, mapping of nodes cannot be done by semantical considerations alone. The mapping is also influenced by the context within which it occurs. For example, `author`s and `editor`s in BibTEX are mapped onto `person` elements in ShaRef, and are distinguished based on an attribute of the `person` element. In this case, the export process has to make use of context sensitive information for producing a correct BibTEX entry.

MapForce™ and Stylus Studio are two industry-grade software solutions for developing XML applications. They provide a graphical interface for performing XML schema mappings, where mappings between nodes are specified by connecting them. Once the mappings are specified, XSLT code that implements the mapping is automatically generated. This is very intuitive for one-to-one mappings. While many-to-one and one-to-many mappings are also handled by the GUI, it usually requires some explicit implementation effort on the part of the designer to get the expected output. Union mappings and context sensitive mappings are either handled only partially or not at all. Thus, their use is limited when specifying mappings moderate complexity. Another drawback is that they handle mappings unidirectionally, i.e. information available from mapping one schema to another cannot be reused when doing the reverse mapping.

Handt and Quantz [1] provide a formal framework for the general schema mapping problem. They call it *XML Schema correspondences*. Their essential idea is to not only specify the mappings between the various elements of the two schemas, but also to describe how the transformation is to be done. In a second step, the above specification is used to generate "transformer" code that actually performs the conversions of instances of one schema into another. The advantage of the approach is that it does provide a generic solution to the mapping problem. However, the fact that the mapping and the transformation specifications are integrated means that the "domain expert" would have to go beyond just mapping the two schemas and specify the transformation as well.

Our approach is to separate the mapping specification from the code that implements the actual mapping itself. We call it the *2-phase* approach. This is done, firstly, by specifying the map as an XML document, each mapping node of which specifies what the mapping from an external format (BibTEX, Endnote 7/8, or MODS) to ShaRef is. In the case of one-to-one, one-to-many or many-to-one mappings, this is straightforward. In the case of union and context sensitive mappings, the mapping node uses attributes to specify that some extra processing needs to be done. Thus, most of the domain specific details are isolated in the map. Also, conversion between two external formats is achieved by going via the internal data model.

	1:1/1:n	Union	Context	2-phase	2-way
MapForce	✓		(✓)		
Stylus Studio	✓		(✓)		
[1]	✓	✓	✓		
ShaRef	✓	✓	(✓)	✓	✓

Table 1: Comparison of the various approaches

Secondly, the code that actually implements the mapping is an XSLT program. The XSLT has as its input an XML document that needs to be transformed into another XML format. For each node in the source XML document, the XSLT refers to the map to produce the corresponding node in the target format. For union and context sensitive mappings, a complete isolation of the domain specific details in the map file would have lead to a complicated specification. This, in turn, would mean implementing XSLT code to decipher the specification. We avoided this by building some domain specific knowledge directly into the XSLT code.

Table 1 shows a comparison of the approaches and software solutions in terms of their capability to handle the different mappings, whether they separate the mapping and implementing stages (2-phase), and whether they support using the mapping information for both mapping directions (2-way). The parentheses around a tick mark (✓) indicate that the mapping is handled in a limited sense.

3. CONCLUSIONS AND FUTURE PLANS

When searching for a solution for the mapping problem in the ShaRef project, it turned out that there was no generic way to handle the problems associated with the application scenario. The approach described here is being used in the ShaRef software and provides a reasonable solution for the mapping problems. While this is sufficient for the immediate project requirements, the approach would need some refinement to be evolved into a more generic approach. We believe that the two-pass approach of separating the matching specification, and the implementation of the actual matching process, is a useful separation between the more knowledge-oriented domain of schema matching, and the more code-oriented domain of implementing instance matching. Even though there is surprisingly little research being done in this area, we hope that the ubiquitous problem of mapping XML instances will receive more attention in the future.

4. REFERENCES

[1] Arne Handt and Joachim Quantz. XML Schema Correspondences. In Robert Tolksdorf and Rainer Eckstein, editors, *Proceedings of XSW 2002 — XML Technologien für das Semantic Web*, volume 14 of *Lecture Notes in Informatics*, pages 93–104, Berlin, Germany, June 2002. Gesellschaft für Informatik.

[2] Erhard Rahm and Philip A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *The International Journal on Very Large Data Bases*, 10(4):334–350, December 2001.

[3] Erik Wilde. References as Knowledge Management. *Issues in Science & Technology Librarianship*, No. 41, Fall 2004.