A METHOD FOR INTEROPERABILITY BETWEEN STRUCTURED DATA SOURCES USING SEMANTIC ANALYSIS

David L. Brock and Jyotsna Venkataramanan
Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, U.S.A.

Keywords: Interoperability, Knowledge representation, Schema, Semantics, Structure data, XML.

Abstract: The vocabulary and hierarchical organization of heterogeneous XML schemas were examined using semantic analysis and the correspondence between disparate data elements estimated. A prototype implementation was developed, and a number of large, real-world schemas automatically analyzed.

1 INTRODUCTION

The Extensible Markup Language has become the standard for exchanging structured information over the Internet. Industry and government have been steadily building and deploying network centric solutions using XML as the communication standard. Many hundreds – if not thousands – of XML-based languages have been developed and encoded as XML schema. This, however, is the essence of the problem. While these languages cover a wide range of applications and domains, the generality and flexibility of the Extensible Markup Language has made the integration and exchange of data *between* domain specific implementations very challenging.

The disparity of representation has made the task of combining data from multiple structured sources difficult, error-prone and expensive. This paper presents a new approach to structured data interoperability by processing disparate XML data representations to an encompassing generic model. By examining this underlying knowledge model we built a correspondence between schemas and facilitated comparison and interoperability.

2 BACKGROUND

Interoperability of structured data has been research academically and developed commercially. However, most of these solutions are mainly manual and while it is tedious to manually compare and match across disparate schemas, automated solutions have proven difficult to develop (Lakshmanan and Sadri, 2003).

Previous research into data integration and schema matching has led to a number of different approaches. Most of these solutions involve a global schema that, similar to our solution, integrates different data sources to one universal model. As an example, the NIEM (National Information Exchange Model) developed by the United States government integrates schema from a vast variety of sources (NIEM, 2010)

A global schema however, differs from our solution in that it is not an abstraction and merely, melds many different data sources. The disadvantage, however, is that without a governing abstraction, such as the one proposed here, global schemas are manually tedious, unwieldy, difficult to manage and hard to expand.

Another approach is to start with a small core and to incrementally add domain specific data models. Dublin Core implemented in XML, for example, provides a small set of semantic elements to describe and catalog information resources (Powell and Johnston, 2010) and Electronic Business XML (ebXML), and its derivatives, were built on multiple layers from a common core (Kotok, 2001). Another United States government standard, the Universal Core (UCore), describes a small set of essential data along with the provision for domain specific enhancements (UCore, 2010).

The difficulty with this approach is the compatibility between multiple disparate extensions. These integration problems, in fact, mirror the difficulty in structure data interoperability in general.

Brock D. and Venkataramanan J..
A METHOD FOR INTEROPERABILITY BETWEEN STRUCTURED DATA SOURCES USING SEMANTIC ANALYSIS.
DOI: 10.5220/0003105402340239
In *Proceedings of the International Conference on Knowledge Management and Information Sharing* (KMIS-2010), pages 234-239
ISBN: 978-989-8425-30-0

There are commercial solutions available that facilitate the integration of XML sources. The XML Schema Mapper, for example, from Stylus Studio provides a visual development environment to quickly generate element-to-element schema mappings (Stylus Studio, 2010). The IBM schema mapping management system, Clio, derives likely mappings by analyzing the schemas and the underlying data using a Naïve-Bayes-based matching algorithm (Fagin, 2009). However these solutions lack an encompassing abstraction and sometimes do not combine semantic understanding with their matches. This is a disadvantage that requires manual input from a knowledgeable user to correct.

If we could automate this process of mapping the schema to semantically sound concept abstractions, the applications that can be produced will be faster, more efficient and more fully comprehensive.

3 CONTENT MODEL

Despite their differences, nearly all XML documents store information hierarchically and use element and attribute names derived from human language. So it should be possible to discover the intended meaning of an XML schema using semantic analysis, so that the XML documents on which they are based may be understood and integrated.

Our approach to an abstraction was to transform disparate representations into a neutral format using a common data structure and a shared vocabulary. Relying on the hierarchical nature of XML the generic content model consisted of a hierarchical organization with well-defined parent-child relations. And to enforce this structure we developed and implemented a common vocabulary. The hierarchical model and the dictionary are the two elements of a generic model, which forms the basis for the integration of different schema.

3.1 Data Model

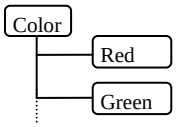
XML languages are composed of a nested structure of tags (W3C, 2008) and therefore a hierarchy is an obvious choice for a basic data model. In lieu of tags XML can also be represented as a hierarchy of concepts. The model defines ontologically the expected relationships between a parent and child tag.

A concept represents one discrete idea behind the tags. To simplify this hierarchy and make it easier to process we have restricted these concepts to be a single word or a compound phrase.

The parent-child relationship in the data model maps almost directly to the relationship between tags within the XML hierarchy. Within the data model hierarchy the parent-child relationship can be one of three kinds:

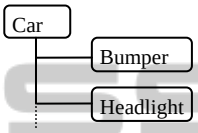
- ‘type-of’ relationship – The parent concept is a generalization of the child node

```
<Color>
    <Red>
    <Green>
     ...
```



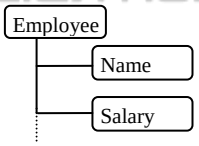
- ‘part-of’ relationship – The child is a subset of the parent.

```
<Car>
    <Bumper>
    <Headlight>
     ...
```



- ‘has-a’ relationship – The child is an attribute of the parents’.

```
<Employee>
    <Name>
    <Salary>
```



This information can be derived from the structure of the XML tags and used to constructing a generic, hierarchy representation of the data.

3.2 Common Vocabulary

In order to harmonize disparate data representations, a common set of terms was required. This set of terms served as a ‘target’ vocabulary into which proprietary terminology could be mapped. Furthermore, this vocabulary contained unambiguous definitions, so that any particular entry would have one and only one meaning.

Our first dictionary, originally derived from WordNet, was modified by assigning numeric extensions to unique definitions (WordNet, 2010). With over 200,000 entries, the dictionary provided a comprehensive base for semantically representing schema elements.

A number of issues, however, emerged from this initial implementation. Firstly, the dictionary entries were not organized by frequency or domain of use. Thus common words mingled with domain specific terms. Secondly, noun phrases, such as ‘Asian country’ could not participate in the ontological hierarchies. Finally, we could not easily accommodate the many proprietary and domain

specific dictionaries.

Therefore, we developed a new dictionary organized by usage frequency subsets and domain of use. Words were gleaned from various sources including children's literature and Simple English articles for the core vocabulary, news articles and Wikipedia entries for common terms and domain specific terms from engineering, science and governmental sources.

Ontological relationships in this dictionary included connections both within and across vocabulary subsets. In the example shown in Fig. 1, the words `frog` and `animal` were located within the same subset, and were linked by a simple *typeof* relation. Another subset containing an expanded vocabulary included the word `amphibian`, which was linked the words in the first subset. Finally, a domain specific vocabulary, in this case scientific classification terminology, linked terms both within and across subsets using *typeof*, *partof* and *attributeof.*

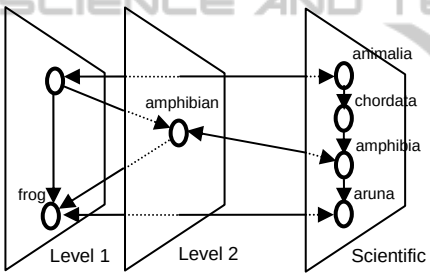


Figure 1: The common dictionary segregated words into varying levels of complexity as well as different usage domains. Ontological relations linked words across multiple word sets.

4 SCHEMA MAPPING

XML schemas from different sources typically have different data models, structures, vocabularies and formats. The model defined in Section 3 transforms disparate xml schemas into a generic model that tries to appease these differences. To reduce the complex structure of XML schema to this form we implemented an approach to translate tag names into words or word phrases formed from the shared vocabulary. Then, we simplified the schema using reduction and normalization techniques. And finally, we compared the resulting schema and identified regions of overlapping data representations.

4.1 Tokenization

Naming conventions within XML documents run a full gamut from single words to complex phrases composed of words, acronyms, abbreviations and prepositions, as shown in Figure 2. The first phase in schema interoperation was to decompose tag names into discrete tokens.

In order to accomplish this task, we first 'dereference' the XML schema. XML Schema often import subsets of vocabulary from other schema to be used in conjunction with natively developed terms. This heterogeneous mix of phrases provides the basis for data representation.

Tag names assume a variety of forms, again shown in Figure 2. Capital and camel case notation is often used in conjunction with underscore formats, and all of these employing various combinations of words, acronyms, abbreviations and prepositions.

The basic tokenization algorithm is quite simple. We tokenized based on underscore characters (if any) and case transitions. In practice, however, tokenization rules are not strictly followed by schema designers. Thus we were therefore required to validate tokens against concept maps, which are discussed in the next section.

```
AFCT_EQPT_CODE_TYPE
SSN
HasDestinationOf
IVrate
```

Figure 2: Schema tags cover a wide range of formats from simple words to complex phrases containing words, acronyms, abbreviations and prepositions.

4.2 Concept Mapping

Once tokens from the schema were determined, they were mapped to terms in the common dictionary. We executed this mapping in multiple phases. Initially, we specify the domain from which the schema originates, such as 'medical', 'business', 'governmental', etc. Future algorithms may automatically derive context from the token corpus.

Secondly, schema tokens were matched if they were identical to words in the common dictionary assuming the same context. If a match was not found, we attempt to match against know abbreviations. In practice, this is a common method for generating tokens for an XML tag name. If neither a word nor abbreviation were found, common acronyms from the appropriate domain were tested.

If a match still was not found, our algorithm extracts the tag documentation (if any) and generated permutations of character strings formed from key words within the text. Specifically, we formed string combinations using words, syllables, syllables with vowels removed and first letters. Many schemas in fact used such methods for tag name generations.

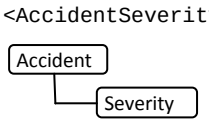
If all the above techniques failed to assign a token to one or more words in the common dictionary, the system reverted to manual intervention. In practice few such words were encountered.

```
AFCT_EQPT_CODE_TYPE
  └► aircraft equipment code type
SSN
  └► social_security_number
HasDestinationOf
  └► has destination of
IVrate
  └► intravenous rate
```

Figure 3: XML tag names were tokenized and mapped to words or phrases from the common dictionary by matching words, abbreviations, acronyms or synthetic strings formed from word fragments.

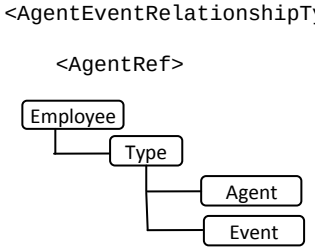
4.3 Tag Phrases

After converting tags to a normalized form we can start to fold it into the concept model we defined by utilizing the dictionary. In the model proposed, the hierarchy was composed of simple or compound words. However most XML tags consist of word phrases, such as `<AccidentSeverityCode>`, which represented a hierarchy of concepts



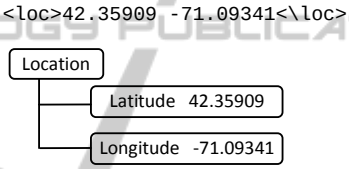
The above example was easy to place in a hierarchy since `severity` is an attribute of the accident. The word ‘code’ in this instance implied the data value was an enumeration. In the context of the concept hierarchy this was not relevant, but in the translation of data value will be critical.

Some tag phrases described relationships. For example `<AgentEventRelationshipType>` described the relationship between an agent and an event. The model shown below illustrates the difficulty in automatic conversion.



Automating the creation of a hierarchy had particular challenges. Many words were irrelevant and were eliminated. Extraneous words were identified and removed while maintaining the integrity of the concept hierarchy as in (Heflin and Hendler, 2000). In our implementation, any tokens not critical to the semantic description of the data were eliminated.

Schemas typically provided different levels of resolution. In the example below, `<loc>` implied ‘latitude’ and ‘longitude,’ but were not explicitly stated in the schema



Our current implementation did not examine instance data together with the schema analysis. Such an approach could correlation through the normalization of information resolution. One of challenges in automatically constructing the hierarchy was the properly ordering the concepts from the word phrase. Parent-child relationships were defined as one of three types ‘type-of’, ‘part-of’ and ‘has-a’. Thus once the words in the tag were separated and redundancies eliminated, the word order was determined by testing sequences against the proposed parent-child relations and by comparing to the stored word associations in the dictionary the correct hierarchical relationships can be validated.

4.4 Schema Reduction

Once the schema elements were mapped to phrases using terms from the common dictionary, we performed a number of additional steps to reduce schema variability.

First, we normalized elements and attributes. In XML schema, the use of elements and attributes is somewhat arbitrary and many designers choose one over the other to describe identical data. For example, an event location described by two

different schemas was encoded as follows

```
<position lat="42.36"
       lng="-71.09"
       time="2010-06-04T14:15" />

<event>
  <point>
    <latitude>42.36</latitude>
    <longitude>-71.09</longitude>
   </point>
   <time> 2010-06-04T14:15</time>
</event>
```

Without definitive rules on the use of elements versus attributes, we converted all attributes to subordinate elements of the parent tag.

Second, synonymous terms were mapped to a single 'class' word representing a particular synonym set. For example, `point`, `location` and `position` were mapped to the single term `position`.

Third, prepositional phrases, such as `LocatedAt`, `HasDestinationOf`, and `EmployedBy` were converted to either single words or simple noun phrases, such as `position`, `destination`, and `employer`.

Finally, generic terms, such as data or information, were simply removed. The resulting simplification provided the basis for schema comparison described in the next section.

4.5 Schema Comparison

Once the tag names were converted and the structure simplified, as described above, the schemas were reconstructed using only terms from the common dictionary. The provided the foundation for comparison and data translation. Using the original examples discussed in Section 4.4, they are both transformed to identical structures as shown below.

```
<event>
  <position>
    <latitude>42.36</latitude>
    <longitude>-71.09<longitude>
  </position>
  <time>2010-06-04T14:15</time>
</event>
```

Our objective was to find common data representations among the disparate schema using our semantic analysis. As an upper bound, we first considered correlations between *any* two elements from the XML schemas. For example, 'name' from a first schema matched 'name' from a second, even though they may refer to different entities. In the following, the first instance 'name' refers to a vessel while in the second the pilot of a vessel

```
<vessel name="Calypso">

<vessel>
  <pilot>
    <name>John Smith</name>
  <pilot>
 </vessel>
```

For our upper bound, we also considered matches valid if they occurred inside tag phrases. For example, `aircraft identification` from one schema matched `identification` from another. Even with these generous assumptions, there was surprising little overlap among the various schemas tested.

As a lower bound on schema correlation, we considered a valid 'match' only if an element name and every predecessor in the element hierarchy matched. Using these two extremes, we bounded the extent of possible overlap and identified areas of correlation.

5 IMPLEMENTATION

To test the approach presented in the last section, we designed and built an automatic schema comparison tool. The desktop application and visual interface, shown in Figure 5, allowed the user to select two XML schemas – one in each panel. The tool automatically applied the techniques described in this paper and produced a quantitative measure of the upper and lower bounds of possible correlation, and identified elements of potential correspondence.

The tool allowed the user to view each step in the process and manually intervene, if desired, to adjust the mappings assigned by the algorithm. The changes were recorded and then used in subsequent executions of the program.

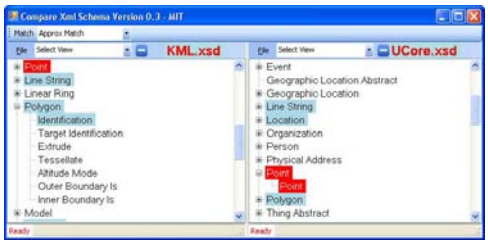


Figure 4: An automatic schema comparison tool was developed based on the schema normalization techniques described in this paper. The application read one or more XML schema files and generated comparison statistics and indicated areas of overlapping description.

6 RESULTS AND DISCUSSION

We tested the schema comparison tool on a number of governmental and commercial schemas including the Universal Core (UCore), a governmental schema for sharing digital content, Cursor on Target (CoT) a simple schema for recording geospatial positions, National Information Exchange Model (NIEM), Keyhole Markup Language (KML), Geographic Markup Language (GML) and many others.

Applying the analysis described in the previous section, the schema comparison tool identified matching elements within pairs of input schema. Considering only semantic correspondence irrespective of hierarchical position, a large quantity of prospective matches would be expected.

Relatively few matching elements, however, were actually identified, even though the schemas describe ostensibly similar data. As shown in Figure 6, the quantities represent the percent of data elements from the schema in the left-hand column contained in the schema on the top-row.

	UCI	CoT	UCore	KML
UCI		5	2	2
CoT	22		5	7
UCore	9	6		7
KML	13	9	7	

Figure 5: The percentage of overlap between disparate schemas was surprisingly small. While the general assumption was schema from similar domains only requires transformed between representations, these results suggest that related schema represent different views of similar data.

Examining the corresponding elements, at least four types of matches were identified. Firstly identical elements from identically imported schema produced obvious matches. Secondly, generic element names, such as `id`, `type` or `name` were common among disparate schema. Thirdly, generic high-level complex types, such as `unit`, `organization` or `address` were present in different schema. Finally, simple types including `latitude`, `organization` or `last_name` were found across multiple schemas.

The key result is the identification of high-level and data value elements that represent ‘bridge’ or ‘nexus’ points between disparate data sources. In other words, these elements provide the means to link dissimilar data.

7 CONCLUSIONS

The techniques developed here attempted to transform multiple, disparate XML sources into a common concept representation while retaining the underlying information. Through this process it became clear that schema from similar domains encoded different aspects of the same data. The future objective should therefore be the assimilation disparate data into a comprehensive knowledge representation that connect these different realms of data through their sparse ‘touch’ points. Based on this research, our current effort is the development of such a knowledge representation that accrues information from many disparate sources and provides tools for data manipulation, storage and presentation.

REFERENCES

Lakshmanan, L. V., Sadri, F., 2003. Interoperability on XML Data. In *Proceedings of the 2nd International Semantic Web Conference (ICSW ’03).*

Rahn, E., Bernstein, P. A., 2001. A Survey of Approaches to Automatic Schema Matching. *Very Large Database (VLDB.*, 10(4):334-350.

National Information Exchange Modelhttp://www.niem.gov/, Accessed June 2010.

Powell, A. and Johnston, P., Guidelines for implementing Dublin Core in XML.

http://dublincore.org/documents/dc-xml-guidelines/Accessed June 2010.

Kotok, A, et al., *ebXML: The New Global Standard for Doing Business on the Internet*, Sams 1st edition, 2001.

UCore|Universal Core 2.0 http://www.ucore.gov, Accessed June 2010.

Stylus Studio, XML Schema Mapper http://www.stylusstudio.com/xsd_to_xsd.html, Accessed June 2010.

Fagin, R., et al., Clio: Schema Mapping Creation and Data Exchange, appearing in *Conceptual Modeling: Foundations and Applications*, Springer 2009.

W3C, 2008, Extensible Markup Language (XML). http://www.w3.org/TR/REC-xml/.

WordNet http://wordnet.princeton.edu/, Accessed June 2010.

Heflin, J., Hendler, J., 2000. Semantic Interoperability on the Web. In *Proc.of Extreme Markup Languages.* Graphic Communications association, 2000, pp.111-120.