Reverse-engineering of XML Schemas: A Survey*

Jakub Klímek and Martin Nečaský

XML Research Group, Department of Software Engineering
Faculty of Mathematics and Physics, Charles University in Prague
Malostranské náměstí 25, 118 00 Praha 1
The Czech Republic
{klimek, necasky}@ksi.mff.cuni.cz

Abstract. As approaches to conceptual modeling of XML data become more popular, a need arises to reverse-engineer existing schemas to the conceptual models. They make the management of XML schemas easier as well as provide means for accomplishing integration of various XML data sources. Some methods for reverse-engineering of XML schemas have been proposed and in this paper, they are compared using various criteria such as used XML schema languages, level of user involvement, number of XML schemas that can be covered by the conceptual model or support for consecutive XML schema evolution. They are also evaluated according to their potential to be used as parts of a system for management, evolution and integration of XML as a whole.

Keywords: XML, schema, reverse-engineering, conceptual modeling

1 Introduction

Today, XML [27] is a technology used in a wide variety of scenarios, from a message format used by web services [7] to storage of data in databases [4]. As the number of possible usages of XML grows, so does the need of easy management of large numbers of XML data sources and their integration. There we can use conceptual modeling of XML data. It allows a domain expert to model the problem domain independently of the implementation (XML in our case) and then create corresponding XML schemas, which are used to describe a structure of XML documents.

A common situation today is that a company uses several XML formats for various purposes and has these formats described by an XML schema. To ease the process of managing those formats and schemas as they evolve in time, the company can use a conceptual model such as [1, 2, 15, 16, 19, 23], to which the schemas would be connected. A problem usually arises when there is a new

* This work was supported in part by the Czech Science Foundation (GAČR), grant number P202/10/0573.

J. Pokorný, V. Snášel, K. Richta (Eds.): Dateso 2010, pp. 96–107, ISBN 978-80-7378-116-3.

format that should be connected to the conceptual model, as most of the conceptual models do not support this operation well enough. During the process of connecting the new format to the conceptual model, the model may need to be extended, if the format contains a concept that was not covered by the model. A special case of this problem is, when the company does not have a conceptual model at all, and wants to create it from the schemas which they already have (they extend an empty model).

Therefore, an important aspect of the approaches used for conceptual modeling of XML data is if and how we can create the model from existing XML schemas (or connect a new schema to an existing model) and once we have it, if we can use it for the management of evolution of our set of schemas. The process of creating a conceptual model from existing XML schemas is what we call *Reverse-engineering of XML schemas*.

In this paper, we compare and evaluate approaches for reverse-engineering of XML schemas according to various criteria, including their usefulness as a method that can be integrated into a system for management of evolution of XML schemas.

1.1 Outline

The rest of this paper is structured as follows. In section 2, an introduction to the frequently used techniques in this area is given. In section 3 we introduce our framework for evolution and integration of XML schemas, against which the approaches will be evaluated. Section 4 contains a description of our comparison criteria. In section 5, we describe approaches to the problem, which reverse-engineer XML schemas to various user-friendlier models. In section 6, approaches to reverse-engineering to ontologies are described. In section 7 we summarize our findings and section 8 concludes.

2 Terms

In this section we provide an introduction to basic techniques used widely in the area of reverse-engineering of XML schemas.

2.1 XML schemas

In this paper, by *XML schema language* we mean one of the XML schema languages such as DTD [27], XML Schema [28], Relax NG [6], Schematron [12] etc. We state this because sometimes an XML schema gets confused with the actual XML Schema language.

2.2 Model-Driven Architecture

Model-Driven Architecture (MDA) [17] is a general approach to modeling software systems and can be profitably applied to data modeling as well. MDA

distinguishes several types of models that are used for modeling at different levels of abstraction. For this paper, two types of models are important. A *Platform-Independent Model* (PIM) allows modeling data at the conceptual level. A PIM diagram is abstracted from a representation of the data in concrete data models such as relational or XML. A *Platform-Specific Model* (PSM) models how the data is represented in a target data model. For each target data model (such as XML), we need a special PSM that is able to capture its implementation details. A PSM diagram then models a representation of the problem domain in this particular target data model, it provides a mapping between the conceptual diagram and a target data model schema.

2.3 UML class diagrams

A large number of approaches to reverse-engineering use UML class diagrams as a PIM. Basically, it consists of *classes* representing concepts, *associations* representing relations and *attributes* of classes, representing properties of the concepts. For a more detailed description see [21, 22].

2.4 Schema matching

Most of the reverse engineering approaches use some methods of schema matching. They include string comparisons, data type compatibility measurements, structural similarity measurements and linguistic resources like thesauri and dictionaries. These methods are surveyed in detail in [26, 10]. There is one major difference between XML schema matching and reverse-engineering of XML schemas to conceptual models. XML schema matching usually works with two different XML schemas (written in XML schema languages) and the goal is to find mappings of components of one schema to the components of the second schema. On the other hand, reverse-engineering of XML schemas works with one XML schema and optionally a conceptual model. The goal is either to create the conceptual model when there is none, or to find appropriate mappings of the XML schema components to the components of the model, which can be written in e.g. UML, and therefore is of a whole different type.

3 Framework for evolution and integration of XML schemas

In this section, we introduce our framework for evolution and integration of XML schemas. It comprises six levels, each representing a different view of an XML system and its evolution. The framework is depicted in Figure 1. The lowest level, called *extensional level*, represents XML documents. Its parent level, called *operational level*, represents operations over XML documents, i.e. XML queries. The level above is called *schema level* and represents XML schemas that describe the structure of the XML documents.

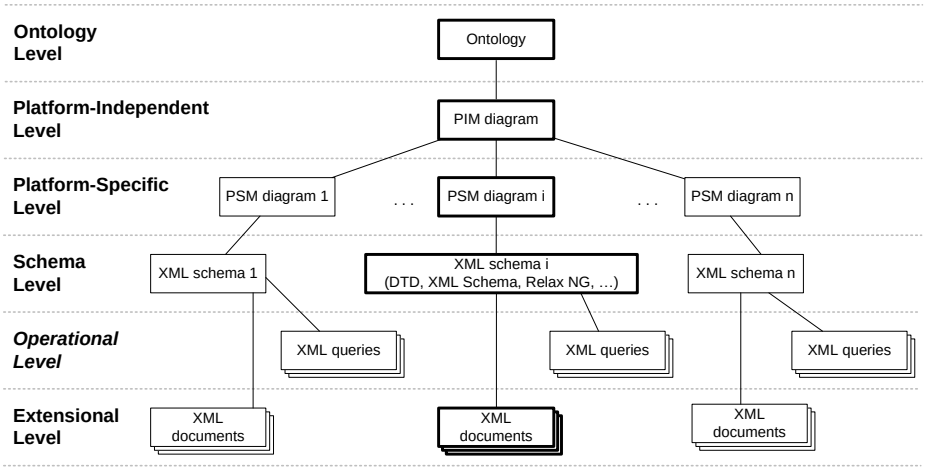

Fig. 1. Six-level XML evolution and integration framework

The platform-independent and platform-specific levels follow MDA [17] which is based on modeling the problem domain on different levels of abstraction. The *platform-independent* level represents the whole problem domain. It consists of a conceptual model that specifies the problem domain independently of its representation in the XML formats below. We call the conceptual model a *platform-independent model* (*PIM*) of the problem domain. The level below, called *platform-specific* level, represents mappings of the problem domain to particular XML formats. For each XML format it comprises a model of mapping of a selected part of the PIM to XML element and attribute declarations. We call this model *platform-specific model* (*PSM*) of the selected XML format. Recently, a number of approaches translating XML schemas to an ontology have appeared. The ontology level is the topmost in our framework.

In our framework, components in documents on individual levels can be formally binded with components in documents on the neighboring levels. These bindings can then be used for evolution of the conceptual model, XML schemas, XML documents and queries. They provide means for automatic detection of all the places on all the levels where a single change has an impact.

4 Comparison criteria

We will firstly introduce several criteria which we will later use to compare current approaches to reverse-engineering of XML schemas. In particular, we will focus on the following criteria:

1. Target model - on what level of our framework does the target model of the reverse-engineering method belong. This criterion is important, because lots of methods only visualize the XML schema in another model (e.g. UML

class diagrams) and therefore their target is on the *platform-specific* level (it is a PSM in our framework). A true conceptual model on the *platform-independent* level (a PIM in our framework) should be independent of the target implementation completely. If it is a conceptual model bent for a specific implementation, it is in fact a PSM. Recently, some approaches to mapping an XML schema directly to an ontology (the top level of our framework) have appeared. This criterion distinguishes among these three types of target models.

2. Number of schemas supported by the model - whether the method is limited to only one schema, or whether it can reverse-engineer multiple schemas to one model. This is also very important, because to be able to manage a set of XML schemas, it is not enough to have a separate model for each schema. We need all the schemas to be related to one model.
3. XML schema languages supported - DTD, XML Schema, Relax NG, Schematron, etc. As can be seen from our framework, the conceptual model can be and should be independent of the actual XML schema language used on the *schema* level, because the target data model (which a PSM should represent) is XML itself, not a specific XML schema language.
4. Mapping to an existing model - whether the method can map a schema to an already existing model or whether it can only generate a new model. This criterion is paramount for evaluating the possibility of integrating an approach to a bigger system for evolution and integration of XML schemas. If it only can create a new model for each input, it cannot be used if we already have the model and only want to add a new XML schema to the system.
5. Level of user involvement - methods can be automatic or semi-automatic (relying on human intervention). It is impossible to infer a conceptual diagram automatically, as so far only a human can determine if two objects represent the same concept. And because we need an exact and reliable match, if we want to use an approach as a part of a system for integration of XML data, we can not rely on an automatic translation (mapping) and we need the user to at least confirm a match.
6. Evolution support - whether the approach is a part of a system that also supports evolving the schema once it is integrated into the system (when the system changes). This criterion indicates, whether the method is developed by itself, or if it already is a part of a system that also helps with the evolution of the mapped schemas (e.g. by preserving the bindings between levels of our framework)

5 Approaches to mapping to user-friendly models

In this section we evaluate different approaches to reverse-engineering of XML schemas to various more user-friendly models.

5.1 Yang Weidong et al.

In [31], there is an algorithm for automatic generation of UML class diagrams (PIMs in our framework) from DTDs according to MDA using a DTD graph as a PSM. The authors also claim that they can generate UML class diagrams from XSDs, but the prototype implementation is not freely available, so it cannot be verified. The main drawback of this approach is that it only serves as an automatic translator from DTD to UML meant to make the schema more understandable to people who do not know DTD or XML Schema.

This approach does not support mapping of multiple XML schemas to the PIM and it does not preserve any mappings between the PIM and the PSM nor between the PSM and the DTD. It is automatic, limited to DTD only and it cannot map a schema to an existing model. The bright side is that it actually uses both PIM and PSM correctly.

5.2 Mikael R. Jensen et al.

In [13], another method of automatic conversion of DTDs to UML class diagrams is presented. Again, it is meant for easier browsing of XML data available on the Internet and to ease the work of a data integrator. In contrast with the previous method, the UML diagrams reflect the structure of the DTD an therefore are only on the PSM level.

This approach does not support mapping of multiple XML schemas and it does not preserve any mappings between the PSM and the DTD. It is automatic, limited to DTD only and it cannot map a schema to an existing model.

5.3 DIXSE framework

In [25], a semi-automatic method of deriving a semantic model from several DTDs is presented. By default, for each element of every DTD a new element is created in the model. This process is done automatically. If the user wants to create a more meaningful model (e.g. wants all *Address* elements to be mapped to one element of the model), a rule written in their DIXml language extending the element in the DTD must be created manually. The model is a PSM as it still preserves the DTD structure. It uses the Telos [18] metamodeling language.

This approach supports semi-automatic mapping of multiple schemas to a conceptual model, but it does not preserve any mappings between the PSM and the DTDs and it is limited to DTD only. It can map a new DTD to an existing model.

5.4 Xyleme

In [24], a project called Xyleme is described. Its focus is to provide a unified view of a large number of heterogeneous XML documents described by DTDs. This enables the user to perform queries on one unified model (called an abstract DTD),

to which all the other DTDs describing the XML documents are mapped automatically. The methods for discovery of mappings are mainly language based (thesauri, discovery of synonyms, abbreviations, etc.). A semi-automatic prototype implementation called SAMAG was used to evaluate the approach. In SAMAG, a user needs to validate each syntactic relationship detected.

This approach supports semi-automatic mapping of multiple schemas to a model which is a PSM. It preserves no mapping between the PSM and the DTDs. It is limited to DTD only.

5.5 XTM - XML Tree Model

In [9], a conceptual model for XML called XTM - XML Tree Model is proposed, including an algorithm for reverse-engineering of XML Schema into XTM. It also has a strong theoretical background. However, it is still only a PSM in our framework.

This approach automatically visualizes one schema at a time, is limited to XML Schema and does not maintain any mappings between the PSM and the schemas.

5.6 Nečaský

In [20], a complex approach to the process of reverse engineering of XML schemas to a conceptual model is presented. The model follows MDA as it uses UML class diagrams as a PIM and their extension as PSMs. It is further described in [19]. Because the model has two levels, the process is divided into two parts. The first part is an automatic translation of an XML schema to a PSM. The PSMs are, however, independent of any specific XML schema language; the approach is presented using XML Schema. The second part is a semi-automatic algorithm for the reconstruction of mappings between the PSM and a PIM, but it has some drawbacks. The most problematic one is the computational cost which is up to m^n, where m is a maximum number of outgoing PIM associations from one PIM class and n is the number of PIM classes in the model. Therefore in practice, the algorithm will not work if the PIM diagram contains a bigger number of associations. Nevertheless, the algorithm uses maximum of information that we can get from a PSM and thus can offer the best results. An implementation in an experimental stage is available in the development version of XCase [14], which is a tool implementing the conceptual model and its evolution.

This approach supports semi-automatic reverse-engineering to PSMs and to a PIM. It supports multiple schemas, is independent of any specific XML schema language and it can also map to an existing conceptual model. It maintains mappings between the PIM and the PSM and therefore support further schema evolution.

6 Approaches to mapping to ontologies

Recently, a number of methods of reverse-engineering of XML schemas to ontologies have appeared. The main difference between a PIM and an ontology is that ontologies are more expressive, as the relations they capture can be more complex and they may even involve logical formulae. A frequent language for ontologies is OWL [30].

6.1 Canonic Conceptual Models (CCMs)

In [8], a method for integration of XML schemas into an ontology is presented. At first, each input DTD is semi-automatically transformed into a so called CCM - Canonic Conceptual Model. It combines the ER [5] and ORM [11] models. A default transformation is made and a user is then allowed to make adjustments where needed. The CCMs are PSMs in our framework. Then, each CCM is integrated (again semi-automatically - user has to verify/adjust) to form the final ontology. The mappings between the individual CCM components and the components of the ontology are preserved. Although the authors call it an ontology, it is in fact more of a conceptual diagram - a PIM in our framework, because it still is a CCM, only independent of the XML structure. This method only provides a unified view of the integrated data so far. No implementation was mentioned.

This approach supports semi-automatic mapping of multiple DTDs to corresponding PSMs and to a PIM. It is restricted to DTD only. It preserves mappings between PSMs and a PIM.

6.2 Xiao et al.

In [32], an algorithm is proposed to match given XML document elements to given ontology concepts to achieve an integrated view of multiple XML documents, when matched to the same ontology. It is based on automatic structural matching of a DTD tree to an ontology tree. However, a precondition is that a domain expert has provided a table of synonyms, i.e. a list of semantically matching strings from the DTD and from the ontology (which seems to be a strong precondition).

This in fact semi-automatic approach maps multiple DTDs, it is limited to DTD only and it does not preserve any mappings between the DTDs and the ontology. It can only map to an existing ontology.

6.3 Bedini et al.

In [3], a general architecture of building ontologies from XML schemas is presented. However, no specific methods are suggested, only a sequence of tasks that a tool for ontology building should follow and also a set of rules according to which concepts and their relations can be extracted from a XML Schema.

The general architecture takes the need for consecutive schema evolution into account. A semi-automatic prototype implementation called Janus is presented briefly. It requires human assistance for merging of similar concepts and it does not preserve any mappings between the schemas and the ontology.

This approach is semi-automatic, it is limited to XML schema and it does not preserve any mappings between the schemas and the ontology. It also only creates new ontologies.

6.4 DTD2OWL

In [29], a method of automatic translation of DTD to OWL ontology is suggested. In addition, this method transforms the actual XML documents into OWL individuals. The authors suggest that the whole web should be transformed this way. This method is pure translation of one DTD to one OWL ontology with no support for schema evolution nor conceptual modeling.

This approach is automatic, it is limited to DTD and it can only map one DTD to one new ontology, not preserving any mappings.

7 Summary

In this section, a brief summary of evaluated approaches and the comparison criteria is given.

	Model	Schemas	Languages	Automatic	Maps to	Evolution
5.1	PSM	One	DTD	Yes	New	No
5.2	PSM	One	DTD	Yes	New	No
5.3	PSM	Multiple	DTD	No	New, Existing	No
5.4	PSM	Multiple	DTD	Yes	New, Existing	No
5.5	PSM	One	XSD	Yes	New	No
5.6	PIM	Multiple	Any[a]	No	New, Existing	Yes
6.1	PIM	Multiple	DTD	No	New, Existing	Yes
6.2	O	Multiple	DTD	No	Existing	No
6.3	O	Multiple	XSD	No	New	No
6.4	O	One	DTD	Yes	New	No

[a] This method is not limited to any XML schema language. Currently implemented for XML Schema

Table 1. Overview of approaches according to various criteria

Let us review the comparison criteria used (the columns in Table 1 correspond to them):

1. Whether the target of the approach is a PIM, PSM or an ontology

2. Whether the approach is limited to only one schema or whether it can handle multiple schemas
3. Which XML schema languages can the approach handle
4. Whether the method is a pure automatic translation or whether the process is semi-automatic - a user is involved
5. Whether the method creates a new target model or whether it can use an existing one
6. Whether the method preserves mappings between the schemas and the target model, which can be used for consecutive schema evolution

The best suitable method for our intention of creating a system for management of XML schema evolution and integration is 5.6. However, it has some issues with computational complexity, which need to be resolved before its implementation.

8 Conclusion

In this paper, we have compared and evaluated several approaches for reverse-engineering of XML schemas according to given comparison criteria. Among them, only one was well suited for being a part of a larger system for evolution and integration of XML schemas. Also, this survey showed a severe lack of support for newer XML schema languages like Relax NG or Schematron in the area of conceptual modeling of XML and reverse-engineering of XML schemas.

References

1. R. Al-Kamha, D. W. Embley, and S. W. Liddle. Augmenting Traditional Conceptual Models to Accommodate XML Structural Constructs. In *Proceedings of 26th International Conference on Conceptual Modeling*, pages 518–533, Auckland, New Zealand, Nov. 2007. Springer.
2. A. Badia. Conceptual Modeling for Semistructured Data. In *Proceedings of the 3rd International Conference on Web Information Systems Engineering Workshops*, pages 170–177, Singapore, Dec. 2002. IEEE Computer Society.
3. I. Bedini, G. Gardarin, and B. Nguyen. Deriving Ontologies from XML Schema. *CoRR*, abs/1001.4901, 2010.
4. R. Bourret. XML and Databases. September 2005. `http://www.rpbourret.com/xml/XMLAndDatabases.htm`.
5. P. Chen. The Entity-Relationship Model–Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, Mar. 1976.
6. J. Clark and M. Makoto. *RELAX NG Specification*. Oasis, December 2001. `http://www.oasis-open.org/committees/relax-ng/spec-20011203.html`.
7. D. Booth, C. K. Liu. *Web Services Description Language (WSDL) Version 2.0 Part 0: Primer*. W3C, June 2007. `http://www.w3.org/TR/wsdl20-primer/`.
8. R. dos Santos Mello and C. A. Heuser. A Bottom-Up Approach for Integration of XML Sources. In *Workshop on Information Integration on the Web*, pages 118–124, 2001.
9. J. Fong, S. K. Cheung, and H. Shiu. The XML Tree Model - toward an XML conceptual schema reversed from XML Schema Definition. *Data Knowl. Eng.*, 64(3):624–661, 2008.

10. H. Hai, Do. *Schema Matching and Mapping-based Data Integration: Architecture, Approaches and Evaluation*. VDM Verlag, Saarbrücken, Germany, Germany, 2007.
11. T. Halpin and T. Morgan. *Information Modeling and Relational Databases*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.
12. ISO. *Information Technology Document Schema Definition Languages (DSDL) Part 3: Rule-based Validation Schematron. ISO/IEC 19757-3*, feb 2005.
13. M. R. Jensen, T. H. Møller, and T. B. Pedersen. Converting XML Data to UML Diagrams For Conceptual Data Integration. In *In: 1st International Workshop on Data Integration over the Web (DIWeb) at 13th Conference on Advanced Information Systems Engineering (CAISE01)*, 2001.
14. J. Klímek, L. Kopenec, P. Loupal, and J. Malý. XCase - A Tool for Conceptual XML Data Modeling. In *Advances in Databases and Information Systems*, volume 5968/2010 of *Lecture Notes in Computer Science*, pages 96–103. Springer Berlin / Heidelberg, March 2010.
15. B. Loscio, A. Salgado, and L. Galvao. Conceptual Modeling of XML Schemas. In *Proceedings of the Fifth ACM CIKM International Workshop on Web Information and Data Management*, pages 102–105, New Orleans, USA, Nov. 2003.
16. M. Mani. Erex: A conceptual model for xml. In *Proceedings of the Second International XML Database Symposium*, pages 128–142, Toronto, Canada, Aug. 2004.
17. J. Miller and J. Mukerji. *MDA Guide Version 1.0.1*. Object Management Group, 2003. http://www.omg.org/docs/omg/03-06-01.pdf.
18. J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: representing knowledge about information systems. *ACM Trans. Inf. Syst.*, 8(4):325–362, 1990.
19. M. Nečaský. *Conceptual Modeling for XML*, volume 99 of *Dissertations in Database and Information Systems Series*. IOS Press/AKA Verlag, January 2009.
20. M. Nečaský. Reverse Engineering of XML Schemas to Conceptual Diagrams. In *Proceedings of The Sixth Asia-Pacific Conference on Conceptual Modelling*, pages 117–128, Wellington, New Zealand, January 2009. Australian Computer Society.
21. Object Management Group. *UML Infrastructure Specification 2.1.2*, nov 2007. http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF/.
22. Object Management Group. *UML Superstructure Specification 2.1.2*, nov 2007. http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF/.
23. G. Psaila. ERX: A Conceptual Model for XML Documents. In *Proceedings of the 2000 ACM Symposium on Applied Computing*, pages 898–903, Como, Italy, Mar. 2000. ACM.
24. C. Reynaud, J.-P. Sirot, and D. Vodislav. Semantic integration of xml heterogeneous data sources. In *IDEAS '01: Proceedings of the International Database Engineering & Applications Symposium*, pages 199–208, Washington, DC, USA, 2001. IEEE Computer Society.
25. P. Rodríguez-Gianolli and J. Mylopoulos. A Semantic Approach to XML-based Data Integration. In *ER '01: Proceedings of the 20th International Conference on Conceptual Modeling*, pages 117–132, London, UK, 2001. Springer-Verlag.
26. P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics*, 4:146–171, 2005.
27. T. Bray and J. Paoli and C. M. Sperberg-McQueen and E. Maler and F. Yergeau. *Extensible Markup Language (XML) 1.0 (Fourth Edition)*. W3C, September 2006. `http://www.w3.org/TR/REC-xml/`.
28. H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. *XML Schema Part 1: Structures (Second Edition)*. W3C, October 2004. `http://www.w3.org/TR/xmlschema-1/`.

29. P. T. T. Thuy, Y.-K. Lee, and S. Lee. DTD2OWL: Automatic Transforming XML Documents into OWL Ontology. In *ICIS '09: Proceedings of the 2nd International Conference on Interaction Sciences*, pages 125–131, New York, NY, USA, 2009. ACM.
30. W3C OWL Working Group. *OWL 2 Web Ontology Language*. W3C, October 2009. `http://www.w3.org/TR/owl2-overview/`.
31. Y. Weidong, G. Ning, and S. Baile. Reverse Engineering XML. *Computer and Computational Sciences, International Multi-Symposiums on*, 2:447–454, 2006.
32. L. Xiao, L. Zhang, G. Huang, and B. Shi. Automatic Mapping from XML Documents to Ontologies. In *CIT '04: Proceedings of the The Fourth International Conference on Computer and Information Technology*, pages 321–325, Washington, DC, USA, 2004. IEEE Computer Society.