International Journal of Internet, Broadcasting and Communication Vol.9 No.2 11-19 (2017)
https://doi.org/10.7236/IJIBC.2017.9.2.11

IJIBC 17-2-2

An Effective XML Schema Conversion Technique for Improving XML Document Reusability using Pattern List

Hye-Kyeong Ko[1*], Minho Yang[2]

[1]*Department of Computer Engineering, Sungkyul University, Korea*
[2]*Division of East Asian Studies & Logistics, Sungkyul University, Korea*
[*]*hkko@sungkyul.ac.kr, minhoyang@hanmail.net*

Abstract

The growing use of XML markup language has made amount of heterogeneous. XML documents are widely available in the Web. As the number of applications that utilize heterogeneous XML documents grow, the importance of XML document extraction increases greatly. In this paper, we propose a XML schema conversion technique that converts reusable XML schema from XML documents. We convert the schema graph and we use the reusability pattern list. The converted XML schema is evaluated in terms of cohesion, coupling, and reusability. The converted XML schema could be used to construct databases for various fields where XML is used as an intermediation of data exchange.

Keywords: *XML schema conversion, Reusability, Schema graph, Pattern list*

1. Introduction

Today the eXtensible Markup Language (XML) promises to enable a suite of nest generation Web applications ranging from intelligent web searching to electronic commerce [1]. XML data, like semi-structured data, is hierarchically structured data models is the notion of a Document Type Definition (DTD) that may optionally accompany an XML document. Unfortunately, DTD capabilities appear to be limited for many application domains. Most important limitations of DTDs are no XML syntax, a limited set of the datatypes and loose structured constraints [2-4]. As a result, the new XML schema has been recently proposed to replace DTDs. Another key feature of XML schema language supports inheritance, so we should improve XML documents reusability. We propose a XML schema conversion technique by deriving features from existing schemas. Inheritance enables efficient software reuse and help developers avoid building everything from scratch again and again.

The syntactical and functional capabilities afforded by W3C XML schemas provide tremendous support and opportunities for metadata reusability. The problem of extracting XML schema from a collection of

Manuscript Received: Mar. 10, 2017 / Revised: Mar. 20, 2017 / Accepted: Apr. 5, 2017
Corresponding Author: hkko@sungkyul.ac.kr
Tel: +82-31-467-8113
Department of Computer Engineering, Sungkyul University

XML document is, to the best of our knowledge, has not been previously addressed in the literature. The current researches should not consider conversion of data format and reusability of schema. Our work different from the current XML schema conversion systems, we do follow the reusability pattern that extracts XML Schema for improving reusability.

The XML schema conversion system makes use of the OEM [7] and the DataGuides [8] which extracts schemas from semi-structured data. We convert XML schema for improving reusability that utilizes 'Pattern List' which uses the Venetian Blind Model [18] and the Chameleon Namespace design [19].

The rest of this paper is organized as follows. Section 2 reviews related works that describe the current XML schema extraction systems and Section 3 presents the proposed XML schema conversion system for improving reusability and 4 describes XML schema conversion system's design and implementation in detail and Section 4 includes experimental results. Finally, Section 5 concludes this paper.

2. Related works

In this section, we present the model of semi-structured data and the types. We present the model of semi-structured data which extraction technique used in the current paper. Assuming a graph model for semi-structured data to find a typing is equal to group objects that have similarly labeled edges [7]. The nodes in the graph represent objects and labels on the edges convey semantic information about the relationships between objects. The sink node in the graph represent atomic objects and have values associated with them. The DataGudes is itself a graph, where every label path in the database appears exactly once in the DataGuides and very label path in the DataGuides exists in the database [8].

Semi-structured data is characterized by the lack of any fixed and rigid schema, although typically the data has some implicit structure. While the lack of fixed schema makes which extracts semi-structured data fairly easy and an attractive goal, presenting and querying such data is greatly impaired. Thus, a critical problem is the discovery of the structure implicit in semi-structured data. The schemas presented in this paper can be depicted in a directed graph as data are depicted. To achieve conciseness, Lore specifies that the DataGuides describe every unique label path of data instance exactly once, regardless of the number of times it appears in those data instance. We utilize the MDL principle [11, 12] to define an information theoretic measure for quantifying and thereby resolving the tradeoff between the conciseness and preciseness of DTDs.

The most popular XML schema extraction systems are XML schema generator, XML spy. The XML schema generator [13] uses the dataset class of .Net as a model and it converts XML schema on Web and it utilized the Separate Complex Type [13] which applies the Russian Doll Design or the Salami Slice Design [18]. Though XML schema generator provides the primitive datatypes, however it completely should not provide the characteristics of XML schema in the dataset class. The characteristics of XML spy [14] developed in ALTOVA which it is an efficient editing of any schema type or DTD. However, the code should not generate the optimization code and it should not consider schema reusability.

3. An XML schema conversion technique for improving reusability

In this section, we present the proposed XML schema conversion technique. The reusability presents one of the greatest opportunities for reducing application development costs, engineering cost.

3.1 Converting XML schema from semi-structured data

We describe an XML schema conversion technique from semi-structured data in Figure 1. By using the OEM and the DataGuides, the procedure extracts both the upper and lower bound schema, and it extracts both the absolute and relative schema in terms of the schema graph. The DataGuides can check whether data source has the given label path that the length *n* and it extracts the lower bound schema which improves the maximum fixed point method.

We use the schema conversion technique of semi-structured data and we generate the upper bound schema make use of the DataGuides and we generate the lower bound schema by simulation. To convert the schema graph, the absolute/relative schema compare information of the label duplicate and it extracts both the occurrence (‘*’, ‘+’) and (‘|’) operator.

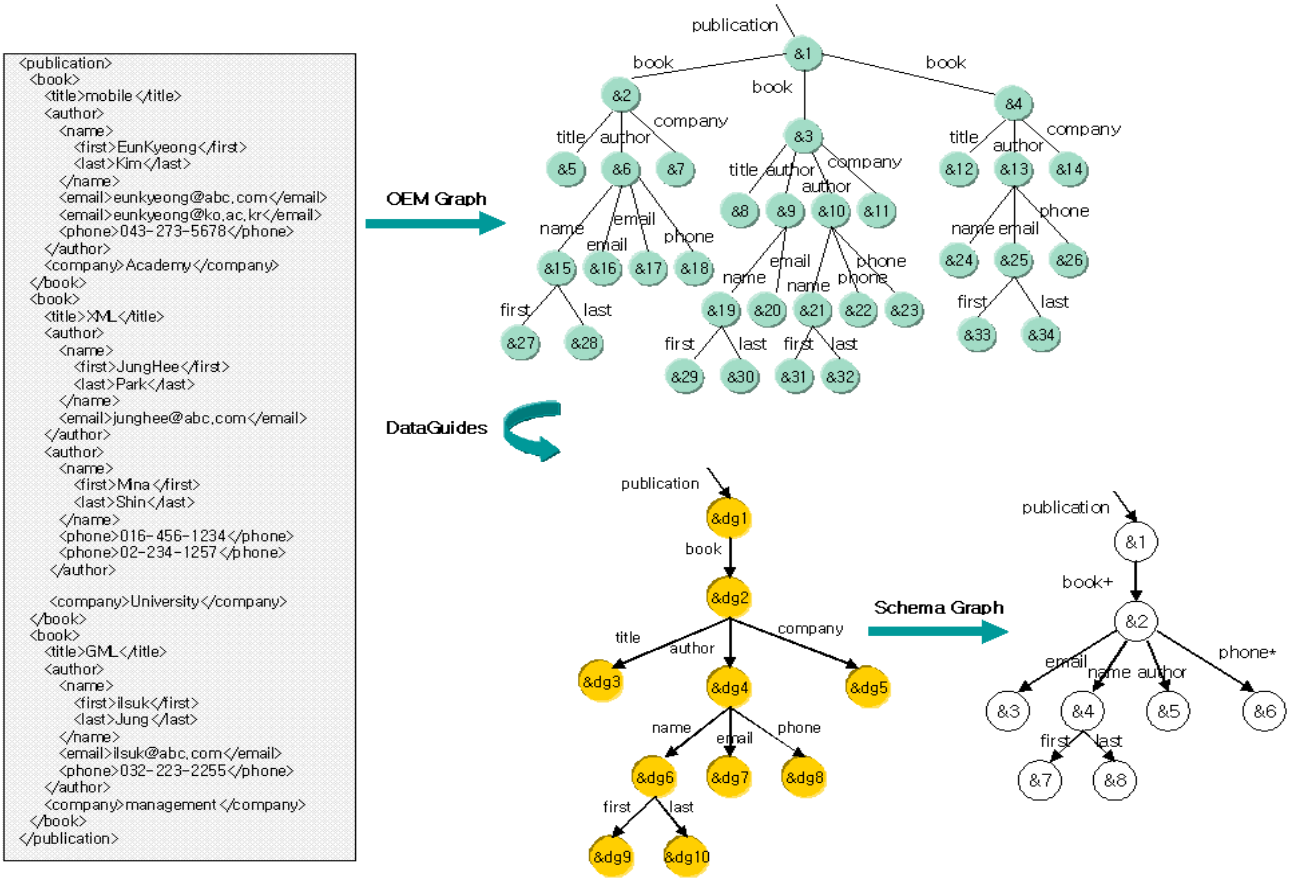

Figure 1. The conversion of schema graph

Using the occurrence indicator, we indicate how many times the element names in its content model may occur. There are three occurrence indicators such as ‘+’, ‘*’, and ‘*’ in the XML syntax. The absolute schema expresses the element which must appears in XML document, and the relative Schema may appear in XML document. To extract the relative schema, we consist of the label which compares both the extracted absolute and relative schema that label should not duplicate in the absolute schema and it consists of the label duplication table.

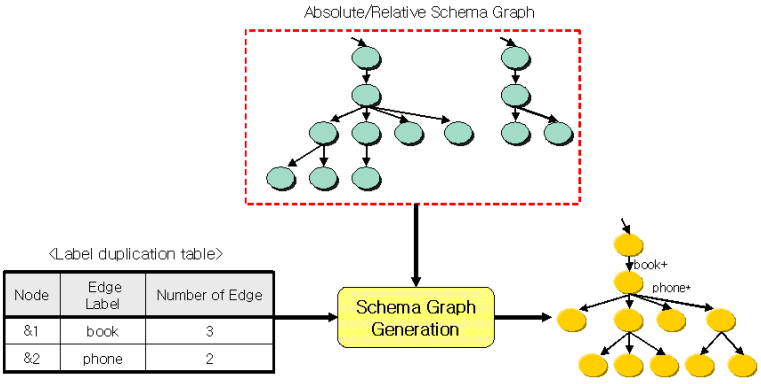

Figure 2. The process of schema graph generation

We show the process which creates the schema graph that utilized both the absolute and relative schema graph in Figure 2. We describe the reusability pattern process approach that makes cohesion which is independence of module strong to design reusability of module high and degree of coupling.

As with the software design, there are patterns associated with XML Schema design. The most popular XML schema design patterns are the Russian Doll, the Salami Slice, and the Venetian Blind Model [18]. The Russian Doll Model has a single global element that nests local element. The Salami Slice Model has all of the elements defined within the global namespace and the referencing the elements. Similar to the Russian Doll, the Venetian Blind Model has a single global element that nests local element. With this model approach we disassemble the problem into individual components, as the Salami Slice Model does, however instead of creating element declarations, we create the type definitions.

The namespaces provide a scoping mechanism for XML if same names are used across multiple schemas [19]. The Heterogeneous Namespace Design gives each schema which is a different targetNamespace, the Homogeneous Namespace Design gives all schemas which are the same targetNamespace and Chameleon Namespace Design gives the ‘main’ schema which is a targetNamespace and it ives no targetNamespace to the ‘supporting’ schemas.

Table 1. Pattern process

Symbol	Pattern Process	XML Schema Generation Pattern
?	Occurrence indicators	<element name= “root” type= “rootType” minOccurs= “0”/> <complexType name= “rootType”> <element name= “count” type= “integer”/> </complexType>
*		<element name="root" type="rootType" minOccurs="0"/> maxOccurs="unbounded"/> <complexType name="rootType"> <element name="count" type="integer"/> </complexType>
+		<element name="root" type="rootType" minOccurs="1"/> maxOccurs="unbounded"/> <complexType name="rootType">

		<element name="count" type="integer"/> </complexType>
,	Connector	<element name="root" type="rootType"> <complexType name="rootType"> <sequence> <element name="shipTo" type="string"/> <element name="billTo" type="string"/> </sequence> </complexType>
|		<element name="root" type="rootType"> <complexType name="rootType"> <choice> <element name="shipTo" type="string"/> <element name="billTo" type="string"/> </choice> </complexType>
#IMPLIED	Default Value of Attribute	<attribute name="manager" type="string" use="optional"/>
#REQUIRED		<attribute name="number" type="string" use="required"/>
#FIXED		<attribute name="office" type="string" use="fixed" value="A"/>

We will get the information of the occurrence indicators and the connector which presents the repeat frequency of element in XML schema that utilizes the schema graph. The extracted XML schema applies the pattern process which uses the Venetian Blind Model [18] and the Chameleon Namespace Design [19]. The pattern process presents in Table 1.

3.2 The proposed XML Schema conversion system

We describe the overall system modules of XML schema conversion systems in Figure 3. The system converts XML schema for improving reusability. The modules of system consist of the OEM graph generator, the DataGuides generator, the absolute/relative schema generator, the schema graph generator, and the reusability pattern processor in Figure 3.

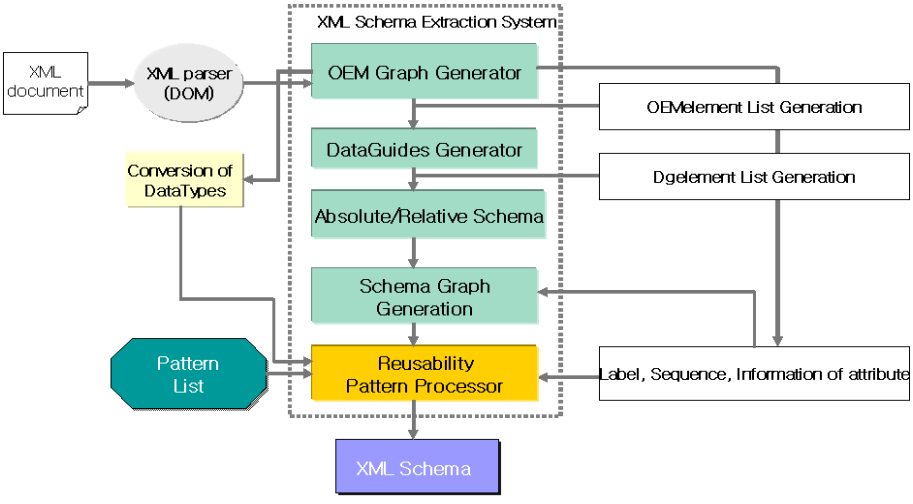

Figure 3. The modules of XML schema conversion system

- OEM graph generator : The OEM graph generator expresses the object list that is uses the OEM modeling which creates each element of OEM graph by OEM element list. The data structure of OEMelement list describes in Table 2.

Table 2. The data structure of OEMelement list

Type	Name	Contents
int	Oid	element characteristic ID
string	Ename	element name
boolean	Hasccontent	atomic object (true/false)
string	Value	data value of atomic object
vector	Edges	list of child element
binary tree	Attribute	attribute list of element

- DataGuides generator : Each object of DataGuides creates the Dgelement list that utilized the DataGuides algorithm to convert structure information for XML document from the OEM element list. The DGdlement list describes in Table 3.

Table 3. Data structure of Dgelement list

Type	Name	Contents
int	DGoid	Characteristic ID of DataGuides element
vector	Oids	Set of OEMelement id
vector	Edgelabel	Label name for Sub Dgelement list, edge list

- Reusability pattern processor : We extract XML schema that uses 'Pattern list' which utilizes the attribute table, the connector table, the schema graph, and the conversion of datatypes table. The reusability pattern processor shows in Figure 4.

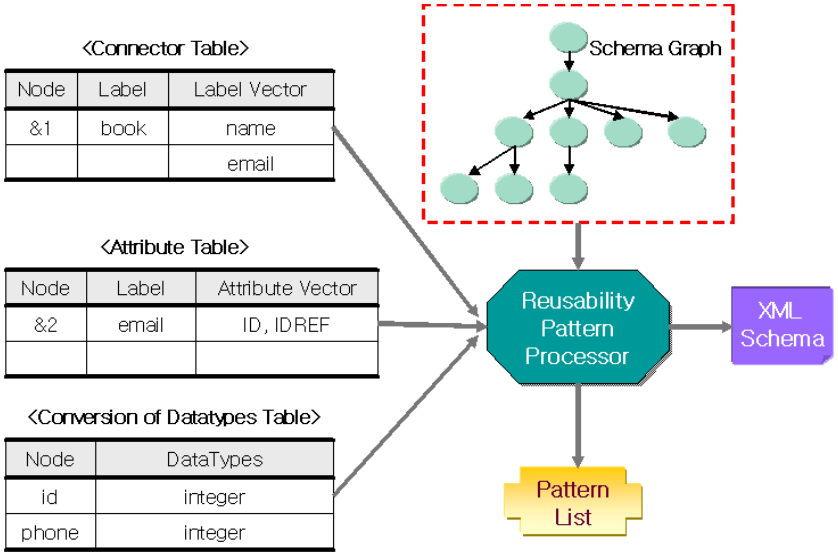

Figure 4. Reusability pattern processor

The processor changes the datatypes that the information defines each attribute of element in the attribute table and the conversion of datatypes table includes the information of occurrence indicators and connector in the schema graph.

4. Performance evaluation

We have implemented XML schema conversion system in Java [20], which converts XML schema for improving reusability. We evaluate the extracted XML schema that we compare to the current conversion systems such as the XML schema generator [13].

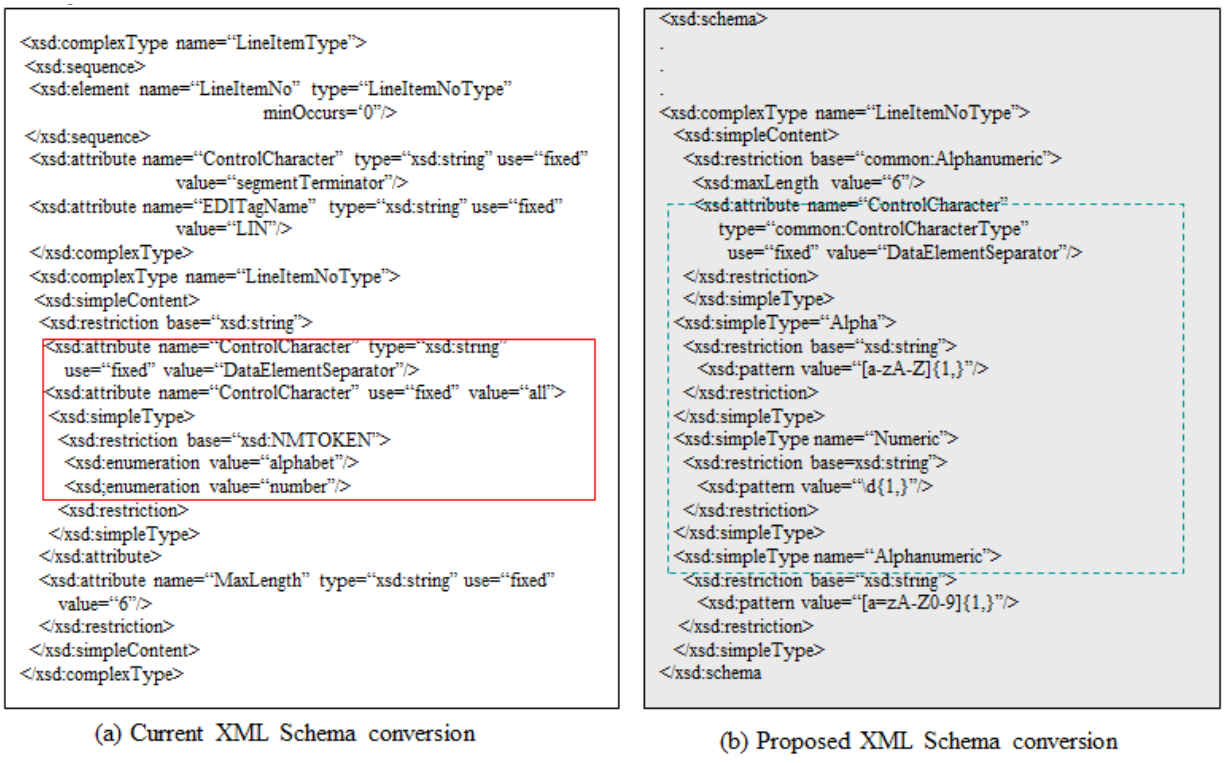

Figure 5. Reusability pattern processor

When we observe the 'ControlCharacter' attribute in Figure 5, the part shown in the solid line is the results of the current schema conversion system, and the dotted line is results of the proposed schema conversion system. However, it will change the string as 'all', 'alphabet', and 'number' in the solid line. Therefore, it makes the datatypes which is the 'AlphaNumeric' and the 'MaxLength' must change the 'maxLength' which is a facet of the string that defines from the W3C. The dotted line defined the primitive datatypes as 'Alpha', Numeric', and 'Alphanumeric. Moreover, it changed 'ControlCharacterType'. The converted XML Schema defined in one schema file with the namespace and it improved XML Schema reusability.

The main goal of design is to minimize coupling and maximize cohesion. The coupling is the level of interdependency between a method and the environment, while cohesion is the level of uniformity of the method goal. The proposed XML Schema evaluation uses the scale of cohesion and the scale of coupling in Table 4.

Table 4. Evaluation of XML schema conversion system

XML Schema Declaration Elements	Cohesion		Coupling	
	Level	Grade	Level	Grade
Local declaration of child element	Procedural	6		
Reference declaration of child element	Procedural	6		
Declaration of local attribute			Stamp	2
Declaration of attribute group	Procedural	6		
Use of naming model group	Functional	7	Stamp	3
Use of ComplexType	Functional	7		
Evaluation of Reusability	Total	32	Total	5

According to the evaluation results of XML schema declaration elements, cohesion is a total of 32 when the total are divided the declaration element. A coupling is a total of 5 when the total are divided the declaration elements, a coupling averaged 2.5. Therefore, a coupling evaluated middle. As a result, the converted XML schema designed to minimize coupling a maximize cohesion.

5. Conclusion

The XML is a standard for exchange and manipulation of structured documents. The structure of XML documents is often described by DTDs. Unfortunately; DTD capabilities appear to be limited for many application domains. Most important limitations of DTDs are a non XML syntax, a limited set of datatypes and loose structured constraints. As a result, new XML Schema language has been recently proposed to replace DTDs. These languages extend the DTD model with novel important features, such as simple and complex types, rich the datatype sets, occurrence constraints and inheritance.

As the XML Schema language is widely accepted as the DTD successor, we align our approach with this language. In this paper, we propose a novel XML Schema conversion technique that converted reusable XML Schema from XML documents. The converted XML Schema applied reusability pattern process that improved reusability after converting XML Schema in the schema graph. The converted XML Schema designed to minimize coupling and maximize cohesion and it selected the Complex Type which is differentiated from the Russian Doll Model of .Net. The converted XML Schema could be used to construct databases for various fields XML is used as an intermediation for data exchange and transform document generated on XML to another document format.

Acknowledgement

This work was supported by Basic Science Research Program through the National Research Foundation on Korea (KRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014R1A1A3051552), South Korea.

References

[1] T.Bray, J. Paoli, and C. Msperberg-McQueen, Extensible Markup Language (XML)1.0. *http://www.w3.org/TR/REC-XML W3C Recommendation*.

[2] W3C. XML SchemaPart 0: Primer. *http://www.w3.org/TR/xmlschema-0.*
[3] W3C. XML SchemaPart 1: Structures. *http://www.w3.org/TR/xmlschema-1*.
[4] W3C. XML SchemaPart 2: Datatypes. *http://www.w3.org/TR/xmlschema-2*.
[5] A. Deutsch, M. F. Fernandez, and D. Suciu, "Storing Semistructured Data with STORED,". in Proc. ACM SIGMOD, pp. 431-442, 1999.
[6] S. Abiteboul, P. Bunneman, and D. Suciu, "Data on the Web:From Relations to Semistructured Data and XML," Morgan Kaufmann , 1999.
[7] R. Goldman, J. McHugh, and J. Widom, "Form Semistructured Data to XML:Migrating the Lore Data Model and Query Language," in Proc. Workshop on the Web and Databases (WebDB), pp. 25-30, 1999.
[8] R. Goldman and J. Widom, "DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases," in Proc. VLDB, pp. 436-445, 1998.
[9] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom, "Integration and Accessing Heterogeneous Information Sources in TSIMMIS," in Proc. AAAI Symposium on Information Gathering, pp. 61-64, 1995.
[10] M. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K.Shim, "XTRACT: A System for Extracting Document Type Descriptors from XML Documents," in Proc. SIGMOD, pp. 165-176, 2000.
[11] A. Brazma, "Efficient Identification of Regular Expressions from Representative Examples," in Proc. ACM SIGACT, pp. 236-242, 1993.
[12] P. Kilpelainen, H. Mannila, and E. Ukkonen, "MDL Learning of Unions of Simple Pattern Languages from Positive Examples," in Proc. Euro COLT, pp. 252-260, 1995.
[13] XML for ASP.NET Developers. XSD Schema Generator. *http://www.xmlforasp.net*.
[14] ALTOVA Corporation. XML Spy. *http://www.xmlspy.com*.
[15] H. Meyerding, "XML and Content Reuse Systems for Instructional Design," E-Learning, 2004.
[16] C. Pelze and J. Murray, "Using XML Schema Effectively in WDSL Design," in Proc. Software Development Conference and Expo, 2004.
[17] R.Glushko and K. Mcgrath, "Patterns and Reuse in Document Engineering," in Proc. Deepx, pp. 1-11, 2002.
[18] Integration Architecture. *http://www.pervasive.com/solutions/integration*.
[19] MedBiguitous Consortium. XML Schema Design Guideline. *http://www.RenderX.com*.
[20] Java. *http://java.sun.com/j2se/1.3/docs/api./*