Semantic Similarity Analysis of XML Schema Using Grid Computing

Jaewook Kim[1,2], Sookyoung Lee[1], Milton Halem[1] and Yun Peng[1]
[1]Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
[2]National Institute of Standards and Technology
jaewook@nist.gov, { slee22, halem and ypeng}@umbc.edu

Abstract

A growing list of e-businesses has been using XML schemas in recent years. Schema mapping now plays a crucial role in integrating heterogeneous e-business applications. Since large-scale XML schema mapping using complex and hybrid similarity measures requires significant amount of processing time, a sophisticated similarity analysis algorithm is needed to handle its complexity and performance. In this paper, we focus on designing a service-oriented architecture (SoA) for schema mapping, based on a grid computing technology in order to enhance the effectiveness of the mapping algorithm. After comparing three different grid computing technologies (MPJ, Hadoop, and Globus), we explain why MPJ is the most suitable. We propose SoA XML schema mapping based on MPJ, and demonstrate its performance.

Keywords: XML Schema, E-business Integration, Schema Mapping, Grid Computing, Service-oriented Architecture

1. Introduction

Recently, many XML schemas have been developed for various e-business applications. Since companies engaged in e-business often define their own XML schemas for business-dependant properties, schema mapping is essential for e-business integration.

However, XML schema mapping is typically very labor-intensive, costly, and error-prone [1] . The time complexity of XML schema mapping largely depends on three factors: the number of elements that an XML schema defines, the structural complexity of each element, and the complexity of the mapping algorithm. Recently, more complicated XML schemas and highly complex mapping algorithms have been introduced. For instance, the Open Application Group's Integration (OAGi) [7] component schema and Human Resource XML (HR-XML) component schema [33] can vary from three hundred to a thousand top-level elements, respectively. Moreover, it can take more than an hour to analyze the mapping using a hybrid similarity algorithm [2]. Clearly, more-efficient architectures are required to improve the performance of XML schema mappings.

In this paper, we propose an efficient XML schema mapping analysis architecture using a Java-based parallel system called MPJ (MPI-like Message Passing for Java) [3]. The main idea of the proposed architecture is to distribute the heavy workload of XML schema mapping analysis among a cluster of processors connected using grid computing based on service-oriented architecture (SoA). The proposed architecture has two key advantages: First, it enhances the efficiency of the XML schema-similarity analysis by reducing its total execution time, and second, it provides high extensibility, so that an end-user can add additional similarity algorithm easily.

The paper is organized as follows. Section 2 provides the background for schema mappings, including brief reviews of selected similarity metrics as well as grid computing technologies. Detailed descriptions of the proposed approach are given in Sections 3. Section 4 discusses the performance of the proposed architecture. Finally, Section 5 concludes with lessons learned and directions for future research.

2. Background

XML schema mapping plays a crucial role in integrating heterogeneous e-business applications. However, the difficulty associated with this approach lies in the fact that semantics of XML schema used for e-business are not formally defined, but are implicitly embedded in the meanings of English words or phrases appearing in the names of the schemas' components and fields, as well as in associated descriptions. Precise understanding of these descriptions is difficult because of, among other things, the lack of clearly documented common approaches to associating and specifying descriptions. For these reasons, it is very costly for experts to identify the reusable standard components that can be shared by other schemas, and to understand how to use them.

Various approaches have recently been developed to automate the process of schema mapping between two schemas [4, 5]. Most of these approaches first attempt to identify semantic relations between the elements of the two schemas and assign measures of meaning similarity. In the following subsection, we review different semantic similarity measures for XML schema mapping.

2.1. Similarity measures

There are several similarity approaches to help schema mapping solutions. Based on our survey of the field of schema mapping [1, 4], the simplest approach is usually a linguistic metric that computes similarity between names and textual descriptions of schema elements. A common measure is obtained using string matching [10] , such as the widely used Jaccard similarity [11] and Cosine similarity [12] measures. In addition, to considering semantic relationships, a few researches have proposed methods based on a linguistic taxonomy [14] such as WordNet [15], from which one can obtain more accurate and less ambiguous semantics for words in the element names.

Another approach is a structural similarity metric, such as those based on three kinds of contexts for schema elements: the ancestor-context, the children-context and the leaf-context [16]. These notions are defined based on the notion of path in schema graphs. Several researches have proposed the structural similarity metrics for XML schema, but they fail to recognize the respective importance of individual entities and relations and the different roles they play in semantic analysis and measurement. A metric based on information content (IC) was proposed to address this problem [17, 18]. This approach measures the similarity between two entities (e.g., between two words, two objects, or two structures), *x* and *y*, based on how much information is needed to describe the commonality between them (e.g., the features or hypernyms that two words share). The more specific the commonality of *x* and *y*, denoted *common*(*x, y*), the more similar *x* and *y* will be. According to information theory, more information is needed for describing more specific objects, and the degree of specificity can be measured by their information content. One can thus define common(*x*, *y*) as the most specific hypernyms, *C*, of both *x* and *y*, and the similarity as

$$Sim(x,y) = I(C) = -\log P(C) \tag{1}$$

where *I*(*C*) is the information content of *C*, and *P*(*C*) can be calculated as word frequencies in a corpus.

Research by [19] shows that better results can be achieved by combining the two approaches. Each of the existing similarity metrics has its strengths and weaknesses. To combine the strengths of different similarity metrics, [2] proposed a hybrid approach called *layered semantic similarity metrics* (or *the layered approach*, for short) that employs a variety of similarity metrics, including lexical, taxonomical, and information content-based metrics. It divides the tree structure of the XML schema into three layers (i.e., top, inner, and atom layers) and applies three different similarity metrics to them. Because each layer typically captures the semantics from different perspectives, it is an effective approach to capturing the semantics in a coherent and justifiable manner. Clearly, this combination of layered measures has a drawback in that it increases the computational time which we address in the following section.

The layered approach proposed two similarity measures: atom level similarity between two atom layers of two elements and label similarity between the labels of elements. For atom level similarity, an IC based measure was proposed as follows:

$$Sim_A(A(x),A(y)) = \frac{2 \cdot \Sigma_{c_i \in A(x) \cap A(y)} \cdot I(c_i)}{\Sigma_{c_i \in A(x)} I(c_i) + \Sigma_{c_j \in A(y)} I(c_j)} \tag{2}$$

where *A(x)* and *A(y)* are the sets of atoms of global elements x and y, respectively. The atom level similarity is used to analyze the atom layers of XML schema.

[2] also proposed a procedure for label similarity as follows:

1) Normalize labels to obtain full words from the concatenations and abbreviations, denoted as *L*(*x*).
2) Calculate the semantic weight of each *L(x) by*

$$w_{IC}(x_i) = \frac{I(x_i)}{\Sigma_{x_k \in L(x)} I(x_k)} \tag{3}$$

   where $I(x_i) = -\log P(x_i)$, and *P*(x_i) are taken as their frequencies in their respective schema;
3) Obtain from the WordNet the description of each word in *L*(*x*) and make the description a set of words of same size, denoted as *W(x);*
4) Measure Sim(*x*, *y*) by *cosine*(*W*(*x*), *W*(*y*)):

$$Sim_T(x,y) = \frac{W(x)W(y)}{|W(x)||W(y)|} = \frac{\sum_{i \subset W(x) \cap W(y)} f_x(i) f_y(i)}{\left(\sum_{i \subset W(x)} f_x(i)^2\right)^{1/2} \left(\sum_{j \subset W(y)} f_y(j)^2\right)^{1/2}} \tag{4}$$

   where $f_x(i)$ is the frequency of the term '*i*' in *W*(*x*).

The label similarity is applied to both top and inner layers of XML schema. For label similarity of the inner layer, the label *x* and *y* are the union of labels of all intermediate nodes. To obtain a unique mapping, the similarities obtained from three layers are combined by a weighted sum: $Sim(x,y) = w_A Sim_A + w_T Sim_T + w_I Sim_I$, where the sum of the weights are normalized to one.

In this paper, we propose an efficient XML schema mapping computational architecture based on the layered semantic similarity metrics using a grid computing technology.

2.2. Grid computing

We now consider various grid computing technologies as a way to address the performance of our compute intensive XML schema similarity analysis. In this subsection, three related distributed computing technologies, Hadoop [22], Globus Toolkit[23] and MPJ [3], are explained. Each technology has different technical backgrounds and performance trade-offs. We compare them with respect to overhead for the initial setup, data management, security, and execution on our computing architecture, which for this study consists of 3 networked Pentium based Intel laptops.

Hadoop is an open source software framework for running applications on large clusters built of commodity hardware. It provides applications both the reliability of a data file system and a parallel computational paradigm named Map/Reduce, which divides the data into many small fragments of work, assigning each fragment a key, value pair which is then distributed to each node in the cluster for execution in parallel and then sorted by the reduce function

The Globus Toolkit provides an alternative approach to distributed computations and also is an open source software that explicitly supports the development of service oriented distributed computing applications and data infrastructures. It better addresses such fundamental issues that relate to security, information infrastructure, resource management, data management, communication, fault detection, and so forth.

A third approach to cluster computing makes use of MPJ which is an extension to the Message Passing Interface (MPI), a standard for the message-passing software layer. It provides a flexible structure based on the master/slave architecture, into which a variety of applications can be easily programmed. Table 1 shows the different features of MPJ, Hadoop and Globus Toolkit.

Table 1: Comparison of MPJ to Hadoop and Globus.

	MPJ	*Hadoop*	*Globus*
Extra SW requirement	Java,	Java, sshd, cygwin	Java, Ant
Setup	System-independent	System-specific	System-independent
Security	No	ssh	WS-security
Data management	No	DFS	GridFTP
Computing	Grid	Clustering	Grid

Hadoop requires more extra software installation and management because it requires the privilege of the administrator to configure the clusters. Moreover, the Map/Reduce paradigm is not effective for all possible parallel computing templates, especially for our XML schema analysis architecture, which did not lend itself well to execute the key value pairs of the Map similarity measure function for all the schema data elements. Globus Toolkit makes extensive use of Web Services to define its interfaces and structure its components, which provide flexible, extensible, and widely adopted XML-based mechanisms for describing, discovering, and invoking network services. It can be applied to our XML schema analysis architecture, but requires a complex environment configuration for Web Services.

On the other hand, MPJ requires simple environment configuration and programming architecture. Processors or machines that cooperatively work together via MPJ are independent of each other in terms of configuration and resource management. Because of this simplicity, our XML schema analysis computing architecture can be more easily implemented as grid computing architecture using MPJ, even if it does not provide any security and data management functionalities.

3. XML schema mapping by SoA and grid

In this section, we propose a service oriented architecture (SoA) for XML schema mapping based on a grid computing technology in order to enhance the computational efficiency of the mapping algorithm. The proposed architecture is extended from the layered approach [2].

3.1. Overview

Various schema mapping tools [25, 26] have been introduced in the e-business market. However, most of the schema mapping tools do not support any semantic similarity analysis methods that have been researched and proposed, for schema mapping.

In this study, we design and implement a Grid computing architecture for XML Schema Mapping based on Service-Oriented Architecture (GridXMLSM-SOA). This system can help not only the existing schema mapping tools, but also e-business vendors, to easily make use of the functionality of semantic similarity analysis. In particular, it uses the layered approach, which can capture the semantics of different perspectives in the XML schemas well.

The layered approach recommends a set of data elements in the target schema as likely mapping/merging candidates for each element in the source schema, based on their semantic similarity scores. In other words, for mapping between a source schema with *n* data elements and a target schema with *m* data elements, we compute the semantic similarities between all possible pairs of source/target elements, generating an *n*m* matrix called the *similarity matrix*. As we explained in Section 2, the semantic similarities can be computed by combining three different similarities for each layer of XML schema. We then recommend a set of candidates according to the similarity ranking.

The performance of this similarity measure approach mainly depends on the number of elements to be compared and the complexity of the semantic similarity algorithms. Because the similarity measure for each pair of source/target elements and the different semantic similarities for each layer can be computed in parallel, grid computing technology can be applied to enhance performance. The next section describes the general architecture and detailed implementation of GridXMLSM-SOA.

3.2. General architecture

Figure 1 shows the overview architecture of GridXMLSM-SOA.

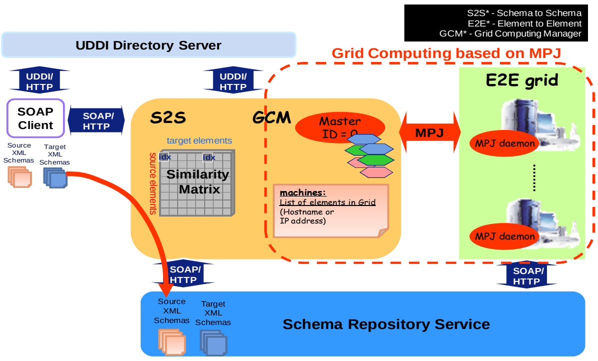

Figure 1: Overview of GridXMLSM-SOA.

The architecture depicted in Figure 1 consists of three main SoA components: a schema mapping SOAP client, Grid enhanced XML Schema Mapping Web Services (GridXMLSM-WS), and a Universal Description Discovery and Integration (UDDI) [27] directory service. The schema mapping SOAP client can be any kind of software that uses the GridXMLSM-WS. It should support messaging using SOAP 1.1 or 1.2 specifications [28]. First, the GridXMLSM-WS publishes its own Web Services Description Language (WSDL) service description at a public UDDI directory server. Any SOAP client can now find the WSDL service description of the GridXMLSM-WS through the public UDDI directory server. Note that there are several supporting tools, such as AXIS2 [13], for generating a SOAP message generator/parser according to the given WSDL. Finally, the schema mapping SOAP client can invoke the GridXMLSM-WS to request a schema mapping analysis for a given source-target XML schema pair.

The GridXMLSM-WS consists of four components: a Schema-to-Schema (S2S) mapping service, a Grid Computing Manager (GCM), an Element-to-Element (E2E) mapping service, and a schema repository service. The S2S mapping service is the main component providing an interface with the similarity mapping analysis function. The similarity mapping analysis produces a similarity matrix that contains the semantic similarities between all comparable source/target element pairs. The GCM is a sub-component of the S2S mapping service that initiates the grid computing network and assigns jobs to the grid cells, which are the E2E mapping services. E2E mapping services execute semantic similarities between the given source/target elements using the given similarity mapping algorithm. Last, the schema repository service is a web service that manages XML schemas via a permanent repository.

3.3. Use cases and scenarios

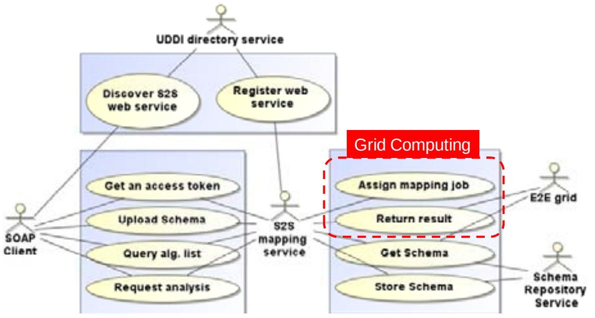

Figure 2: Use case diagram.

Figure 2 illustrates a use case model of GridXMLSM-WS. As shown in the general architecture in Figure 1, there are five actors: a SOAP client, a UDDI directory service, a S2S mapping service with GCM, an E2E mapping service, and a schema repository service. The use case scenario is as follows: 1) S2S mapping service publishes its WSDL service description on UDDI directory service, 2) a SOAP client finds the WSDL service description of the S2S mapping service via UDDI directory service, implements an SOAP messaging generator/parser, invokes the S2S mapping service to upload source/target schemas and request a schema mapping analysis, 3) S2S mapping service creates a similarity matrix and distributes the similarity analysis jobs for every cell in the similarity matrix to the E2E mapping service with indices of the source and target elements to be analyzed, 5) E2E mapping service computes the semantic similarity using the given similarity algorithm and source/target schema from the schema repository service, and 6) S2S mapping service collects all semantic similarity results and, finally, returns the mapping target candidate elements for each element in the source schema based on their semantic similarities. The detailed scenario is shown in Figure 3 as a flow diagram.

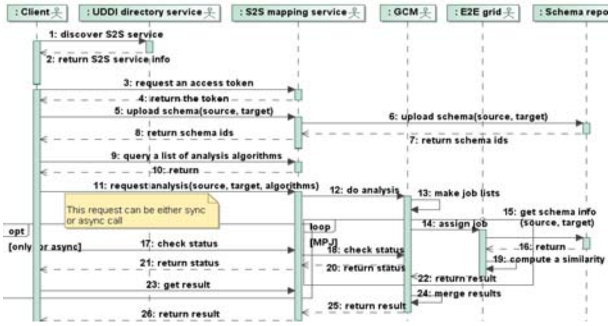

Figure 3: Flow diagram.

3.4. MPJ Implementation

The proposed XML schema analysis structure uses MPJ which was developed to enable high-performance computing (HPC) using Java. Figure 4 shows the pseudo-code of MPJ implementation.

```
1.  public GCM(source, target, args) throws MPIException {
2.  MPI.Init(args);
3.  my_pe = MPI.COMM_WORLD.Rank();
4.  npes = MPI.COMM_WORLD.Size();
5.  float simMatrix[][] = new float[source][target];
6.  if (my_pe == 0) { // Master
7.  for (int i = 0; i < source; i++) for (int j = 0; j < target; j++) {
8.  int send[] = { i, j };
9.  if (k++ < npes) {
10. MPI.COMM_WORLD.Isend(send, 0, send.length, MPI.INT, k%npes, 1);
11. } else {
12. status = MPI.COMM_WORLD.Recv(result, 0, 3, MPI.FLOAT, MPI.ANY_SOURCE,
    2);
13. simMatrix[(int) (result[0])][(int) (result[1])] = result[2];
14. MPI.COMM_WORLD.Isend(send, 0, send.length, MPI.INT, status.source, 1);
15. }}
16. } else { // E2E Element
17. status = MPI.COMM_WORLD.Recv(param, 0, 2, MPI.INT, 0,MPI.ANY_TAG);
18. MPI.COMM_WORLD.Isend(result, 0, result.length, MPI.FLOAT, 0, 2); }
19. MPI.Finalize(); }
```

Figure 4: Pseudo-code of MPJ implementation.

The MPJ approach is well-suited to handling computations where a task is divided up into subtasks, with most of the processes used to compute the subtasks, and only a few processes (often just one process) used for managing the tasks. The manager is called the "master," and the others the "slaves."

The first step to implementing grid computing is to initialize the MPJ (lines 2–4). After that, the processors are divided into two communicators, with one processor as the master (lines 6− 15) and the others the slaves (lines 16− 18). The master assigns initial subtasks to the active slaves and then waits until each slave finishes its task. Once a slave returns the result of its given task, the next subtask is assigned. Thus, faster processors will process more subtasks.

4. Experiments and results

A prototype system with an example SOAP client was implemented using Eclipse [24], JDK 6, and the Google Web Toolkit [8] based on Tomcat and AXIS2. We evaluated its performance using actual industry XML schemas and three semantic similarity algorithms.

4.1 Real data

To test and evaluate the proposed approach, we obtained two actual industry XML schemas from two different workgroups at the Automotive Industry Action Group (AIAG) [6]. The AIAG Resource schema and the Truck and Heavy Equipment (T&HE) schema were used as the target and source, respectively. There were a total of 139 global (top) elements defined in the T&HE schema that needed to be mapped onto the set of 145 global elements of the AIAG schema. Thus, the semantic distances of 139 x 145 (~20,000) pairs of elements needed to be examined.

4.2. Performance Analysis

We tested the execution time to obtain the semantic similarity results using three different algorithms. Without help of grid computing, the execution time was 420 sec. By increasing the number of processors in the grid computing network, the execution time was reduced. Figure 5 shows an exponential decay graph [9] which means the execution time decreases exponentially as the number of processes increases. Note that it is not ideal exponential decay graph, due to trade-off between networking overhead and performance.

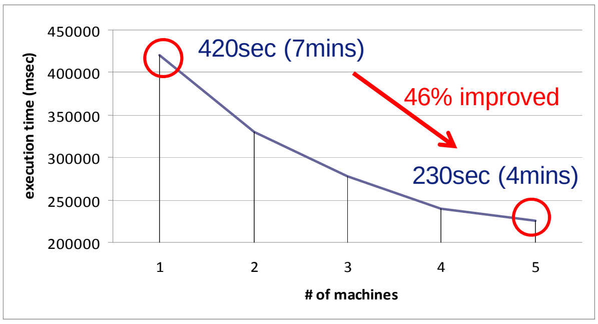

Figure 5: The number of machines vs. execution time.

5. Conclusions and directions for future research

In this paper, we proposed a service-oriented architecture for XML schema mapping based on a grid computing technology in order to enhance the effectiveness of the semantic similarity analysis. We also implemented a prototype computer system architecture to evaluate the proposed approach. The proposed approach and the prototype system can provide efficient and highly extensible XML schema mapping web services. The existing schema mapping tools can extend the software functionality to support automated schema mapping simply by adapting our web services.

A series of experiments were conducted using actual industry XML schemas and the complex semantic similarity algorithms. These showed encouraging improvements in performance. A grid computing network implementing MPJ could successfully improve performance by reducing the computation time of the semantic similarity between two large-scale XML schemas.

We investigated other two grid computing technologies: Hadoop and Globus Toolkit. Hadoop is not appropriate for the XML schema mapping analysis architecture based on the layered approach and Globus Toolkit requires a complex environment configuration for Web

Services. However, both provide better functionalities for grid computing such as security, resource/data management, communication, and fault detection. This calls for further examination of grid computing technologies and for exploring other automated schema mapping approaches that may be fit to Map/Reduce paradigm of Hadoop.

The following immediate steps are planned for future research: 1) extend our experiments using globally established grid computing infrastructures such as SURAgrid [29], TERAGRID [30], and NorduGrid [31], 2) explore other automated schema mapping approaches that can apply Map/Reduce paradigm of Hadoop, and 3) investigate the existing schema mapping tools how to integrate with GridSM-WS. We also plan to conduct similarity measures for large (n,m) schema element pairs employing Hadoop on large numbers of processors at the Multicore Computational Center at UMBC [32] where the order of n, the number of schema elements is of the order of compute nodes in order to to validate the conclusions for large data elements and clusters.

These further researches should be investigated first for more utilization of GridSM-WS tools in real e-business integration works, but the work discussed in this paper shows that service-oriented architecture based on grid computing technology can accomplish XML schema mapping and integration tasks more efficiently.

Acknowledgements

This work was supported in part by NIST award 60NANB6D6206.

Disclaimer

Certain commercial software products are identified in this paper. These products were used only for demonstration purposes. This use does not imply approval or endorsement by NIST, nor does it imply that these products are necessarily the best available for the purpose.

References

[1] E. Rahm and P.A. Bernstein, “A Survey of Approaches to Automatic Schema Matching”, *VLDB Journal*, volume 10, issue 4, 2001, pp. 334-350.

[2] J. Kim, Y. Peng, B. Kulvatunyou, N. Ivezik, A. Jones, “A Layered Approach to Semantic Similarity Analysis of XML Schemas,” IEEE International Conference on Information Reuse and Integration, 2008.

[3] B. Carpenter, *V. Getov, G. Judd, A.* Skjellum, and G. Fox, “MPJ: MPI-like message passing for Java,” Concurrency: Pract. Exper., 12(11):1019–1038, September 2000.

[4] P. Shvaiko and J. Euzenat, “A Survey of Schema-Based Matching Approaches”, *Journal on Data Semantics IV*, LNCS 3730, 2005, pp. 146-171

[5] Y. Peng, “On Semantic Similarity Measures”, Technical Report from Syllogism.Com to NIST, 2006.

[6] Automotive Industry Action Group (AIAG) Website, http://www.aiag.org

[7] The Open Application Group, “Open Application Group Integration Specification”, version 8.0. 2002.

[8] Google Web Toolkit, http://en.wikipedia.org/wiki/Google_Web_Toolkit

[9] Exponential Decay definition from Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Exponential_decay

[10] H. H. Do and E. Rahm, “COMA - A System for Flexible Combination of Schema Matching Approaches”, in Proceedings of the Very Large Data Bases Conference (VLDB), 2001, pp 610–621.

[11] Jaccard similarity, http://www.dcs.shef.ac.uk/~sam/stringmetrics.html#jaccard

[12] Cosine similarity, http://www.dcs.shef.ac.uk/~sam/stringmetrics.html#cosine

[13] Apach Axis2, http://en.wikipedia.org/wiki/Apache_Axis2

[14] D. Yang and D.M.W. Powers, “Measuring Semantic Similarity in the Taxonomy of WordNet”, in the 28th Australasian Computer Science Conference (ACSC2005), Newcastle, Australia, 2005, pp. 315-322.

[15] WordNet, http://wordnet.princeton.edu/

[16] Thang H.Q. and Nam V.S., “XML Schema Automatic Matching Solution,” International Journal of Computer Systems Science and Eng., 4 (1):pp. 68-74.

[17] D. Lin, “An Information-Theoretic Definition of Similarity”, in Proceedings of International Conference on Machine Learning, Madison, Wisconsin, July, 1998.

[18] P. Resnik, “Using Information Content to Evaluate Semantic Similarity in a Taxonomy”, in Proceedings of the 14th International Joint Conference on AI, Montreal, CA, 1995, pp. 448-453.

[19] A. Formica, “Similarity of XML-Schema elements: a structural and information content approach”, *The Computer Journal*, volume 51, issue 2, 2008, pp. 240-254.

[20] English Stopwords List Website, http://www.ranks.nl/tools/stopwords.html

[21] UN/CEFACT TBG17 Harmonisation workgroup http://www.uncefactforum.org/TBG/TBG17/tbg17.htm.

[22] Hadoop, http://en.wikipedia.org/wiki/Hadoop

[23] Globus Toolkit, http://www.globus.org/toolkit/

[24] Eclipse, http://en.wikipedia.org/wiki/Eclipse_(software)

[25] Altova XML Schema mapping tool, http://www.stylusstudio.com/xsd_to_xsd.html

[26] F. Duchateau, Z. Bellahsene, and E. Hunt. XBenchMatch: a Benchmark for XML Schema Matching Tools. In VLDB, pages 1318–1321, 2007.

[27] Universal Description Discovery and Integration (UDDI), http://en.wikipedia.org/wiki/UDDI

[28] SOAP Version 1.2, http://www.w3.org/TR/soap12-part0/

[29] SURAgrid , http://www.sura.org/programs/sura_grid.html

[30] TERAGRID, http://www.teragrid.org/

[31] NorduGrid, http://www.nordugrid.org/

[32] Multicore Computational Center at UMBC, http://www.mc2.umbc.edu/

[33] HR-XML schema ver 2.5, http://www.hr-xml.org/