APPROVAL SHEET

Title of Dissertation: A Semantic Analysis of XML Schema Matching for B2B Systems Integration

Name of Candidate: Jaewook Kim
Doctor of Philosophy, 2011

Dissertation and Abstract Approved: _____

Yun Peng
Professor
Department of Computer Science and Electrical Engineering

Date Approved: _____

Curriculum Vitae

Name: Jaewook Kim.

Degree and date to be conferred: Doctor of Philosophy,
Computer Science, May 2011.

Secondary education: Anyang High School, Anyang, Kyunggi, Korea, 1996.

Collegiate institutions attended:

Pohang University of Science and Technology, Master in Computer and Communication Engineering, 2006.

Pohang University of Science and Technology, Bachelor in Industrial Engineering, 2004.

Major: Computer Science.

Professional publications:

[1] J. Kim and B. Kulvatunyou, "An iterative procedure for efficient testing of B2B - a case in messaging service tests," in *Proceeding of the 2007 International Conference on Interoperability for Enterprise Software & Application – I-ESA 2007*, Madeira, Portugal, Mar 28 – 30, 2007, also published in: R. Goncalves, J. Muller, K. Mertins, and M. Zelm (eds.) *Enterprise Interoperability II: New Challenges and Approaches*, Springer, ISBN 978-1-84628-857-9, 2007, pp. 647 – 658.

[2] J. Kim, Y. Peng, B. Kulvatunyou, N. Ivezic, and A. Jones, "A layered approach to semantic similarity analysis of XML schemas," in *Proceeding of the 2008 IEEE International Conference on Information Reuse and Integration – IRI 2008,* Las Vegas, NV, July 13-15, 2008, pp. 274 – 279.

[3] J. Shin, J. Kim, and N. Ivezic, "Application information mapping test: an efficient content-level semantic equivalence test procedure for B2B integration," *International Journal of Computer Integrated Manufacturing*, 22 (10), October 2009, pp 976 – 986.

[4] J. Shin, J. Kim, and N. Ivezic, "Message-independent test harness for content-level test of B2B standards: an application information mapping test case study," in *Proceeding of the 2009 IEEE International Conference on Information Reuse and Integration – IRI 2009,* Las Vegas, NV, August 10-12, 2009, pp. 74 – 78.

[5] J. Kim, S. Lee, M. Halem, and Y. Peng, "Semantic similarity analysis of XML schema using grid computing," in *Proceeding of the 2009 IEEE International Conference on Information Reuse and Integration – IRI 2009*, Las Vegas, NV, August 10-12, 2009, pp. 57 -– 62.

[6] J. Kim and Y. Peng, "A semantic similarity analysis for data mappings between heterogeneous XML schemas," in: E. Kajan (ed.) *Electronic Business Interoperability: Concepts, Opportunities and Challenges*, IGI Global, Hershey , PA, 2010, pp. 37 – 55.

[7] J. Kim, Y. Peng, N. Ivezic, and J. Shin, "Semantic-based optimal XML schema matching: a mathematical programming approach," in *Proceeding of the 2010 IEEE International Conference on E-Business, Management and Economics – ICEME 2010*, Hong Kong, China, Dec. 28 – 30, 2010.

[8] J. Shin, N. Ivizic, J. Kim, F. Ameri, C. McArthur, S. DeFlitch, and T. Scacchitti, "An experimental evaluation platform for state-of-the-art manufacturing supplier discovery methods," submitted in *Electronic Commerce Research and Applications*, Dec 2010.

[9] J. Kim, Y. Peng, N. Ivezic, and J. Shin, "An optimization approach for semantic-based XML schema matching," *International Journal of Trade, Economics, and Finance*, 2 (1), February 2011, pp. 78 – 86.

[10] Y. Kang, J. Kim, and Y. Peng, "eXtensible Dynamic Form approach for supplier discovery," submitted to *Proceeding of the 2011 IEEE International Conference on Information Reuse and Integration – IRI 2011*.

Professional positions held:

Guest Researcher at NIST (National Institute of Standards and Technology), Gaithersburg, MD, USA, 2006 – Present.

Developer at KorBIT (Korea B2B/A2A Interoperability Testbed Consortium), Pohang, Korea, 2004 – 2005.

Assistant Manager at Penta Security Systems Co., Seoul, Korea, 2002 – 2003.

Assistant Manager at Ints Co., Seoul, Korea, 2000 – 2002.

ABSTRACT

Title of Dissertation: A Semantic Analysis of XML Schema Matching for B2B Systems Integration

Jaewook Kim, Ph.D. Computer Science, 2011

Directed By: Yun Peng, Professor
Department of Computer Science and
Electrical Engineering

One of the most critical steps to integrating heterogeneous e-Business applications using different XML schemas is schema matching, which is known to be costly and error-prone. Many automatic schema matching approaches have been proposed, but the challenge is still daunting because of the complexity of schemas and immaturity of technologies in semantic representation, measuring, and reasoning.

The dissertation focuses on three challenging problems in schema matching. First, the existing approaches have often failed to sufficiently investigate and utilize semantic information imbedded in the hierarchical structure of the XML schemas. Secondly, due to synonyms and polysemies found in natural languages, the meaning of a data node in the schema cannot be determined solely by the words in its label. Thirdly, it is difficult to correctly identify the best set of matching pairs for all data nodes between two schemas.

To overcome these problems, we propose new innovative approaches for XML schema matching, particularly applicable to XML schema integration and data transformation between heterogeneous e-Business systems. Our research supports two dif-

ferent tasks: integration task between two different component schemas; and transformation task between two business documents which confirm to different document schemas.

For the integration task, we propose an approximate approach that produces the best matching candidates between global type components of two schemas, using their layer specific semantic similarities. For the transformation task, we propose another approximate approach that produces the best sets of matching pairs for all atomic nodes between two schemas, based on their linguistic and structural semantic similarities. We evaluate our approaches with the state of the art evaluation metrics and sample schema sets obtained from several e-Business standard organizations and e-Business system vendors. A variety of computer experiments have been conducted with encouraging results that show the proposed approaches are valuable for addressing difficulties in XML schema matching.

Keywords:

E-business, XML schema matching, layered approach, semantic similarity, information contents, mathematical programming, maximum-weighted bipartite graph

A SEMANTIC ANALYSIS OF XML SCHEMA MATCHING FOR B2B SYSTEMS INTEGRATION

By

Jaewook Kim

Dissertation submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2011

ACKNOWLEDGEMENTS

This dissertation is the outcome of four years experiences at University of Maryland, Baltimore County (UMBC) and National Institute of Standards and Technology (NIST). There are many people who helped me in this work, either directly by working on projects with me or indirectly by giving me vision and support.

First and foremost, I would like to express my deep gratitude to my honorific advisor Prof. Yun Peng. He offered me so much advice, continuous encouragements, patiently supervising me, and always guiding me in the right direction. I have learned a lot from him, without his help I could not have finished my dissertation successfully.

Special thanks are also given to Dr. Nenad Ivezic, Dr. Boonserm (Serm) Kulvatunyou, Dr. Albert Jones and Prof. Hyunbo Cho. They offered me the great opportunity on working with them at NIST. I appreciate all their contributions on time, ideas, and funding to make my PhD experience productive and stimulating.

I would like to thank Prof. Charles Nicholas, Prof. Zary Segall, Prof. Milton Halem, Prof. Hyunbo Cho and Dr. Nenad Ivezic for kindly serving on the committee and reviewing my dissertation.

I would like to thank all members of UMBC and NIST who worked together, including Edward J. Barkmeyer, Sookyoung Lee, Salifou Sidi Mahaman Malick, Igor Miletic, Marko Vujasinovic, Jungyub Woo, and Yang Yu. I especially thank Junho

Shin and Yan Kang who hardly worked with me and contributed efficiently to the development of proposed solutions.

I would like to thank all my friends, especially Jaemin Kim who has always support me through his kind and active friendship.

I would like to thank Mr. Youngchae Kim and Mr. Hyoungik Kim who give the heartfelt prayers for me to successfully finish my Ph.D. degree.

I am forever indebted to my parents for their understanding, supports, endless patience and encouragement when it was most required.

Last but not least, I would like to express my deep gratitude to my wife, Jumi Yoon, and my daughter, Yubin Kim. They form the backbone and origin of my happiness. Their love and support without any complaint or regret has enabled me to complete this Ph.D. project. I owe my every achievement to both of them.

Without the help of the above people, this work would never have been completed, nor even begun.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

This dissertation studies the problems of the XML schema matching for integration and interoperability among heterogeneous electronic business (e-Business) systems. We propose new innovative approaches for matching two XML schemas based on semantic similarity measures. The proposed approaches utilize the semantics embedded in XML schemas and compute the semantic similarities between elements or attributes of two schemas, using several semantic similarity measures, mathematical programming techniques, and linguistic thesauri resources.

I.1. The Motivations

An important issue in e-Business integrations is to provide support for the seamless exchange of information either within or across enterprises. A common approach to successful integration of enterprises' systems is to standardize the business data exchange requirements [Langenberg 2005 and Murphy 2008]. These requirements are most often specified using abstract data models, called schemas, which defines syntax and some extent semantics of data in business documents.

Over the past decades, the eXtensible Markup Language (XML) has emerged as one of the primary languages to help information systems in sharing structured data

[W3C 1998a]. Especially, XML schemas [W3C 2001a; W3C 2001b; and W3C 2001c] have been widely used in the e-Business industry for enterprises to exchange the business documents with their partners (e.g., suppliers and customers) in the supply chain. The XML technology has clearly emerged as the most promising standardization effort for business documents' representations and data transformations on the Internet [Alan 2001]. As a platform of independent representation technology, XML offers several benefits: 1) simultaneously human- and machine-readable format, 2) self-documenting format that describes structures and field names as well as specific values, and 3) the hierarchical structure format suitable for most (but not all) types of documents.

Many enterprises and organizations have defined their own XML schemas to describe structures and contents of the business documents to be used in their Business-to-Business (B2B) transactions. Many organizations have also published standard XML schemas to be shared in the B2B transactions within specific industry domains (e.g., e-manufacturing, e-government, and e-health industries) [Bussler 2001; Medjahed 2003; and Shim 2000]. The popularity of the XML and XML schema leads to an exponential growth of B2B transactions. This success, however, leads to several problems: 1) individual enterprises often create their own XML schemas with information most relevant to their own needs; 2) different enterprises or organizations often choose different XML representations for the same or similar concepts; and 3) the enterprises often extend or re-define the existing standard XML schemas in their own ways for their own needs.

Schema matching has thus become a critical step to achieve the seamless exchange of information among heterogeneous e-Business systems supported by different XML schemas. Schema matching is a process that takes two heterogeneous schemas as input plus possibly some auxiliary information, and returns a set of dependencies between semantically related two elements or two attributes [Shvaiko 2005]. This process has been largely performed by human engineers, manually, who are at best supported by some graphical interface tools. This manual matching process is known to be very labor-intensive, costly, and error-prone [Rahm 2001]. As the e-Business systems grow to handle more complex databases and applications, their schemas become larger and more complex. This further increases the search space to be examined as well as the number of correspondences to be identified. As a result, it is critical to automate the schema matching task as much as possible so as to reduce the costs of labor-intensive data integration work.

Many schema matching approaches have been proposed [Rahm 2001 and Shvaiko 2005]. However, the challenge is still daunting because of the complexity of schemas and immaturity of technologies in semantic representation, measuring, and reasoning. For a better understanding of XML schema matching problems, we analyze the general use cases and challenges of the matching problems in the next section.

I.1.1. Use cases and challenges for XML schema matching problems

An XML schema defines a set of discrete elements and attributes for a class of XML documents, aiming at defining the structure, content, and to some extent semantics of XML documents [W3C 2001a]. XML documents that adhere to an XML

schema are said to be instances of that schema (i.e., XML instances). XML schemas or instances are typically viewed as labeled trees where each node represents a data element or an attribute named by a label consisting of English word, concatenation of words, or their abbreviations [Rahm 2001]. Most schema matching approaches analyze the similarity between two labeled trees based on their structural and linguistic information [Rahm 2001 and Shvaiko 2005]. For the structural similarities, they analyze the differences in the hierarchical tree structures. For linguistic similarities, they typically analyze the meaning (semantics) of nodes in the labeled tree. The semantics is often obtained by lexical analysis of English words in the labels of nodes.

According to the types of the e-Business standard schemas, XML schemas can be classified into two types. The first type is the *component schema*. This type of schema defines a set of global type components (often called global elements) that can be either extended or reused by other components [Meadows 2004]. The term "components" here refers to either elements or types [W3C 2001a]. The examples of component schema include 1) Common Core Components (CCC) of Open Applications Group Integration Specification (OAGIS) developed by the Open Applications Group (OAG) consortium [OAGIS 2002], 2) ebXML Core Component [ebXMLCC 2008], 3) UN/CEFACT TBG 17 [TBG 2008], and 4) Health Level Seven International (HL7) Services-Aware Interoperability Framework (SAIF) Core Components [HL7 SAIF 2010]. Component schemas can be thought of as a collection of labeled trees, each of which corresponds to a global element, as shown in Figure I.1 (a).

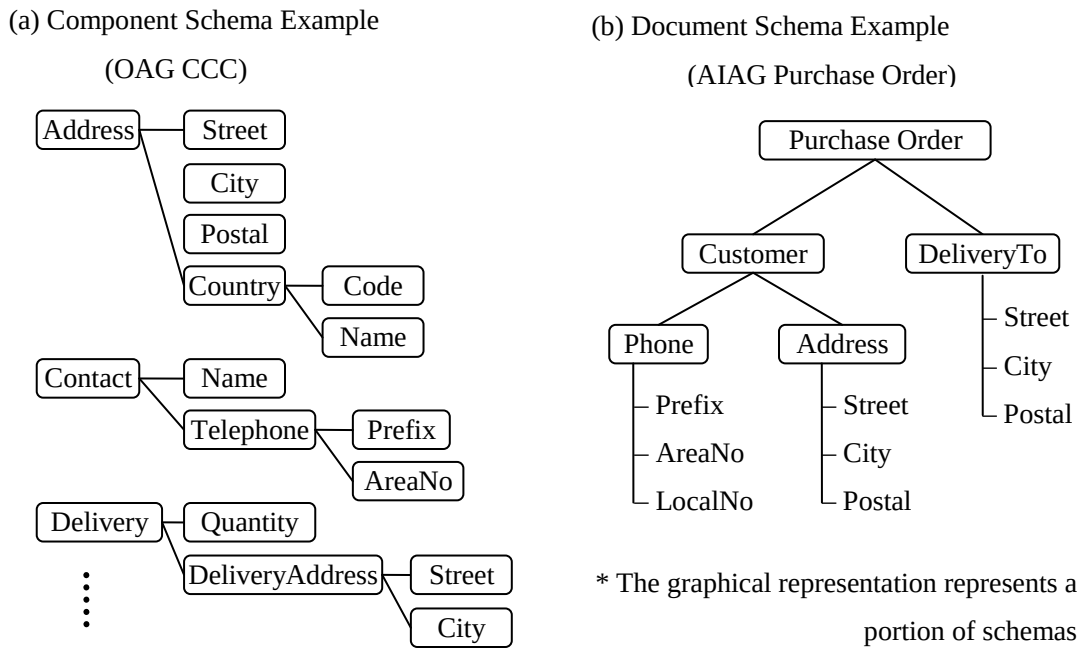


Figure I.1. An example of tree representations for component schema and document schema.

The second type is the *document schema*. It defines the syntax and structure of a single global element for a class of valid XML document (instance), often called Business Object Document (BOD). The examples of the *document schema* include 1) OAGIS BODs [OAGIS 2002] – e.g., "purchase order schema", 2) BODs developed by Standard for Technology in Automotive Retail (STAR) [STAR 2008] – e.g., "replacement part order schema", and 3) BODs developed for Inventory Visibility and Interoperability (IV&I) project [IV&I 2008] of Automotive Industry Action Group (AIAG) [AIAG 2008] – e.g., "order view schema". The *document schema* can reuse or extend the components defined by the *component schemas*. It can be viewed as a single labeled tree as shown in Figure I.1 (b).

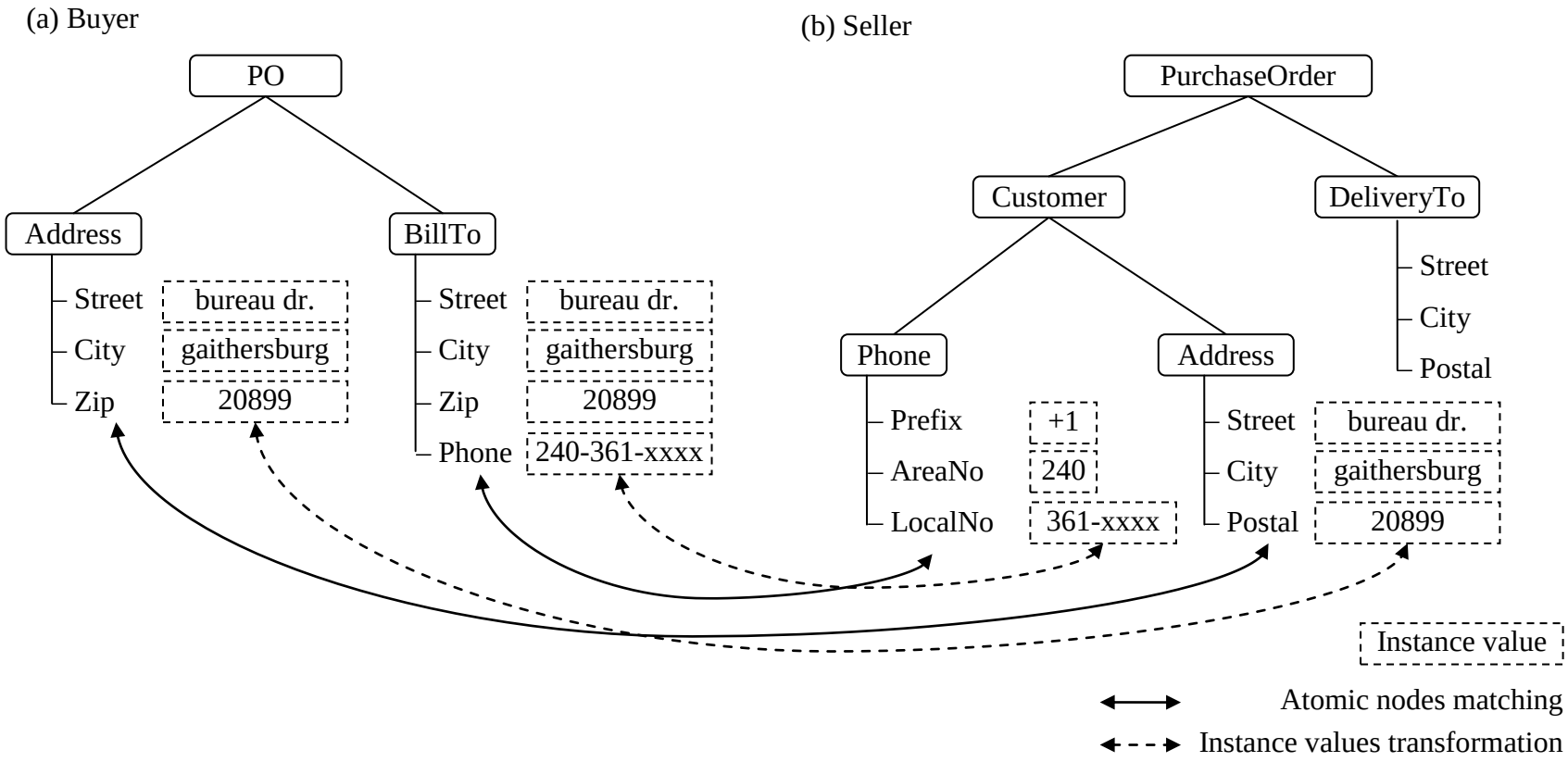


Figure I.2. An example of tree representations for matching between two XML instances.

These two types of schemas lead to different matching problems. The *component schema matching* primarily seeks to identify the relations between two sets of labeled trees (i.e., two sets of global type components) – we call it *c-matching*, whereas the *document schema matching* identifies relations between nodes (elements or attributes) of two labeled trees (i.e., two schemas) – we call it *d-matching*.

The *d-matching* problems can be further classified according to their purposes. If two document schemas need to be *fully* matched to create an integrated schema, all or most nodes in one schema should be matched to some nodes in the other schema which – we call it *f-matching*. However, if the matching is to determine how to transform one instance into another, only leaf nodes (also called *atomic nodes* because it cannot be further decomposed) in the schema trees need to be matched – we call it *a-matching*. Matching between *atomic nodes* of two XML document schemas helps to determine how a certain value in one XML instance can be transformed to a certain value of the other for successful exchange of information. Figure I.2 illustrates an example of *a-matching* between two different "Purchase Order" XML schemas. The figure also shows the possible data transformations between two XML instances based on the results of *a-matching*.

Among these different matching problems, this dissertation focuses on *c-matching* and *a-matching*. Many organizations have published different standard XML schemas. *C-matching* is one of the most important steps to integrate different XML standard schemas, specifically for global type components. As an example of *c-matching*, we have undertaken a matching task between two XML schemas devel-

oped by two different workgroups of Automotive Industry Action Group (AIAG) [AIAG 2008]. The two schemas, called Truck and Heavy Equipment (T&HE) and AIAG Common Resources, have been developed individually. However, they include many similar components since both are extensions of OAGIS Common Core Components [OAGIS 2002]. Both schemas are large and complex with hundreds of components. AIAG has tried to merge them into a single standard schema with a significant amount of cost in terms of human-hours with only limited success.

A-matching is also very important for successful e-Business systems integration and B2B data transactions. The XML technology in the B2B transactions has been mostly used for document transformation. *A-matching* is one of the most important steps to transform one business document into another. A good example of the *a-matching*, are the matching tasks among different Purchase Order (PO) BOD schemas individually defined by different organizations.

We propose new innovative techniques to address three challenging problems in these types of schema matching. First, the existing approaches have often failed to sufficiently examine and utilize semantic information imbedded in the hierarchical structure of the XML schema, which the schema designer intended to. Our analysis shows that the data elements in the XML schema can be divided into several layers according to the level of hierarchy. Moreover, typically different layers carry different aspects of semantics of the data elements, which require different layer-specific approaches to measure the similarities.

Secondly, due to synonyms (different words meaning the same thing) and polysemies (one word having different meanings in different contexts) found in natural languages, the meaning of a data node in the schema cannot be determined solely by the words in its label. Although XML does not provide means to formally define the semantics, the *semantic ambiguity* can be reduced by contextual information such as the meaning of the words composing the labels of its neighboring nodes or external document corpus resources.

Thirdly, it is difficult to correctly identify the *best set* of matching pairs for all data nodes between two schemas. This is because a data node in one schema may match more than one data node in the other schema (with different semantic similarities). Furthermore, best-matching pairs identified in isolation do not necessarily form the globally optimal matching between two sets of data nodes.

I.2. Thesis Statement

The objective of this dissertation is to develop effective approaches to XML schema matching, particularly applicable to XML schema integration and data transformation among heterogeneous e-Business systems. Our research supports two different tasks: *integration task* between two *different component schemas*; and *transformation task* between two business documents which confirm to *different document schemas*.

For the *integration task*, we propose an approximate *c-matching* approach that produces the best matching candidates between *global type components* of two schemas, using their layer specific semantic similarities. For the *transformation task*, we

propose an approximate *a-matching* approach that produces the *best sets* of matching pairs for all *atomic nodes* between two schemas, based on their linguistic and structural semantic similarities. We validate and evaluate our approaches with the state of the art evaluation metrics and sample schemas obtained from several e-Business standard organizations and e-Business system vendors.

I.3. Dissertation Outline

The remainder of this dissertation is organized as follows. Chapter II explains about a literature survey regarding schema matching and the related work including similarity measures, parallel and distributed computing technologies, and evaluation metrics. In particular, the similarity measure, which is the main subject of this research, is intensively surveyed. Chapter III describes a formal definition of XML schema matching problems in order to clearly specify the scope of problems addressed in this research. In this chapter, the assumptions that underlie our proposed solution are also specified. The proposed schema matching approaches for semantic similarity analysis are described in Chapter IV (for *c-matching*) and Chapter V (for *a-matching*). Chapter IV addresses the use of layered approaches in *c-matching*. A large-scale schema matching approach based on the parallel and distributed computing technologies is also addressed in the Chapter IV to reduce the computation time of the matching algorithm. Chapter V presents an optimization approach using different mathematical programming techniques for *a-matching*. Both chapters provide experiment results for the comparative analyses. Chapter VI introduces an XML instance matching application to address the issue of supplier discovery, based on the

proposed *a-matching* approach. Finally, in Chapter VII we close this dissertation with a summary of contributions.

CHAPTER II

BACKGROUND AND RELATED WORK

This chapter surveys existing research related to schema matching. It is divided into the subject areas of classification of schema matching, state of the art schema matching approaches, parallel and distributed computing technologies for schema matching, and performance measures.

II.1. Classification of Schema Matching

Many schema matching methods have been already proposed (summarized in surveys by [Rahm 2001 and Shvaiko 2005]). These methods usually first attempt to identify semantic relationships between the elements of two schemas. According to Shvaiko & Euzenat (2005), the schema matching techniques can be distinguished as two main alternatives by the granularity of the matching: element-level and structure-level. The element-level approaches determine the matching elements in the target schema for each element of the source schema; whereas structure-level approaches refer to matching combinations of elements that appear together in a structure. In the ideal case of a structure-level approach, all components of the structures in the two schemas fully match. These techniques can be further classified by different types of elementary matching techniques they use.

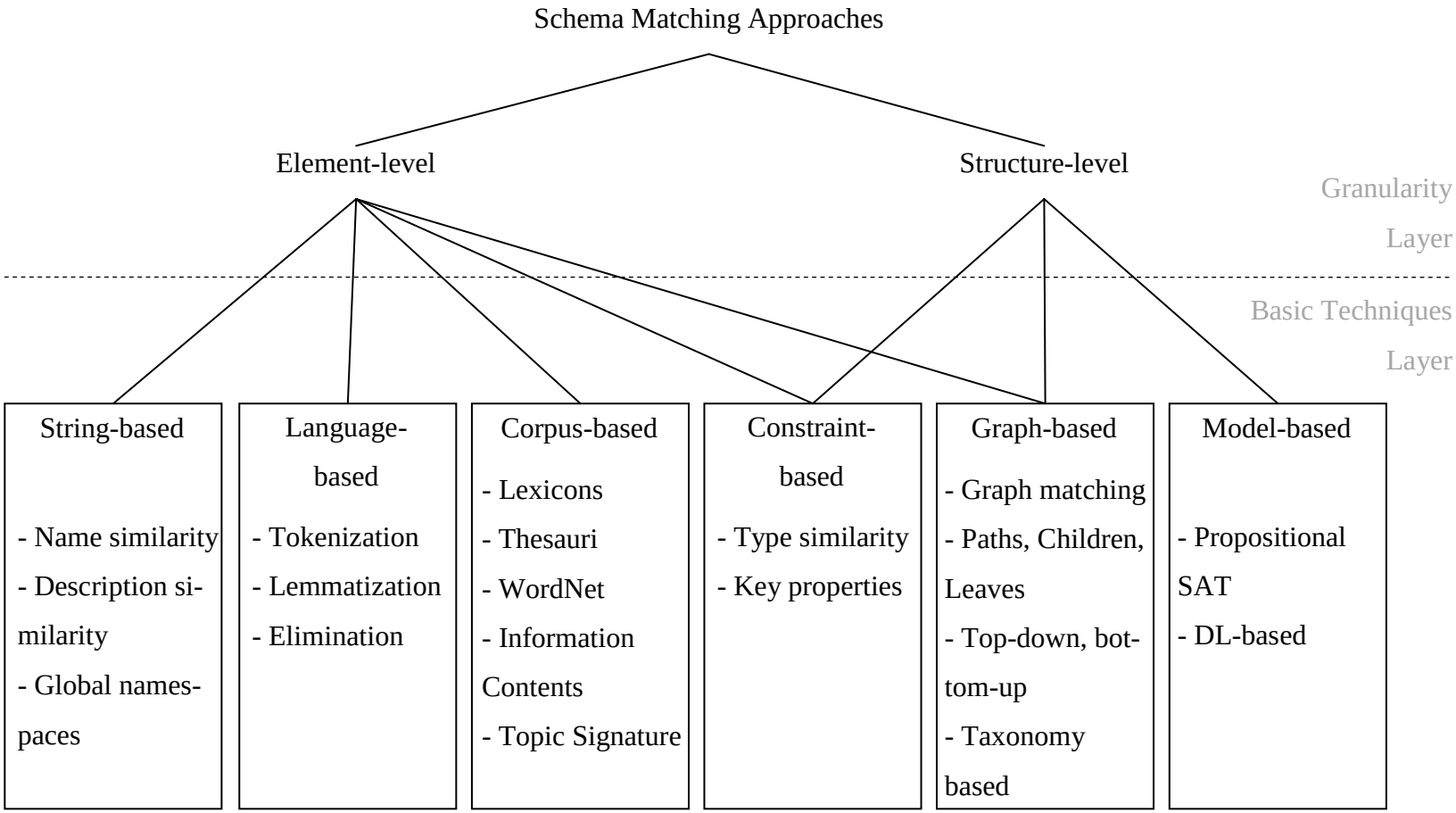


Figure II.1. Classification of schema matching approaches.

Figure II.1 shows a simplified version of the classification of schema matching techniques suggested by [Shvaiko 2005] based on frequently used techniques. Note that all ontology related techniques are omitted. Also, corpus-based metric includes all the techniques using corpus (including linguistic resources), and graph-based metric includes all the techniques related to the graph analysis techniques (including taxonomy-based metric). The leaves in the figure represent classes of elementary matching techniques and their concrete examples.

II.1.1. String-based metric

The simplest matching technique is the string-based metric which computes similarity between two terms or their descriptions using its linguistic information. There are a variety of string-based metrics, such as the widely used cosine similarity and Jaccard coefficient measures.

The cosine similarity [van Rijsbergen 1979] is a measure of similarity between two vectors of *n* dimensions by finding the cosine of the angle between them. For two vectors of attributes, A and B, the cosine similarity is represented using a dot product and magnitude as:

$$Sim_{\cos}(A,B) = \frac{A \cdot B}{|A||B|}. \qquad \text{(II.1)}$$

The Jaccard coefficient [Sneath 1957] is an alternative measure of cosine similarity, which appears to be more popular in the context of similarity search [Markov 2007]. It measures similarity between two sample sets, and is defined as the size of

the intersection divided by the size of the union of the sample sets. For the two sets of A and B, the Jaccard coefficient is given as:

$$Sim_J(A,B) = \frac{|A \cap B|}{|A \cup B|}. \quad \text{(II.2)}$$

The similarity between two strings can be also measured by counting the number of the occurrences of different *n*-grams [Kondrak 2005], i.e., the substrings of length *n*, in the two strings. The more similar the strings are, the more *n*-gram they will have in common. The similarity can be defined as:

$$Sim_{n\text{-}grams}(A,B) = \frac{2 \times |n\text{-}grams(A) \cap n\text{-}grams(B)|}{|n\text{-}grams(A)| + |n\text{-}grams(B)|}, \quad \text{(II.3)}$$

where *n-grams*(A) is a multi-set of letter n-grams in A.

The *n*-grams can be used with various length, but *bi*-grams (*n*=2) and *tri*-grams (*n*=3) are particularly popular text similarity measure.

II.1.2. Language-based metric

The string-based metric can be enhanced using language-based techniques which are a kind of preprocessor for the input string. It is based on Natural Language Processing (NLP) techniques exploiting morphological properties of the input string [Aikins 1981]. There are three types of basic techniques: tokenization, lemmatization, and elimination [Madhavan 2001 and Shvaiko 2005]. The tokenization parses an input text into sequences of tokens (e.g., Vehicle_Activities → {vehicle, activities}). The lemmatization finds all possible basic forms of the input word (e.g., Activities →

Activity). The elimination removes all stop-words from the input text. These techniques can improve the results of the string-based metric by reducing the noise in the input text.

II.1.3. Corpus-based metric

Corpus-based metric can also improve the string-based metric by obtaining more accurate and less ambiguous semantics (e.g., synonyms or hyponyms) for words in the element labels. Not only the common knowledge corpora such as WordNet [Miller 1995], but also the domain specific corpora can be used to enrich the meaning of the words. One of the important resources in a corpus is the lexical taxonomy among words (e.g., parents, children, ancestor, and descendant relationships). Some researches have been proposed based on a lexical taxonomy of the corpus [Qin 2009 and Yang 2005].

Another important resource obtained from corpus is the contents linked to topically related words. Topically related words form the Topic Signatures [Lin 2000] which provide word vectors related to particular topics. Topic Signatures are built by retrieving a group of words that are related to a target word from corpus. The topic signature can be defined as a family of related terms $\{t, <(w1,s1)\ldots(wi,si)\ldots>\}$, where t is the topic (i.e. the target concept) and each w_i is a word associated with the topic, with strength s_i.

II.1.4. Information contents-based metric

Corpus also provides the statistical information related to the importance of words. Different importance individual entities and relationships have played the differential role in semantic similarity measurement. The information content (IC)-based metric was proposed to utilize this statistical information [Lin 1998 and Resnik 1995]. This approach measures the similarity between two entities (e.g., two words, two objects, or two structures) *A* and *B* based on how much information is needed to describe *common*(*A*, *B*), the commonality between them (e.g., the features or hypernyms that two words share). According to information theory [Cover 1991], entities which appear widely in many objects have less information than those which appear rarely. In other words, more specific entities carry more information than generic and common entities. Therefore, the more specific the *common*(*A*, *B*) is, the more similar *A* and *B* will be. In information theory, the information content of a concept or word *C* is defined as $I(C) = -\log P(C)$. Then *common*(*A*, *B*) can be measured by the information content of the most specific common hypernyms of *A* and *B*, and the similarity between *A* and *B* is given as

$$Sim_{IC}(A,B) = \max_{C \in S(A,B)} I(C) = \max_{C \in S(A,B)} -\log P(C), \qquad \text{(II.4)}$$

where *S*(*A*, *B*) is the set of all concepts that subsume both *A* and *B*, *I*(*C*) is the information content of *C*, and *P*(*C*) can be calculated as word frequencies in a corpus.

However, Eq. II.4 is not a good similarity measure because it does not produce the maximum value when the two concepts are identical and it does not normalize to

be in [0, 1]. To improve this measure, Lin (1998) proposed another information content based measure. In a general form, this measure is defined as

$$Sim_I(A,B) = \frac{I(comman(A,B))}{I(description(A,B))}, \quad \text{(II.5)}$$

where *description*(*A*, *B*) is the sum of *common*(*A*, *B*) and *difference*(*A*, *B*). Eq. II.5 can be seen as a normalized version of Eq. II.4.

The difficulty with Eq. II.5 is that the functions *description*(*A*, *B*) and *common*(*A*, *B*) are specified on application and need to be figured out before the similarity can be measured. Thus, for tree-like IS-A taxonomies, Lin also suggested

$$Sim_I(A,B) = \frac{2 \cdot \log P(C)}{\log P(A) + \log P(B)}, \quad \text{(II.6)}$$

where *C* is the most specific subsumer of *A* and *B* with the smallest prior probability and the probabilities can be obtained according to the frequencies in a corpus.

Information contents of words or concepts can also be used as their weights when computing composite similarity measure between two groups of words.

II.1.5. Constraint-based metric

Constraint-based technique is also useful to compare the internal constraints between two elements, such as types, cardinality of attributes, and keys. The datatype comparison involves comparing the various attributes of a class with regard to the datatypes of their values. For example, an element ‘delivery’ of the datatype ‘date’

can be considered closer to an element ‘transport’ of the datatype ‘deliverydate’ than an element ‘shipping’ of the datatype ‘integer’.

II.1.6. Graph-based metric

A variety of graph-based techniques have been proposed for structure-level matching. Typically, the graph-based metric quantifies the commonality between two components by taking into account the linguistic similarities of multiple structurally-related sub-components of these terms (e.g., children, parents, and leaf components). Because most schemas can be viewed as hierarchical graphs containing terms and their parent-child relationships, many matching algorithms have been developed based on either top-down or bottom-up traversal techniques to analyze all elements [Rahm 2001]. Among the existing approaches, TransScm [Milo 1998] and Tess [Lerner 2000] are based on the top-down approach, while Cupid [Madhavan 2001] and Similarity Flooding [Melnik 2002] take the bottom-up approach.

Another technique of graph-based metrics is a taxonomy-based technique that can be applied to ‘IS-A’ taxonomy such as ontology. The edge counting approach is a well-known traditional approach based on conceptual distance in taxonomy [Rada 1989]. The principle of the edge counting is simple and intuitive. It computes the shortest path between two nodes in the taxonomy, presents the most intuitive method to evaluate the semantic similarity in a hierarchical taxonomy. Another taxonomy-based approach, known as bounded path matching [Noy 2001], takes two paths, with links among classes defined by the hierarchical relationships, compares terms and their positions along these paths, and identifies similar terms.

The graph-based metric typically provides a more comprehensive measure than the string-based and corpus-based similarity metrics do, because it looks beyond the individual labels and considers terms' relationships to others. However, it often fails to recognize the semantics in the language and corpus.

II.1.7. Hybrid metric

Each existing similarity measure has its strengths and weaknesses. More importantly, each measure typically makes use of only a part of the available semantic information. Therefore, a matching that uses just one approach is unlikely to achieve a matching performance as good as the combined approaches that combine several matching approaches [Rahm 2001].

The combined matchers are grouped into hybrid and composite matchers. Hybrid matchers utilize two or more different criteria in an integrated approach whereas composite matchers combine the outcomes of two or more different matchers which have been run independently.

II.2. The State of the Art Schema Matching Approaches

After considering the classifications of schema matching approaches, we now look at the state of the art schema matching approaches. Most of them combine several semantic metrics as either composite or hybrid approach.

LSD (Learning Source Descriptions) [Doan 2001] provides a composite matching algorithm based on machine-learning techniques to automatically combine several match results. The algorithm matches a pair of schemas based on *1-to-1* atomic-level

matching, similar to *a-match* which we defined earlier in this dissertation. The LSD is based on the combination of several match results obtained by independent learners. The predictions of individual learners are combined by a so-called meta-learner, which weighs the predictions from a learner according to its accuracy shown during the training phrase.

TranScm [Milo 1998] provides a schema matching method for data translation and conversion based on the syntactic analysis of the structures. Input schemas are transformed into labeled graphs, which is the internal schema representation. Edges in the schema graphs represent component relationships. All other schema information (name, optionality, #children, etc.) is represented as properties of the nodes. The matching is performed node-by-node, considering *1-to-1* matching cardinality in a top-down fashion. There are several matchers which are checked in a fixed order. Each matcher is a "rule" implemented in Java. They require that the match is determined by exactly one matcher per node pair. If no match is found or if a matcher determines multiple match candidates, user intervention is required to provide a new rule (matcher) or to select a match candidate.

Tess [Lerner 2000] provides a matching algorithm that deals with schema evolution. Tess takes definitions of the old and new types and identifies pairs of types as matching candidates. It then recursively tries to match their substructure in a top-down fashion. Like TransScm, it presumes a high degree of similarity between two schemas. It identifies pairs of types as match candidates, and then recursively tries to match their substructure in a top-down fashion. Two elements are match candidates if

they have the same name, have a pair of subelements that match (i.e., that are of the same type), or use the same type constructor (in order of preference, where name matching is most preferred). The recursion bottoms out with scalar subelements. As the recursive calls percolate back up, matching decisions on coarser-grained elements are made based on the results of their finer-grained subelements. In this sense, Tess performs both structure-level and element-level matching.

Similarity Flooding (SF) [Melnik 2002] approach provides a hybrid matching algorithm based on similarity propagation. This method represents schemas as directed labeled graphs. The algorithm manipulates the directed labeled graphs in an iterative fix-point computation to produce an alignment among the nodes of the input graphs. The matching begins with a string-based comparison of the schema elements and analyzes the structure-level relationships on the assumption that if two nodes from two schemas are similar, then their neighbors may also be somehow similar. From iteration to iteration the spreading depth and the similarity measure are increasing till the fix-point is reached. The result of this step is a refined alignment which is further filtered to finalize the matching process.

Cupid [Madhavan 2001], proposed by Microsoft Research[1], is another hybrid matching approach. It is comprised of element- and structure-level matching approaches, and it computes the similarity with domain-specific thesauri as the linguistic information resources do. The algorithm consists of three phases: 1) linguistic

[1] http://research.microsoft.com/

matching, 2) structural matching, and 3) mapping elements generation. The linguistic matching phase computes linguistic similarity between element labels of two schemas based on string-based techniques and domain-specific thesauri. The structural matching phase computes structural similarity weighted by leaves which measure the similarity between two contexts in which elementary schema elements occur. The mapping elements generation phase computes weighted similarity and generates the overall alignment by choosing pairs of schema elements with weighted similarity which are higher than a threshold. The Cupid algorithm provides an effective algorithm to traverse the tree in a combined bottom-up and top-down manner.

S-Match [Giunchiglia 2004] follows a graph-based matching algorithm, which decomposes the tree matching problem into a set of node matching problems. Each node matching problem is translated into a propositional formula, which then can be efficiently resolved using state of the art propositional satisfiability (SAT) deciders. S-Match also utilizes simple structural information because the word-sense is applied to the parent node's name and concept as well as those of the node to be evaluated. Its output is one of the possible relations between two schema/ontology nodes – equivalence, more or less general, mismatch, and overlapping. This type of outputs makes users more comfortable to make their own decision than to use other tools giving a coefficient of relationship degree.

COMA (COmbination of MAtching algorithms) [Do 2003] provides various ways for combining different matchers. COMA provides an extensible library of matching algorithms and composes various individual similarity coefficients. Though

the present COMA utilizes 6 elementary matchers (e.g., n-gram and edit distance), 5 hybrid matchers, and one reuse-oriented matcher; its matching library is extensible for any other matchers. The reuse-oriented matcher uses previously obtained results for entirely new schemas or for its fragments. The reuse of previous matching results makes COMA efficient. However, this matcher should be carefully applied. This is because the reuse can give undesirable low similarity coefficients between similar or even identical two elements and cause multiple (i.e., *M-to-N*) correspondences. Moreover, once a wrong relationship is established at a previous matching, the incorrect relationship will always appear whenever the reuse-matcher is called.

II.3. Quality of Matching Measures

To evaluate the quality of the matching measures, several performance evaluation scoring functions have been proposed [Do 2003]. The typical method to evaluate the matching measures is to compare the derived matchings to the real matchings by experienced human integrators. The human integrators first have to manually generate a set of real matchings which can be used as a "gold standard" which compares it to the automatically derived matchings. The comparison of the real matchings to derived matchings is shown in Figure II.2.

The set of derived matchings can be categorized as *True Positives* (B); *False Positives* (C); *False Negatives* (A); and *True Negatives* (D). Note that among all derived matching; only the *True Positives* are considered as correct matchings. Based on these categories, two basic measures of matching quality, *Precision* and *Recall* [van Rijsbergen 1979], can be computed.

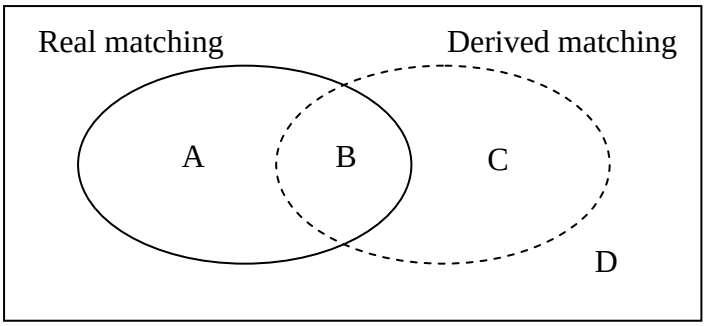


Figure II.2. Comparing true results and derived results.

In their original definitions, *Precision* expresses the proportion of correct matchings among all the derived matchings, which can be defined as:

$$Precision = \frac{\text{number of correct matchings derived}}{\text{total number of matchings derived}} = \frac{|B|}{|B|+|C|}. \quad \text{(II.7)}$$

Recall expresses the proportion of the found correct matchings among all the correct matchings, which can be defined as:

$$Recall = \frac{\text{number of correct matchings derived}}{\text{total number of correct matchings}} = \frac{|B|}{|A|+|B|}. \quad \text{(II.8)}$$

Precision = 1 indicates that all the matchings derived by the matching measures are correct, while *Recall* = 1 means that all correct matchings are found by the matching measures. A trade-off between *Recall* and *Precision* is provided by the *F-measure* [van Rijsbergen 1979]:

$$F\text{-}measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \quad \text{(II.9)}$$

Although formal *F-Measure* supports different relative importance to be attached to *Precision* and *Recall*, Eq. II.9 is a special case when *Precision* and *Recall* are considered equally important. Another combined measure, called *Overall*, estimates the post-match efforts needed for both removing wrong and adding missed matches [Melnik 2002]. It is defined as follows:

$$Overall = Recall \times \left(2 - \frac{1}{Precision} \right). \quad \text{(II.10)}$$

These traditional performance metrics can only be calculated for binary relevance (i.e., correct or incorrect). Matching algorithms often provide multiple matching results ranked according to their relevance (similarity) scores. *Discounted Cumulative Gain* (*DCG*) is a precision-like metric that supports graded relevance and discounting by rank [Jarvelin 2002]. *DCG* estimates the usefulness (or gain) of a matching not only by examining the results derived by the matching algorithm but also by considering the order in which the derived results are presented. The *DCG* provides a means for performance analysis of the matching algorithm that derives a ranked list of results. It accumulates the gains of a matching from the top of the result list to the bottom with the gain of each result discounted at lower ranks.

DCG originates from more primitive metric called *Cumulative Gain* (*CG*) which does not include the position of a result in the consideration of the usefulness of a result set. The *CG* at a particular rank position *p* is defined as:

$$CG_P = \sum_{i=1}^{p} rel_i \,, \tag{II.11}$$

where rel_i is the graded relevance of the result at position i in the ranked list of results.

DCG then penalizes the highly relevant matching ranked in lower positions with weights reduced by a reciprocal logarithm function. The *DCG* accumulated at a particular rank position p can be calculated as follows:

$$DCG_P = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i} \,. \tag{II.12}$$

DCG_{p} should be normalized to reflect the size of the result set that may vary among different matching measures. The normalized *DCG* is defined as

$$nDCG_P = \frac{DCG_P}{IDCG_P} \,, \tag{II.13}$$

where $IDCG_p$ is an ideal *DCG* that can be calculated from a flawless sequence of results (i.e., ranked list of correct matchings).

II.4. Parallel and Distributed Computing Technologies for Schema Matching

As more complicated and larger schemas and highly complex matching algorithms have been introduced, large-scale schema matching has become a challenging problem in terms of the computational cost. Several research groups have actively studied this issue and offered their solutions. [He 2004] proposed a ‘holistic schema

matching' approach that can match many schemas at the same time and find all matchings at once. Another similar approach [Saleem 2008] creates a mediated schema tree from a large set of input XML schema trees and defines matchings from the contributing schema to the mediated schema.

Alternatively, one can address the performance of computationally intensive similarity analysis in large-scale schema matching by parallel and distributed computing technologies. The parallel computing is a computation technology in which multiple concurrent processes work simultaneously and cooperate with each other for a single task, while the distributed computing deals with the development of applications that execute on different computers interconnected by networks. Thus, the parallel and distributed computing can refer to a computation technology in which many calculations are carried out simultaneously, operating on multiple computers interconnected by networks for a single task. The parallel and distributed computing in local networks is also called cluster computing and called grid computing in wide-area networks.

Many parallel and distributed computing technologies have been introduced [Asanovic 2006]. There is a well-known cluster computing technology called Hadoop[2] [Borthaku 2007 and Dean 2004] which is a Java software framework that supports data intensive distributed applications. Hadoop uses a new programming model called Map/Reduce for processing and generating large data sets. This platform al-

[2] http://lucene.apache.org/hadoop

lows programmers without any experience with parallel and distributed systems to utilize easily the resources of a large distributed system.

Table II.1. Comparison of Hadoop, Globus toolkit and MPJ

	Hadoop	Globus toolkit	MPJ
SW requirement	Java, SSHD[3]	Java, Ant[4]	Java
System Setup	System-specific	System-independent	System-independent
Security	SSH[5]	WS-Security[6]	No support
Data manage	DFS[7]	GridFTP[8]	No support
Type	Clustering	Grid	Grid

For grid computing technology, Globus Alliance[9] provides an open source toolkit called Globus Toolkit[10] [Foster 1997]. The Globus Toolkit makes extensive use of Web Services to define interfaces and structures of its components, which provide flexible, extensible, and widely-adopted XML-based mechanisms for describing, discovering, and invoking network services. The grid computing can also be imple-

[3] SSHD (SSH Daemon) is a software that supports Secure Shell or SSH connections

[4] Apache Ant is a Java library and command-line tool for automating software build processes. Available at http://ant.apache.org/

[5] Secure Shell or SSH is a network protocol that allows data to be exchanged using a secure channel between two network devices.

[6] WS-Security is a flexible and feature-rich extension to SOAP to apply security to web services.

[7] DFS (Distributed File System) is a file system that allows access to files from multiple hosts sharing via a computer network.

[8] GridFTP is an extension of the standard File Transfer Protocol (FTP) for use with Grid computing.

[9] http://www.globus.org/

[10] http://www.globus.org/toolkit/

mented using the Message Passing Interface (MPI) [Asanovic 2006] standard which is an Application Programming Interface (API) specification that allows many computers to communicate with one another. MPI-like Message Passing for Java (MPJ)[11] [Carpenter 2000] provides a Java software toolkit for the MPI standard. These tools have different technical backgrounds and performance trade-offs. Table II.1 shows different features of Hadoop, Globus Toolkit and MPJ.

The Hadoop and Globus Toolkit have advantages for easy handling of a large distributed system. However, they require extra software installation and management for clusters and Web Services, respectively. On the other hand, MPJ requires a simple environment configuration and programming architecture, but does not provide any security measures or data management functionalities.

[11] http://mpj-express.org/

CHAPTER III

XML SCHEMA MATCHING PROBLEM

This chapter provides a formal definition of XML schema matching problem. We also specify the assumptions that underlie our proposed methods.

III.1. Schema Matching, Mapping, Merging and Reuse

Schema matching, mapping, merging, and reuse have been developed as means for integrating heterogeneous e-Business operations between and within organizations. These techniques involve the identification of data elements that are semantically related among different schemas.

The terms "matching" and "mapping" are often used interchangeably. In this dissertation, we differentiate the two terms as follows. The term "match" or "matching" refers to an activity of identifying that two objects are semantically related whereas the term "map" or "mapping" refers to the transformations between the semantically related two objects [Bellahsene 2011]. For example, the output of matching may include statements such as "possible matches for Field *p* in specification *A* are Field *q*, *r*, and *s* in specification *B*" or "Field *x* in specification *A* could match with Field *y* in specification *B* with a similarity *Sim*", whereas the output of mapping may include statements such as "Field *p* in specification *A* is mapped to Field *q* in specification *B*" or "Field *x* in specification *A* is decomposed into fields *y* and *z* in specification *B*". In

other contexts, the term “map” or “mapping” may have a broader scope to include “matching” activity as well (i.e., matching then mapping).

The term “reuse” refers to the exercise that looks for standard message specifications to use in an integration project. This exercise typically involves a cycle of search, match, map, extend, and restrict activities. In the search activity, integrators look for message specifications (and/or their components) that are appropriate for the integration requirements. These requirements are typically manifested in other specifications. If some message specifications and/or their components are discovered, match and map usually occur to identify gaps between the requirements and the discovered artifacts. With the objective to close the gaps, the “extend” activity is usually followed, which may involve identifying other existing components or defining new components as an extension and/or composition of existing components and/or new components. The “restrict” activity usually occurs after requirements have been satisfied. The objective of this activity is to document some extraneous components that are inherited from reused specifications or components that are unnecessary in regard to the integration requirements. Some documentation is involved in restricting additional structural information (e.g., cardinality).

The term “merging” is also related to the term “matching” and “mapping”, and is similar to the idea of “reuse”. Merging is an exercise in which integrators attempt to combine two or more specifications into a single specification. This exercise typically involves a cycle of search, match, map, copy, delete, rename, and reorganization activities. Simply by the number of steps involved, we may predict that the merging

exercise is the most time consuming compared with the other activities described above. Gaps identified in matching and mapping activities may be closed by simply copying components from one specification to another and deleting and renaming any duplicates. There are cases in which gaps are better closed by the reorganization of several similar components. Reorganization involves creating new components that are a composite or extension of other components and deleting some components. In other words, merging involves additional design decisions beyond matching, mapping and reuse.

It should be noted that it is easy to confuse the match and map activities, particularly in the context of automation. This is because the matching algorithm often tries to perform some mappings. The matching is regarded in this research as an "approximate mapping" activity which is to provide a set of mapping candidates in a target schema based on the semantic similarity measures for each element in a source schema, without additional activities such as merging and reuse. All other schema integration activities (i.e., "merging", "reuse", "extend", and "restrict") are not considered in this dissertation.

III.2. Semantic-based XML Schema Matching

We define a schema matching as an activity of identifying *semantic* relationships between elements (or attributes) of two schemas. This is often called more precisely *semantic-based* schema matching as the activity results in identified correspondences of *semantically close elements* in different schemas. In a general schema matching problem, we assume that schema designers fully understand the meaning of

contents in their schemas. Hence, the *semantic-based* schema matching assumes the intention to capture the actual meaning of concepts and semantic relationships as planned by designers, and identifies how two or more semantic concepts are similar in the respective schema representations based on some similarity measuring functions.

As we mentioned before, XML schema does not provide means to formally define the semantics, but some implicit *semantics* can be obtained from contextual information within XML schemas. In this research, we employ two types of contextual information – structural and linguistic contexts – based on the assumption that the more similar the structural and linguistic contexts of two schemas, the more semantically similar the two schemas are.

To focus on those contexts in XML schemas, we propose to model XML schemas as labeled trees, called *schema trees*, where nodes and edges relating nodes reflect the structural contexts, and English words in labels of nodes reflect the linguistic contexts. It helps to make the contexts obvious as we describe the matching process.

III.3. General Schema Matching Architecture

To understand the schema matching task, we first identify the type of information that the matching task takes as input and produces as output. According to Shvaiko & Euzenat (2005), the input in schema matching is a pair of schemas that are not homogenous. The output is given in the form of matchings which isolate elements

from these two schemas that have a semantic relationship. The general concept of schema matching is illustrated in Figure III.1 below.

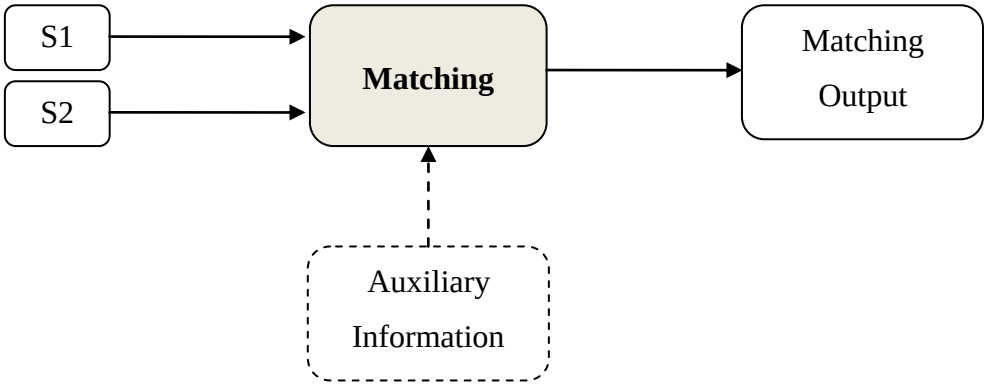


Figure III.1. General schema matching architecture.

In general, the input information of a matching problem may include any type of knowledge about schemas to be matched, such as their instances and their domains. In the proposed matching approaches, the primary input data consists of two schemas (i.e., source and target XML schemas) defined above as S1 and S2. The input schemas contain different information types such as names, descriptions, data types, and constraints, but we only consider the information of labels in elements and attributes and their relationships. They are modeled as schema trees for the matching process. To analyze the linguistic contexts in the schema tree, we only consider English words (including their abbreviations and acronyms) and also concatenated words (e.g., Address, PurchaseOrder, ShipLocation) in the labels of nodes. We use a preprocessor to handle abbreviations, acronyms, stop words, and any other non-English words (see Appendices B and C). The preprocessor not only removes unnecessary data such as stop words, separators, and non-characters, but also converts abbreviations and acronyms into the original forms.

To increase performance of semantic matching, several matching algorithms often require auxiliary information such as thesauri and dictionaries. The main auxiliary information used in this research is WordNet, an electronic lexical database where relations such as homonymy are available to relate English word meanings [Miller 1995]. Other domain-specific dictionaries can be used in the case where used words do not belong to WordNet.

The outcome of the matching process typically provides similarity score ranging in [0,1] interval, where 1 (or 0) means that two matching pairs are exactly the same (or totally different). In the case of multiple matching results, we propose two approaches. The first is to produce a collection of *k* best matching candidates for each matching pair, called top-*k* matching, ranked according to their semantic similarity scores. The second is to produce all possible matching pairs based on the given matching cardinality. An element from one schema can participate in zero, one, or several matching correspondences with elements from the other schemas (hereafter called *1-to-1*, *1-to-n* or *n-to-1*, or *n-to-m*, respectively). Note that a top-*k* matching is a special type of *1-to-n* matching, because an element from one schema can be matched to *k* elements from other schemas at most. In this dissertation, the proposed *c-matching* approach produces top-*k* matching as an output, whereas the proposed *a-matching* approach produces *1-to-1* pair-wise matching.

III.4. Schema Tree and Matching Model

We formally present a labeled tree model, also called *schema tree*, used to represent XML schemas as follows:

Definition III.1: (Labeled tree or schema tree) A labeled tree is a rooted tree, denoted $G = (N, E)$, that has labels associated with each node (vertex), where

1) $N = \{n_1, n_2, ..., n_n\}$ is a finite set of nodes (vertices), each of which has zero or more child nodes and at most one parent node. Each node corresponds to either an element or an attribute in the schema, with a label that is a string encoding English words, concatenation of words or their abbreviations for describing the meaning of elements or attributes.

2) $E = \{(n_i, n_j) \mid n_i, n_j \in N\}$ is a finite set of edges, each of which denotes the relationship between two nodes (i.e., two elements, two attributes, or element and attribute) where n_i is called the parent of n_j and n_j is called the child of n_i. Typically a child node is a subcomponent of its parent.

Nodes with the same label may appear in more than one place in the labeled tree (e.g., the child nodes of 'Address' and 'DeliveryTo' nodes in Figure I.1 (b)). Each node is uniquely identified by its *path context* from the root to the node. The *path context* is defined as follows:

Definition III.2: (Path contexts or path) A path from the root node n_1 to node n_k is a sequence of nodes $n_1, n_2, ..., n_k \in N$, and for any two consecutive nodes, n_i and n_{i+1} $(1 \le i \le k-1)$, there exists an edge $e_i(n_i, n_{i+1}) \in E$. The length of a path is the total number of edges on the path; that is, $k - 1$ for the path $P = (n_1, n_2, ..., n_k)$, denoted as length $|P| = k\text{-}1$.

The labeled tree has three types of nodes: root, leaf (or atomic), and inner nodes. Given a schema tree $G = (N, E)$, they are defined as follows:

Definition III.3: (Root node) If a node $n_i \in N$ does not have a parent, then n_i is called the *root node* of the schema tree G, denoted R^G. The *root note* is unique in *G.*

Definition III.4: (Leaf node or atomic node) Any node n_i which has no children is called a *leaf node*. The *leaf nodes* are also called *atomic nodes* because they are the smallest unit and cannot be further divided. The set of all *atomic nodes* is denoted A^G.

Definition III.5: (Inner node) Any node n_i which is neither *root node* nor *atomic node* is called an *inner node*. The set of all inner nodes is denoted as I^G.

In Chapter I, we defined two types of schemas: component schema and document schema. Component schema can be represented as a set of labeled trees, each of which represents the global type component in the schema, whereas document schema can be represented as a single labeled tree. They can be defined as follows:

Definition III.6: (Component schema) A component schema $S_c = \{G_{c1}, G_{c2}, \ldots, G_{cn}\}$ is a finite set of labeled trees, each of which represents the labeled tree structure of a global type component defined in the schema S_c. Here, n is the number of global type components in the schema S_c and $G_{ci} = (N_{ci}, E_{ci})$ is a labeled tree whose *root node* is i^{th} global type component in the schema S_c, where $1 \le i \le n$. Nodes in the node set N_{ci} represent either elements or attributes in the i^{th} global type component of the schema S_c.

Definition III.7: (Document schema) A document schema S_d is a single labeled tree $G_d = (N_d, E_d)$. Nodes in the node set N_d represent either elements or attributes in the schema S_d.

For *a-matching* between two document schemas, we focus on atomic nodes. The *path contexts* of atomic nodes can be defined as follows:

Definition III.8: (Atomic node path context) A path from node n_1 to node n_k, is an atomic node path, if and only if the starting node n_1 is a root and n_k is an atomic node.

For *c-matching*, we consider two component schemas $S_{c1} = \{G_{c11}, G_{c12}, \ldots, G_{c1m}\}$ and $S_{c2} = \{G_{c21}, G_{c22}, \ldots, G_{c2n}\}$, where S_{c1} is a source schema which defines *m* number of global type components and S_{c2} is a target schema which defines *n* number of global type components. The matching is to find the semantic correspondences between two sets of labeled trees S_{c1} and S_{c2}.

For *a-matching*, we consider two document schemas S_{d1} and S_{d2}. S_{d1} is a source schema denoted $G_{d1} = (N_{d1}, E_{d1})$ and S_{d2} is a target schema denoted $G_{d2} = (N_{d2}, E_{d2})$. The matching task is to find the semantic correspondences between two sets of atomic nodes $A^{G_{d1}}$ and $A^{G_{d2}}$ (see Definition III.4), considering the structural relationships (two sets of edges E_{d1} and E_{d2}) and the linguistic information in labels associated with nodes in N_{d1} and N_{d2}.

The simplest matching unit, called *matching pair*, is a pair of two objects that are matched with certain similarity measure. For example, a matching pair in *c-matching* is a pair of two global type components, where one comes from the source and another from the target component schemas. Whereas, a matching pair in *a-matching* is a pair of two atomic nodes, where one comes from the source and another from the target document schemas. The schema matching usually compares all possible matching pairs, called *pair-wise comparison*, and identifies a set of matching pairs, called *pair-wise matching*.

In real world applications, one data element in the source schema may match more than one element in the target schema, which is referred to as *1-to-many pair-wise matching*. One can also find *many-to-1 pair-wise matching* or even *many-to-many pair-wise matching*. The simplest case of these matchings is *1-to-1 pair-wise matching* which is defined as follows:

Definition III.9: (*1-to-1* pair-wise matching or *1-to-1* matching) Let $m \subseteq U \times V$ be a matching of two sets *U* and *V*. Matching *m* is said to be *1-to-1 pair-wise matching* if for every $\{(a,b),(c,d)\} \subseteq m$, $a \neq c$ if and only if $b \neq d$.

Based on *1-to-1* pair-wise matching result of two sets of data *U* and *V*, we can calculate the overall similarity between *U* and *V* by using the average similarity score of all matching pairs. The overall matching similarity can be calculated as follows:

$$Sim_{overall}(U,V) = \frac{2}{|U|+|V|}\sum_{i=1}^{k} sim(u_i, v_i). \qquad \text{(III.1)}$$

The top-*k* matching is a special type of *1-to-many* pair-wise matching, ranked by the similarity scores. The top-*k* matching can be defined as follows:

Definition III.10: (top-*k* matching) Given two object sets *U* and *V*, and a similarity score function *Sim*(*u*, *v*) for any pair (*u*, *v*) ($u \in U$ and $v \in V$), top-*k* matching for an object $u \in U$ is the *k* number of matching pairs (u, v_i), where $i = 1...k$ and $v_i \in V$, whose similarity scores are the first *k* largest.

III.5. Assumptions

The following assumptions are made to develop the proposed approaches.

Assumption 1: The more similar the structural and linguistic contexts of two schemas, the more semantically similar the two schemas are.

Assumption 2: The data models are in forms of the XML schema which conforms to the standard XML Schema specification [W3C 2001a; W3C 2001b; and W3C 2001c] recommended by W3C[12]. All other forms of schema models, e.g., UML[13], EDI-FACT[14], and database schemes are not considered in this dissertation.

Assumption 3: For *a-matching*, only *1-to-1* pair-wise matching is considered, even though *1-to-many*, *many-to-1*, and *many-to-many* matchings may exist.

---

[12] World Wide Web Consortium (W3C). http://www.w3.org/

[13] Unified Modeling Language (UML)

[14] Electronic Data Interchange For Administration, Commerce, and Transport (EDIFACT)

Assumption 4: XML schemas should follow the recommended naming rule provided in Appendix A. One of the most important rules is that labels of elements and attributes should be composed of words in the English language, using the primary English spellings provided in WordNet 3.0[15] [Miller 1995].

Assumption 5: The schema matching results manually generated by human engineers used for the validation of the proposed approaches are considered free of matching errors.

[15] Available at http://wordnet.princeton.edu/

CHAPTER IV

A LAYERED APPROACH FOR XML COMPONENT SCHEMA MATCHING

This chapter describes the proposed *c-matching* algorithm for analyzing schema matching between two XML component schemas using semantic similarity measures. Several key innovations are introduced to increase utilization of available semantic information. These innovations include: 1) a layered structure analysis of XML schemas, 2) layer-specific semantic similarity measures, and 3) an efficient semantic similarity analysis using parallel and distributed computing technologies.

IV.1. Layered Semantic Structure of XML Schema

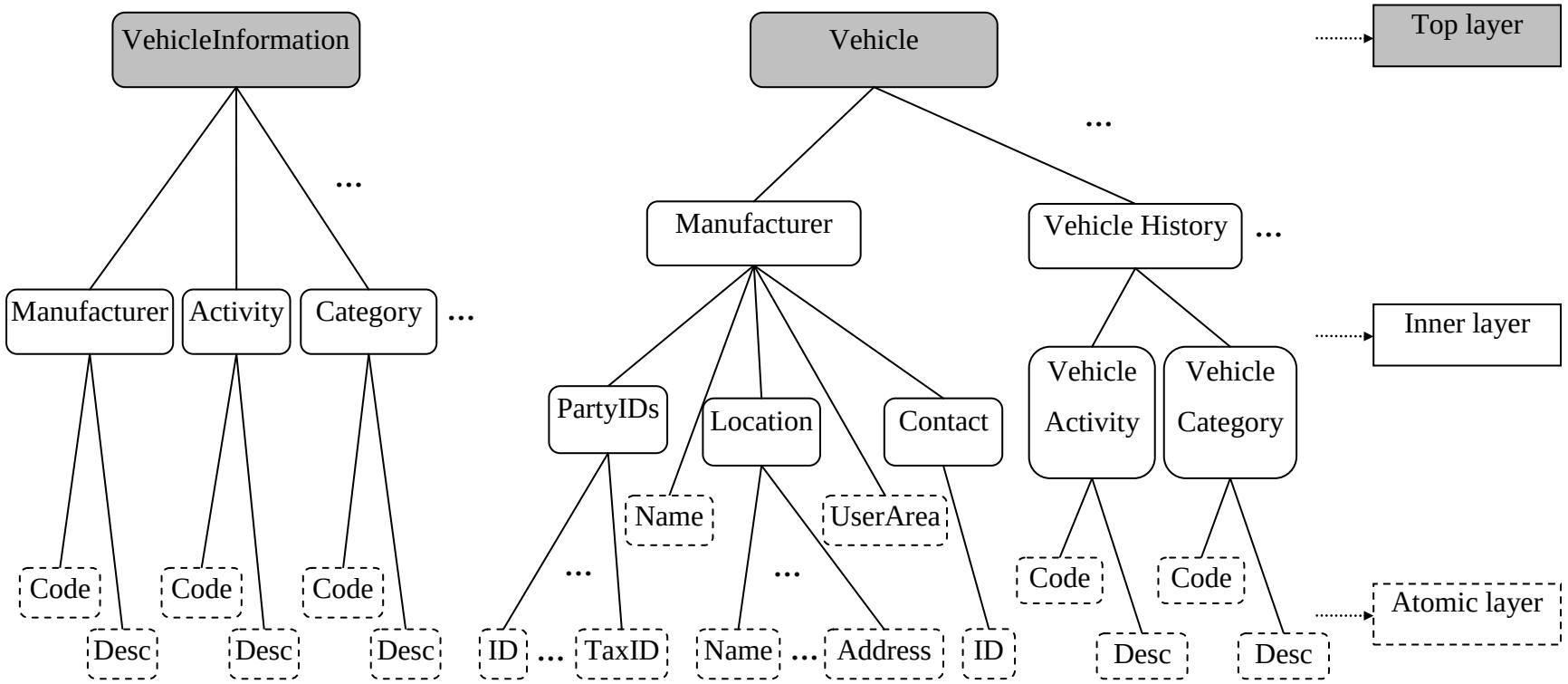
An XML component schema defines a set of global elements, each of which can be represented as a labeled tree with a set of linked nodes (see Definition III.6). Each node in the tree has zero or more child nodes. As defined in Definition III.3 – III.5, there are three types of nodes: 1) the root, 2) the leaf (atomic), and 3) the inner nodes (those with both a parent and children).

Each tree can be divided into three layers: 1) the *top layer* (containing the root of the tree), 2) the *atomic layer* (containing atomic nodes), and 3) the *inner layer* (containing inner nodes). Note that some trees may have empty inner layer, whereas

others may have only one node that is considered to be in both the top and atomic layers.

Each layer typically captures the semantics of a global element from a distinct perspective. Through its label and namespace, a top layer node specifies the data object that the global element is intended to describe. The atomic layer includes the atomic nodes (e.g., attributes, simpleType, and simpleContent) that the designers felt were necessary to describe the global element (the root). The inner layer provides the structural information of the global element by specifying how the atomic elements are grouped into inner nodes and, eventually, into the global element. The linguistic information in the labels of both atomic and inner nodes may also help to qualify the semantics of the global element.

Figure IV.1 shows the labeled graph examples of two different XML schemas describing ‘vehicle’ (e.g., maker, model, model year, dealer information, mileage at failure, mileage at repair, etc). The two schemas were defined by two different workgroups at the Automotive Industry Action Group (AIAG): (a) Truck and Heavy Equipment (T&HE) and (b) AIAG Resource schemas [AIAG 2008]. Both intended to describe the same object ‘vehicle’. However they have different labels (names) and different structures.



(a) T&HE's VehicleInformation

(b) AIAG's Vehicle

Figure IV.1. Three layers of two XML schemas.

The labels in the top layer nodes of the two schemas indicate that both are intended to represent the "vehicle" object with similar conceptualizations. However, the designers' thoughts differ in regard to what atomic elements are needed. In Figure IV.1 (a), the root element 'VehicleInformation' only includes the generic atomic elements such as the 'Code' and the 'Desc', whereas in Figure IV.1 (b), the root element 'Vehicle' includes more specific atomic elements such as 'TaxID', 'Address', 'Name', and so on. They also differ on how these atomic elements should be organized (see their different inner layers). Moreover, the same set of ingredients (atomic nodes) can produce elements of different semantics depending on how they are structured or packaged (i.e., the identity of the top layer node). For example, several party elements (CustomerParty, DealerParty, and SellingParty) defined in AIAG schema all contain the same atomic and inner nodes but are intended for semantically different data objects.

IV.2. Semantic Similarity Algorithms for the Layered Approach

The complex relationship among nodes at different layers requires layer-specific semantic analysis tools and a mechanism to combine these layer-based similarities. For this reason, we have developed three similarity measures. The first one, called *atomic-layer similarity*, measures the similarity between atomic layers of two global elements. The second one, called *label similarity*, measures the similarity between two labels (names). The top-layer only uses the label similarity. The last one, called *inner-layer similarity*, measures the similarity between inner layers of two global elements. These three measures and the process for their combination are described in the following subsections.

IV.2.1. Atomic-layer similarity measures

Not every atomic node is equal in determining semantic similarity. The sharing of an atomic node that is widely used by two elements is not as strong as in an indication of similarity compared with the sharing of a rarely used atomic node [Lin 1998 and Resnik 1995]. To account for the degree of importance of individual atomic nodes, an IC-based measure for atomic layer similarity has been developed.

Specifically, let $A(x)$ and $A(y)$ denote the sets of atomic nodes of global elements x and y, respectively. Then, the atomic level similarity between x and y is defined as follows:

$$Sim_A(A(x), A(y)) = 2 \times \frac{\sum_{c_k \in A(x) \cap A(y)} I(c_k)}{\sum_{c_i \in A(x)} I(c_i) + \sum_{c_j \in A(y)} I(c_j)}. \quad \text{(IV.1)}$$

Eq. IV.1 can be seen as a combination of the IC-based measure and the Jaccard coefficient (see Eqs. II.2 and II.5). The probability of each atomic node is taken at its frequency using any corpus related to the source and target schemas. For instance, we can use a corpus formed by all node labels in both the source and target schemas or by all words extracted from some domain specific documents.

Eq. IV.1 is based on the assumption that the source and target schemas share a significant number of atomic nodes. Two atomic nodes can be treated as either completely similar (with a similarity score of 1) if they have the same label or completely dissimilar (with a similarity score of 0) if they do not. Eq. IV.1 can be generalized for use in situations where similarity scores among many atomic node pairs are between

0 and 1 [Peng 2006]. For that, we partition $A(x)$ into two sets: $A_1(x)$ contains those components of *x* that have similar counterparts in $A(y)$ (i.e., with non-zero pair-wise similarity), and $A_2(x) = A(x) - A_1(x)$. Similarly, we partition $A(y)$ to $A_1(y)$ and $A_2(y)$. For every $c_i \in A(x)$, we define its map to $A(y)$ as follows:

$$m(c_i) = \max_{c_j \in A(y)} Sim(c_i, c_j) > 0 . \quad \text{(IV.2)}$$

Then the similarity of *x* to *y* is given as

$$Sim_A(x, y) = \frac{\Sigma_{c_i \in A_1(x)} Sim(c_i, m(c_i)) \cdot I(c_i)}{\Sigma_{c_i \in A_1(x) \cup A_2(x) \cup A_2(y)} I(c_i)} . \quad \text{(IV.3)}$$

The numerator of Eq. IV.3 is the sum of information content of the similar components of *x* and *y*, weighted by the similarities of individual pairs. It measures the commonality under the non-exact matches of components. The denominator, as a normalization factor, is the sum of information content of all components of *x* and *y*.

IV.2.2. Label similarity measures

The label or name *x* of a node is a word or concatenation of words (or their abbreviations). There is one approach for label similarity measure known as the string-based metric. It computes similarity between two labels. As we discussed in Chapter II.1.1, the string-based metric can be enhanced using language-based metric and linguistic resources. Therefore, before two labels are compared, a pre-process called "label normalization" (denoted as *L*(*x*)) is conducted to obtain full words from the concatenations and abbreviations. For example, *L* (VehicleInformation) = {vehicle,

information}. To better ascertain the semantics of these words and to deal with the problem of synonyms, each word is expanded using its description found in WordNet. The descriptions can be also obtained from a variety types of sources such as schema annotation, web search, business related documentation, and so on.

The descriptions of all the words in $L(x)$ are then put together under two constraints to form a vector of words, $W(x)$. First, for a fair comparison, $W(x)$ should be independent of the lengths of descriptions, which vary greatly from word to word. One possible way to achieve this is to normalize *W(x)* by making the length of descriptions for all *W(x)* the same. Secondly, words in $L(x)$ are not equally important in defining x's semantics (e.g., "vehicle" is certainly more important than "information" in the label "VehicleInformation"). There are several ways to address this issue such as noun phrase analysis from natural language processing. An easiest way is to duplicate more important words and to truncate less important words in the *L(x)*. The importance of the words can be obtained from its information contents. Finally, the similarity of labels x and y can be measured by a variety of string-level similarity measures (see Chapter II.1.1) on the normalized $W(x)$ and $W(y)$. We use the cosine similarity measure for our experiments.

Another approach of label similarity is to obtain the semantics for each label using topic signature [Hovy 1999]. The topic signature is defined as a family of related terms:

$$TS = \{t, <(w_1, s_1)...(w_i, s_i)...>\}, \quad \text{(IV.4)}$$

where t is the topic (i.e. the target concept) and each w_i is a word associated with the topic, with strength s_i.

To obtain the topic signature for a specific word, we retrieved the most frequent words from several document sources such as Google, dictionary and domain specific documents, after removing stop-words (see Appendix C). Then, we could collect all related terms to the labels and compute the similarity of labels x and y using string-level similarity measures (e.g., cosine similarity measure). This approach is alternatively applied to compute the label similarities for top layer nodes (i.e., Sim_T).

IV.2.3. Inner-layer similarity measures

Any type of structure-level similarity measure can be used to compute inner-layer similarity. However, two schemas (i.e., AIAG and T&HE) used for experiments have very different labels and structures in their inner-layers (see Figure IV.1). Therefore, currently we only extend the label similarity measure for the inner-layer's similarity measures (i.e., Sim_I). In this case, x (and y) is the union of labels of all inner nodes. Other structure-level similarity measures will be investigated as further research.

IV.2.4. Combined similarity score

A variety of algorithms for combining individual similarity measures for the three layers (Sim_A, Sim_T, Sim_I) can be used, such as average(a, b, c), max(a, b, c), additive ($a + b + c - ab - bc - ac + abc = 1 - (1 - a)(1 - b)(1 - c)$), and weighted sum. The weighted sum can be defined as follows:

$$Sim(x, y) = w_A Sim_A + w_T Sim_T + w_I Sim_I \,, \tag{IV.5}$$

where $w_A + w_T + w_I = 1$.

The weighted sum has the advantage of allowing the adjustment of weights to best reflect the importance of measures at individual layers. However, finding the best weights is a challenge. Currently, the weights are obtained from the domain experts or learned from human semantic matching data. This is another area for future research.

IV.3. Experiments and Results

IV.3.1. Experimental data

To test and evaluate the proposed approach, we obtained schemas and manual matching data from two different workgroups at the Automotive Industry Action Group (AIAG), the AIAG Resource schema and the Truck and Heavy Equipment (T&HE) schema as the target and source, respectively. Table IV.1 summarizes the characteristics of the AIAG Resource and T&HE XML schemas.

Table IV.1. Characteristics of AIAG and T&HE schemas

	AIAG Resource Schema		T&HE Schema	
Total # of atoms	67688		53812	
# of distinct atoms	793		825	
	non-OAG	OAG	non-OAG	OAG
	90	703	119	706

Both schemas are based on the OAG common core component schema [OAG 2002] and have many overlapping concepts. More than 70% of atomic nodes in the

two schemas are defined in the OAG schema. However, they define certain elements of similar concepts quite differently as shown in Figure IV.1. At the component level there are the set of 139 global (top) elements defined in the T&HE schema, that need to be mapped into the set of 145 global (top) elements of the AIAG Resource schema. The semantic distances of 139 x 145 (~ 20,000) pairs of elements thus need to be examined. Roughly 140 human hours were spent to map 49 of the 139 top elements in T&HE to those in AIAG Resource schema. Substantial amount of time is further required to merge at the document (message) level. This is an indication that manual matching is very time consuming.

The 49 manual matchings produced by human integrators were used as the basis to evaluate the performance of the proposed approach. For each of the 49 T&HE global elements, the matching algorithm recommends the top-*k* matching (the most similar AIAG elements) based on a similarity measure. The performance of the proposed approach has been evaluated using a set rather than a single recommendation, because the objective is not to fully automate the process, but rather to assist a human expert. A recommendation is considered successful (called hit) if it contains the manual matching from the human integrators.

IV.3.2. Performance measures for evaluation

The matching algorithm generates the top-*k* matching for each recommendation (the *k* number of best matchings in the recommendation) ranked according to their semantic similarity scores. To identify the optimal size of *k* in this experiment, we analyzed the matching results by different size of top-*k* with certain parameters (i.e.,

the same parameters used by the combined measure of the experiment #1 in the next section). Figure IV.2 shows the matching results from top-1 to top-10 matchings.

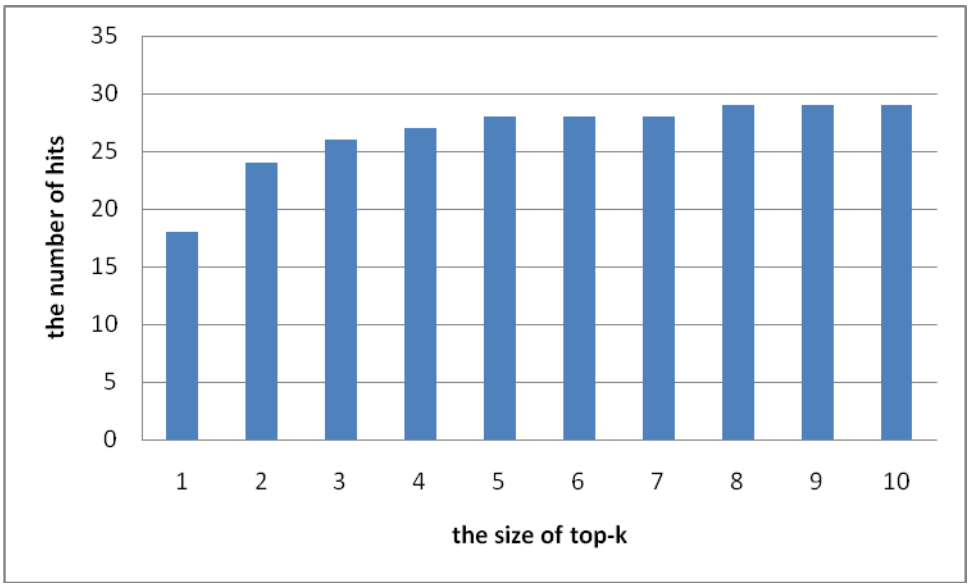


Figure IV.2. Matching results by different size of top-*k*.

Most of hits were identified by the top-five recommendations. Therefore, only top-5 matching (i.e., the five most similar) results were used to analyze the performance of our experiments.

The performance of our approach is measured by the *Recall*, which is the ratio of the number of hits and the number of the correct matchings (see Eq. II.8.). Because our matching results contain *k* number of the best matchings for each hit with different similarities, the *Recall* may provide an inaccurate measure without considering the ranking of the target object within a recommendation. We employed a new *Recall* performance measure, called *Weighted-Sum Recall* (*WSR*), which gives each hit a

weight according to the ranking of the correct matching in the recommendation. The *WSR* can be defined as follows.

Definition IV.1: (*Weighted-Sum Recall*) Let k be the size of the best matchings in the recommendations of top-k matching, and w_i be a weight assigned to a hit in which the correct matching is ranked as i^{th} best in the recommendation.

$$\text{WSR} = \frac{\sum_{i=1}^{k}(w_i \times \text{number of hits with correct matching ranked as } i^{\text{th}} \text{ best})}{\text{total number of correct matching}} \qquad \text{(IV.6)}$$

In this experiment, $k = 5$ and to give more weights to the best ranked hits, the following weights were used: 1.0 (w_1), 0.8 (w_2), 0.6 (w_3), 0.4 (w_4), and 0.2 (w_5).

IV.3.3. Results analysis

A prototype system is implemented. The system not only computes Sim_A, Sim_T, Sim_I as given in Eqs. IV.1 and IV.3 but also supports several combination rules, including Eq. IV.5. A variety of experiments has been conducted with different parameters. To combine individual similarity measures for the three layers (Sim_A, Sim_T, Sim_I), we used the weighted sum algorithm by Eq. IV.5 with the balanced weights for all layers (i.e., $w_A = w_T = w_I$). As described in Chapter IV.2.2, two types of semantic sources were used for top-layer measure (i.e., WordNet and topic signature). For the first experiment, WordNet semantic source was used. Results from various similarity measures (individual and combined) were obtained and reported in the table below.

Table IV.2. Experiment #1 Results

Similarity measure	*# of hits*
Sim_T	35
Sim_I	8
Sim_A	23
Weighted sum	28

Evidently, atomic-layer and inner-layer measures alone produce poor results (with 22 and 8 hits, respectively). This is because, as discussed earlier, the same set of atomic and inner nodes can be used to produce several semantically different elements (just like the same ingredients can be cooked into several kinds of dishes). The top-layer measure produces the best matching result (with 35 hits), even better than the combined measure by the weighted sum (with 28 hits). It suggests that more weight needs to be given to top layer label similarities. The combination weights can be adjusted according to the ratio of the number of hits in each individual measure (approximately with 0.53, 0.12, and 0.35, respectively). The adjusted combination weights increased the number of hits to 31 matchings, but it is still worse than top-layer measure (with 35 hits). Detailed matching results of this second experiment are shown in Table IV.3.

In terms of the total number of hits by top-5 matching, the top-layer measure is still best (with 35 hits). However, in terms of the numbers of hits by matchings ranked as 1st and 2nd best in the recommendations, and *WSR*, the combined measure produces better results than top-layer measure (with 27 and 20 hits, and 0.56 and 0.54 *WSR*, respectively).

Table IV.3. Experiment #2 Results

		Sim_T	Sim_I	Sim_A	**weighted sum**
# of hits	1st best	19	2	8	**18**
	2nd best	1	1	8	**9**
	3rd best	7	3	3	**3**
	4th best	4	1	1	**0**
	5th best	4	1	2	**1**
	top-5 total	35	8	22	**31**
	Recall	0.71	0.16	0.45	**0.63**
Weighted-Sum Recall		**0.54**	**0.11**	**0.35**	**0.56**

This conclusion is also supported by the third experiment where the topic signature is used for top-layer measure. Table IV.4 shows the results of individual and combined measures for the third experiment.

Table IV.4. Experiment #3 Results

		Sim_T	Sim_I	Sim_A	**weighted sum**
# of hits	1st best	21	2	8	**21**
	2nd best	4	1	8	**4**
	3rd best	4	3	3	**6**
	4th best	0	1	1	**3**
	5th best	2	1	2	**1**
	top-5 total	31	8	22	**35**
	Recall	0.63	0.16	0.45	**0.71**
Weighted-Sum Recall		**0.55**	**0.11**	**0.35**	**0.60**

In this experiment, the combined measure has clearly better result than others in terms of top-5 matching and WSR. The weighted sum leads to roughly 71% correct matchings representing 35 of the 49 manual matchings. The results of these experi-

ments are encouraging, considering the difficulty of the problem for even experienced integrators.

IV.4. Large-Scale Schema Matching by Parallel and Distributed Computing

This section presents a schema matching architecture using a grid computing technology called MPJ, to enhance the computational efficiency of the proposed XML schema matching algorithm.

IV.4.1. General architecture

The layered approach can be expanded to improve the computational efficiency by using grid computing technologies. We have developed a Grid computing architecture for XML schema matching based on Service-Oriented Architecture (GX-SOA) [16]. This system can assist not only the proposed layered approach of schema matching, but also the e-Business vendors to employ easily the functionality of semantic similarity analysis. Figure IV.3 shows an overview of the GX-SOA.

[16] Service-oriented Architecture (SoA) is a flexible set of design principles used during the phases of systems development and integration in computing.

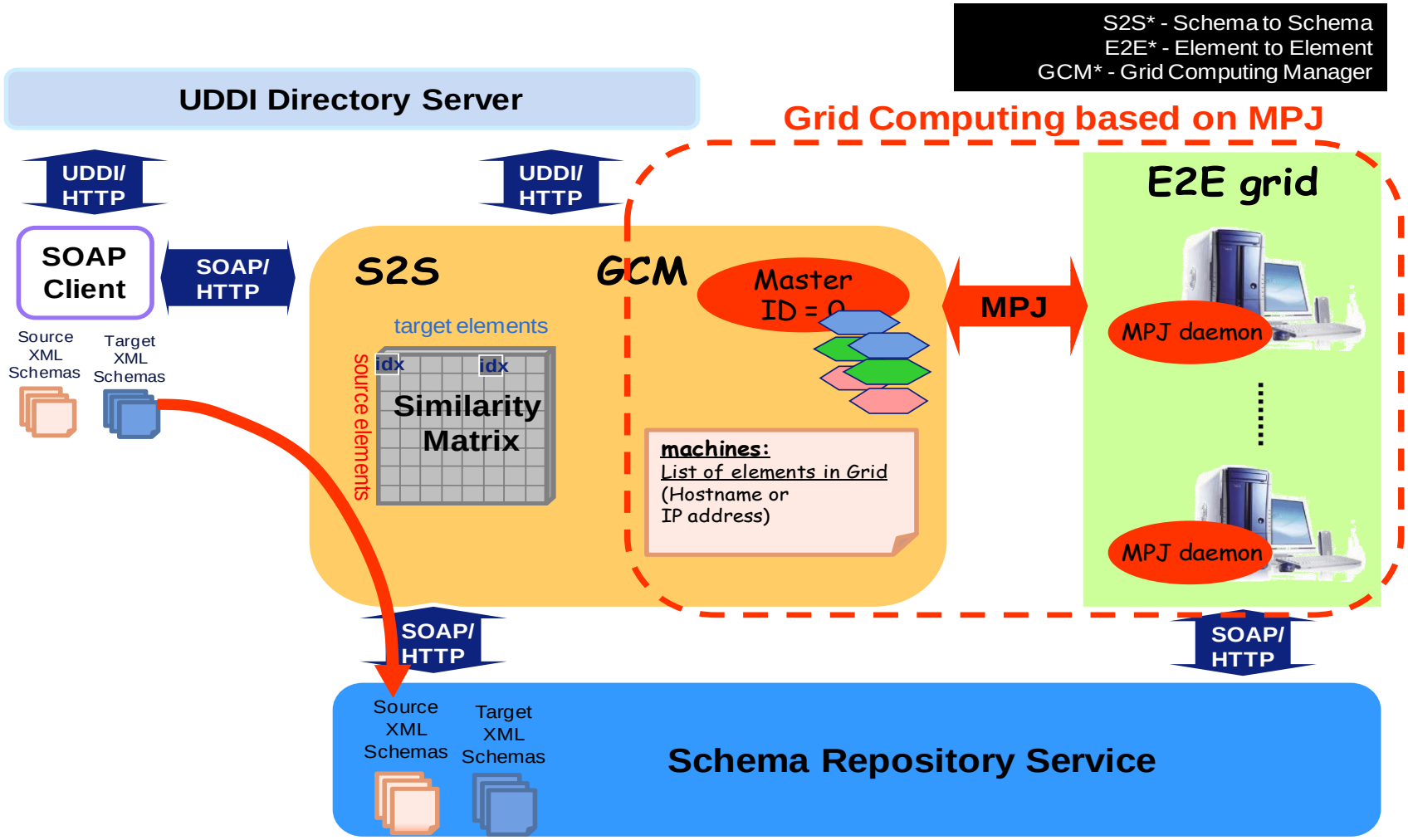


Figure IV.3. Overview of the GX-SOA.

The architecture consists of three main SOA components: a schema matching SOAP[17] client, Grid enhanced XML schema matching Web Services (GX-WS), and a Universal Description Discovery and Integration (UDDI)[18] directory service. The schema matching SOAP client can be any kind of software that uses the GX-WS as long as it supports messaging using SOAP 1.1[19] or 1.2[20] specifications.

First, the GX-WS publishes its own Web Services Description Language (WSDL)[21] service description at a public UDDI directory server. Any SOAP client can find the WSDL service description of the GX-WS through the public UDDI directory server. Note that there are several supporting tools, such as AXIS2[22] and Tomcat[23], for generating a SOAP message generator/parser according to the given

---

[17] Simple Object Access Protocol (SOAP) is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks.

[18] Universal Description, Discovery and Integration (UDDI) is a platform-independent, Extensible Markup Language (XML)-based registry for businesses worldwide to list themselves on the Internet and a mechanism to register and locate web service applications.

[19] Available at http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[20] Available at http://www.w3.org/TR/soap12-part0/ and http://www.w3.org/TR/soap12-part1/

[21] Web Services Description Language (WSDL) is an XML-based language that provides a model for describing Web services.

[22] Apache AXIS2 is a core engine for Web Services. It not only provides the capability to add Web services interfaces to Web applications, but can also function as a standalone server application.

[23] Apache Tomcat is an open source servlet container developed by the Apache Software Foundation (ASF). Available at http://tomcat.apache.org/

WSDL. Finally, the schema matching SOAP client can invoke the GX-WS to request a schema matching analysis for a given source and target XML schemas.

The GX-WS consists of four components: a Schema-to-Schema (S2S) matching service, a Grid Computing Manager (GCM), an Element-to-Element (E2E) matching service, and a schema repository service. First, the S2S matching service is the main component providing an interface for SOAP clients to access the matching service. The S2S matching service produces a similarity matrix that contains the semantic similarities between all comparable source and target element pairs. Secondly, the GCM is a sub-component of the S2S matching service that initiates the grid computing network and assigns jobs to the grid cells, which are the E2E matching services. Thirdly, E2E matching services execute semantic similarity measures between two elements by using the given similarity matching algorithm. Last but not least, the schema repository service is a web service that manages XML schemas via a permanent repository.

IV.4.2. Use cases and scenarios

Figure IV.4 illustrates a use case model of GX-WS. Five actors are defined, such as a SOAP client, a UDDI directory service, a S2S matching service with GCM, an E2E matching service, and a schema repository service.

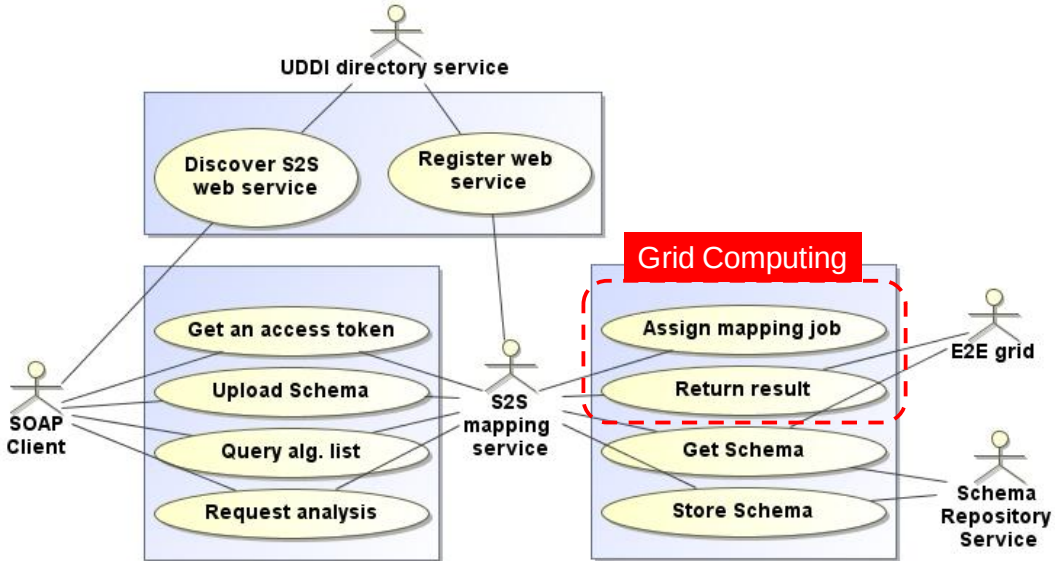


Figure IV.4. Use case diagram for GX-SOA.

The use case scenario is as follows: 1) S2S matching service publishes its WSDL service description on UDDI directory service; 2) a SOAP client invokes the schema matching analysis web services according to the WSDL service description of S2S matching service registered in UDDI; 3) S2S matching service creates a similarity matrix and distributes the similarity analysis jobs for every cell in the matrix to the E2E matching service with indices of the source and target elements to be analyzed; 5) E2E matching service computes the semantic similarity by using the given similarity algorithm with the source and target schemas obtained from the schema repository service; and 6) S2S matching service collects all similarity results and, finally, returns the matching target candidate elements for each element in the source schema based on their semantic similarities.

IV.4.3. MPJ implementation

The proposed GX-SOA uses a MPI-like Message Passing for Java (MPJ) software toolkit. Figure IV.5 shows the pseudo-code of MPJ implementation.

Algorithm *GCM*(*source*,*target*,*args*)

```
1.  initialize MPJ engine with args;
2.  machine_type:= MPJ.Type(); // Master or Slave
3.  num_or_machines:= MPJ.Size();
4.  if machine_type is Master then
5.    for (i = 0; i < source; i++)
6.      for (j = 0; j < target; j++)
7.        job_index:= { i, j }; k++;
8.        if k is smaller than num_or_machines then
9.          send initial job assign request to k^th machine
            with job_index;
10.       else
11.         wait until any machine completes the assigned job;
            if a machine completed the assigned job then
12.           simMatrix[job_index]:= retrieve the result
                                     from the machine;
13.         send new job assign request to the machine;
14.       end if;
15.     end if;
16.   end for;
17.   end for;
18.   finalize MPJ engine;
19.   return {simMatrix};
20. else if machine_type is Slave then
21.   wait until Master sends the job assign request;
22.   --- job processing ---
23.   finalize MPJ engine;
24.   return the result to Master machine;
25. end if;
```

Figure IV.5. Pseudo-code of the MPJ implementation.

The MPJ approach is well-suited to handle 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 to manage the tasks. The manager is called the "master," and the others the "slaves."

The first step to implement grid computing is to initialize the MPJ (lines 1 – 3). After that, the processors are divided into two communicators, with one processor as the master (lines 4 − 19) and the others as the slaves (lines 20 − 24). 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.

IV.4.4. Experiments and results

A prototype system with an example SOAP client was implemented using Eclipse[24], JDK 6[25], and the Google Web Toolkit[26] based on Tomcat[27] and AXIS2. Its performance was evaluated with the same sets of XML schemas (i.e., AIAG and T&HE).

---

[24] Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. Available at http://www.eclipse.org/

[25] Java Development Kit (JDK) version 6. Available at http://www.oracle.com/technetwork/java/javase/downloads/index.html

[26] Google Web Toolkit (GWT) is an open source set of tools that allows web developers to create and maintain complex JavaScript front-end applications in Java. Available at http://code.google.com/webtoolkit/

[27] Apache Tomcat is an open source servlet container developed by the Apache Software Foundation (ASF). Available at http://tomcat.apache.org/

The execution time for semantic similarity analysis was calculated while the system runs the combined measure of the layered approach. Without the help of grid computing, the execution time was 420 sec. Increasing the number of processors in the grid computing network reduced the execution time. Figure IV.6 shows that the execution time decreases as the number of processes increases.

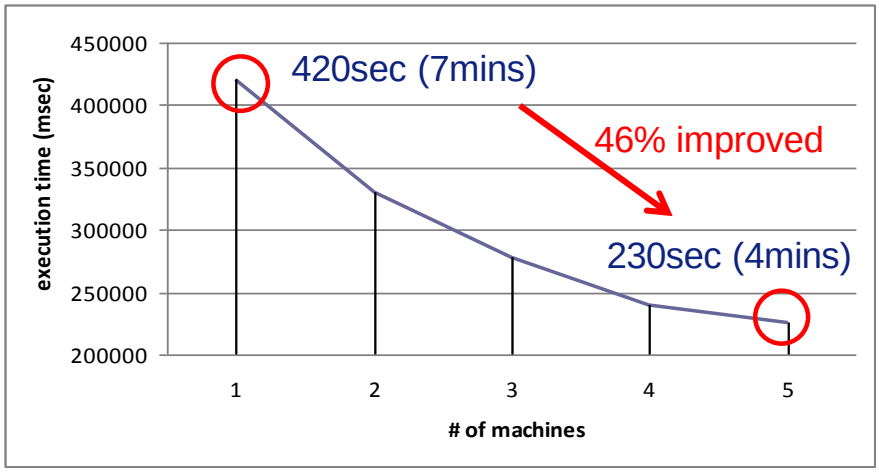


Figure IV.6. The number of machines vs. execution time.

In the ideal case of distributing the computation, the execution time should be decreased by inverse proportion to the number of machines as *execution_time*(n) $\propto$ *execution_time*(1) / n, where n is the number of machines. However, the actual execution takes longer than the ideal due to the trade-off between networking overhead and performance. Figure IV.7 shows the comparisons of the execution time among actual, ideal, and other cases.

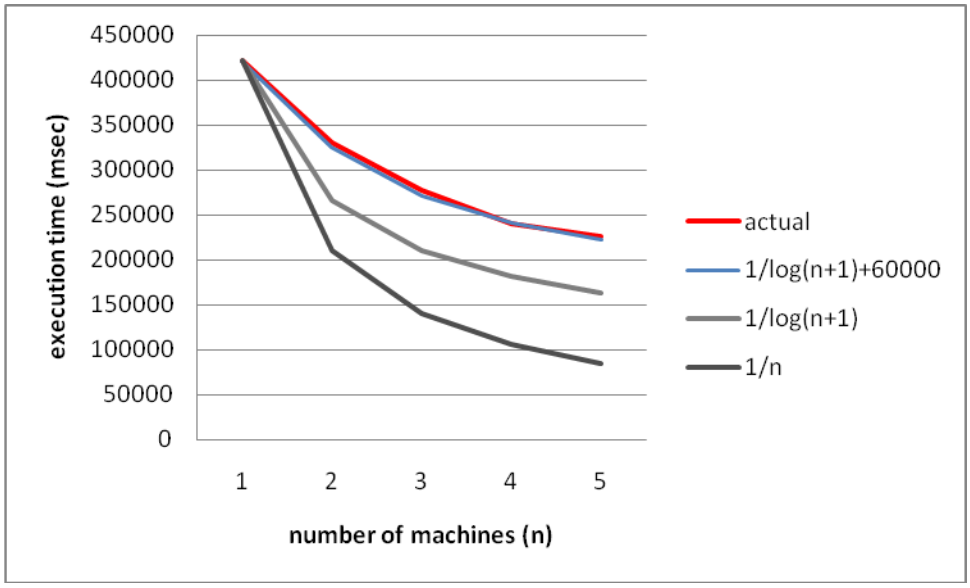


Figure IV.7. Comparison of execution time by actual vs. ideals.

If five machines are used, the actual execution takes twice as much time as the ideal (by the graph of the function "1/n" in Figure IV.7). As shown in Figure IV.7, the curve of actual execution time is more likely to be similar to the curve of a reciprocal logarithmic function as follows:

$$\text{execution\_time}(n) = \text{execution\_time}(1) / \log_2(n+1) + c, \qquad \text{(IV.7)}$$

where c is a constant value.

When c = 60000, especially, both curves are almost identical. This result implies that the proposed grid computing requires the constant time (60 secs) to initiate the MPJ module and the networking overhead reduces the performance to a logarithmic proportion. However, the results also show that the proposed network could successfully improve in performance by reducing the semantic similarity computation time for the two large-scale XML schemas.

IV.5. Chapter Summary

In this chapter, we proposed an innovative semantic similarity analysis approach for *c-matching* of XML schemas which exploits semantic information embedded in XML schemas beyond existing methods. This was done by dividing data elements into layers and measuring semantic similarity using layer specific metrics. We also implemented a prototype system to evaluate the proposed approach. This system recommends for each element in a source XML schema a set of matching candidates in a target schema based on the semantic similarity measures between the elements in these two schemas. The proposed approach and prototype system have the potential to provide valuable help for the human integrators solving the problem of XML schema matching, merging and reuse.

A series of experiments have been conducted with encouraging results, the system found a match to the human experts' matching results in 31 of 49 cases in a real world application. The experiments also revealed that the problem is much more complicated than we initially thought. One observation is that the scores of or similarity measures vary greatly among the manual matchings (ranging from 0 to 1). This calls for further examination of the similarity measures and of the way they are combined, and for exploring more elaborated matching procedures.

We also proposed a service-oriented architecture for XML schema matching based on a grid computing technology in order to reduce the computational cost for the layered approach with large schemas. We implemented a prototype to evaluate the proposed approach which can provide efficient and highly extensible XML schema

matching web services. The existing schema matching tools can extend their software functionalities to support automated schema matching simply by invoking our web services. The experiment results showed encouraging improvements in performance by reducing significantly the computation time of the semantic similarity between two large-scale XML schemas.

CHAPTER V

AN OPTIMIZATION APPROACH FOR XML DOCUMENT SCHEMA MATCHING

In the previous chapter, we proposed the layered approach for analyzing *c-matching* between two component schemas. Another important matching task for B2B systems integration is the *document schema matching* (*d-matching*), which identifies relationships between nodes of two document schemas. The *d-matching* problems can be classified into one of two types: *f-matching* (all nodes are fully matched to create an integrated schema) and *a-matching* (only atomic nodes are matched to determine how to transform one instance into another).

In this chapter, we propose an *a-matching* algorithm for analyzing schema matching between two XML document schemas using semantic similarity measures. Our approach focuses on a combinatorial optimization problem of finding the best matching between two sets of atomic nodes. It finds the optimal set of matching pairs between two sets of atomic nodes in a principled manner by mathematical programming. The proposed approach utilizes both structural and linguistic information in XML schemas. For structural information, we focus on the *path-contexts* which identify one type of the structural context of atomic nodes. For linguistic information, we utilize the WordNet to obtain the semantic information for the words in the labels of nodes.

V.1. Matching Algorithm Overview

The matching algorithm takes two schemas as input and identifies the set of matching pairs of all atomic nodes with the highest semantic similarity among all possible sets of pair-wise matchings. Figure V.1 illustrates the matching process of our approach for two input schemas, S1 and S2.

The algorithm breaks the complex combinatorial optimization problem of *a-matching* into four matching stages: tree-to-tree (between the sets of atomic nodes of the two schema trees), path-to-path (between the sets of nodes on the paths of two atomic nodes), node-to-node (between sets of words in the labels of two nodes), and word-to-word (between multiple senses of two words) matchings.

The main advantage of this decomposition is to make the matching problem simpler. The optimal matching problem for each stage can be more easily addressed. Another advantage is to provide a framework for users to iteratively analyze the results and input their feedbacks. It is widely accepted that the matching process cannot be fully automated and user intervention is always required [Rahm 2001 and Shvaiko 2005]. In most existing matching algorithms, the user input is requested at pre-match (to provide an initial matching) or at post-match simply to validate the matching output. In our proposed approach, users can iteratively review the matching results of each stage and input their feedbacks in terms of matching corrections between two word senses, two words, two nodes, or two paths. We assume that users know the contexts of schemas and understand the matching result to decide whether a matching is correct to produce a desired transformation pair.

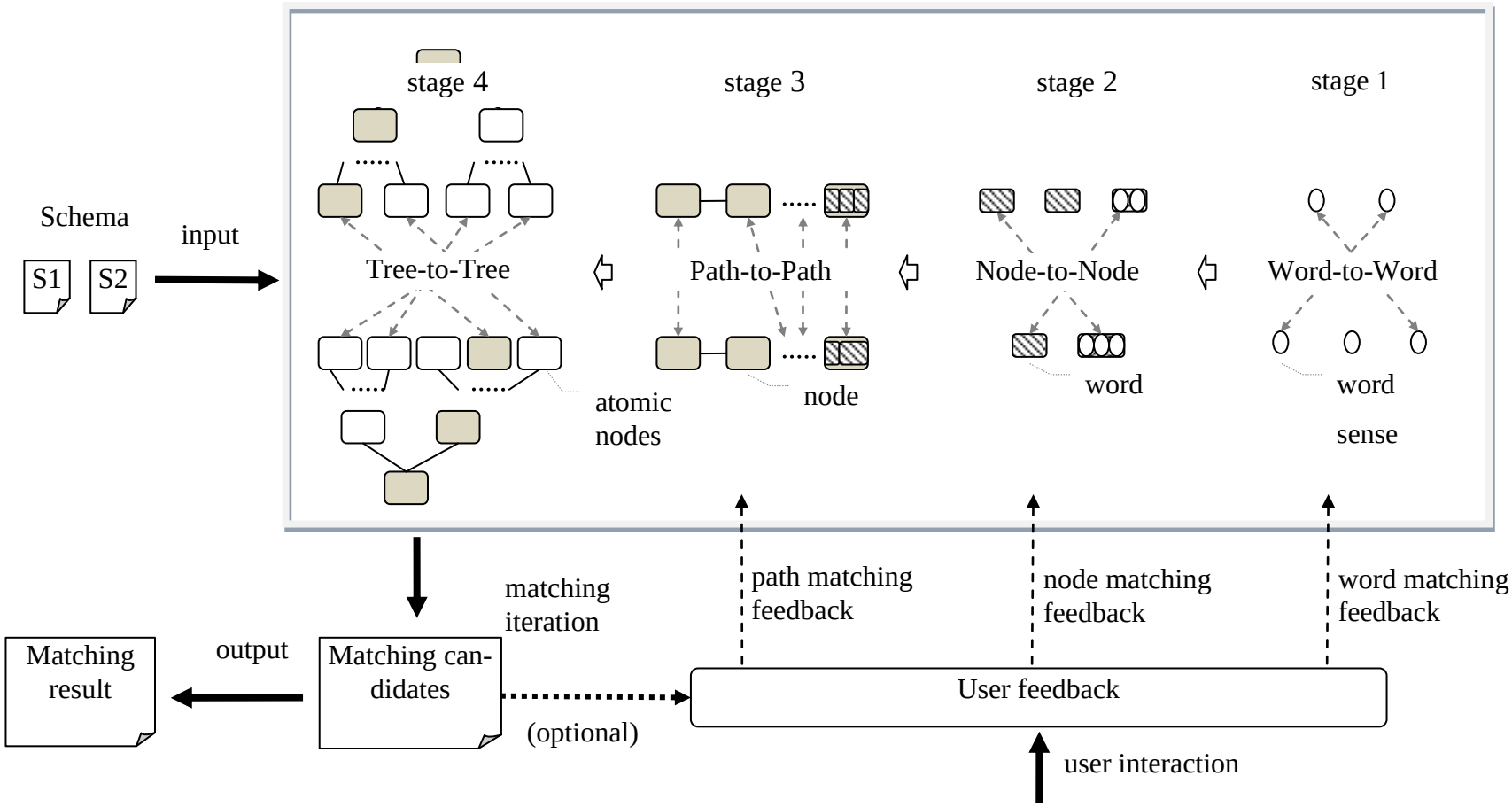


Figure V.1. Matching algorithm overview.

As can be seen in Figure V.2, each stage works on a bipartite graph, consisting of two sets of vertices and a weight matrix between them, with the objective of finding the *1-to-1* matching between vertices in one set to the other with the highest combined weight. Therefore, we formulate these sub-problems as maximum-weighted bipartite graph matching problems [Dulmage 1958].

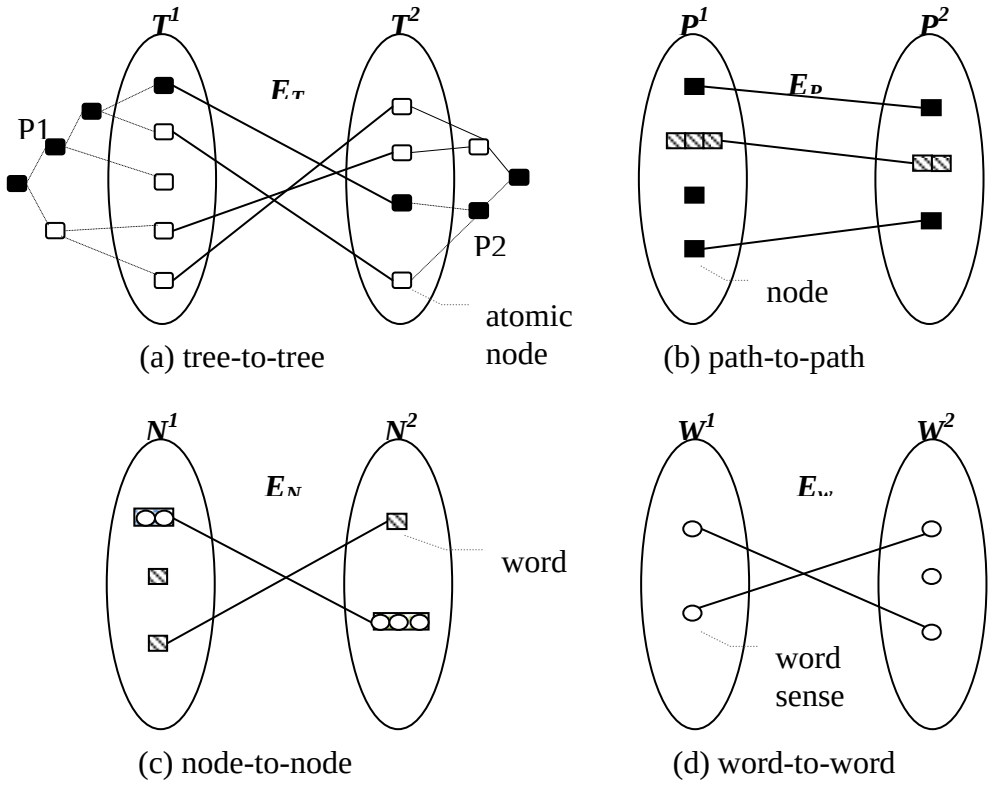


Figure V.2. Weighted bipartite graph modeling for different levels of matching.

Except for the word-to-word matching at the bottom stage, the weight matrix between pairs of edges for each stage is a similarity matrix calculated by the previous stage. For example, the similarity matrix for tree-to-tree matching stage is provided by path-to-path matching stage. The word-to-word matching stage uses WordNet to

compute the semantic similarity between two words by identifying the optimal matching pairs in respective senses.

Except the path-to-path matching stage, optimal matching at each stage can be obtained according to the general *Maximum-weighted Bipartite Matching* algorithm (*MBM*) [Douglas 1999]. The path-to-path matching requires an additional *ordering criterion* [Carmel 2002] that path P^1 includes most of the nodes of path P^2 in the correct order as shown in Figure V.2 (b). This is called *Ordered Maximum-weighted Bipartite Matching* (*OMBM*) problem. Algorithms for solving the *MBM* and the *OMBM* problems are described in the following sections.

V.2. Maximum-Weighted Bipartite Matching Algorithm

Tree-to-tree, node-to-node, and word-to-word matching stages can be formulated as the general *weighted bipartite graph matching* problems. Let *G* be a weighted bipartite graph with two sets of vertices, $U = \{u_1, u_2, ..., u_m\}$ and $V = \{v_1, v_2, ..., v_n\}$, and the set of edges, *E*. Edge e_{ij} in the graph connects the vertices u_i and v_j whose weight w_{ij} is given in the weight matrix *W*. Vertices of the same set are not connected.

A matching *M* of graph *G* is a subset of *E* such that no two edges in *M* share a common vertex. In other words, the matching *M* consists of a set of matching pairs that satisfies *1-to-1* pair-wise matching. The *maximum-weighted bipartite matching* is a matching whose sum of the weights of the edges is the highest among all possible sets of pair-wise matchings. The optimal matching *M* can be found by integer programming as defined below:

Definition V.1: (Optimal maximum-weighted bipartite matching) Given a matching M between two vector sets U and V, let x be an incident set where $x_{ij} = 1$ if $(i, j) \in M$ and 0 otherwise. w_{ij} is a weight for matching between i and j where $i \in U$ and $j \in V$. One can formulate the optimal maximum-weighted bipartite matching problem as follows:

Maximize: $$\sum_{i \in U, j \in V} w_{ij} x_{ij}, \qquad \text{(V.1)}$$

subject to:

$$\sum_{i \in U} x_{ij} \le 1 \qquad \forall j \in V,$$

$$\sum_{j \in V} x_{ij} \le 1 \qquad \forall i \in U, \text{ and}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in U, j \in V.$$

Because integer programming is typically NP-hard (i.e., harder than a nondeterministic polynomial-time problem and in the worst case with running time exponential to the problem size) [Papadimitriou 1981], we approximate it by a simple greedy algorithm as follows:

Algorithm *MBM-greedy*(*U*,*V*,*W*)

```
1.  m:= |U|, n:= |V|, M:= ∅;
2.  sort W;
3.  while (|U|>0 and |V|>0)
4.    Choose vertices u and v connected with an edge e that
    has the highest weight w in the weight matrix W;
5.    if edges in M share neither u nor v
6.      then M:= M∪{e}, U:= U-{u}, V:= V-{v}, wsum:= wsum+w;
7.    end if;
8.    W[u,v]:= 0;
9.  end while;
10. Sim:= 2*wsum/(m+n);
```

```
11. return {M, Sim};
```

Figure V.3. Greedy algorithm for maximum-weighted bipartite matching.

The greedy algorithm simply sorts the weight matrix *W* in descending order, and at each iteration it chooses an edge with the highest weight. The initial weight matrix *W* is calculated by the previous matching stage. The chosen edge will be the matching candidate if it shares no vertex with edges already in *M*. This process is repeated until there is no vertex to be matched in either *U* or *V*. The algorithm returns a (sub) optimal matching *M* and the average weight of all edges in *M* as the measure of similarity between *U* and *V*. In this greedy algorithm, the most expensive step is the sorting of the weight matrix *W* of size $|U| \times |V|$. We use a quicksort algorithm [Hoare 1962] that takes $O(k \log(k))$ to sort *k* items. Therefore, the complexity of this greedy algorithm is $O(|U||V| \log(|U||V|))$.

V.3. Ordered Maximum-Weighted Bipartite Matching Algorithm

Some have suggested using the longest common sequence (LCS) to address the ordering criterion of *OMBM* problems such as our path-to-path matching [Boukottaya 2005; Douglas 1999; and Mong 2002]. However, these suggestions only employ the exact string matching between nodes on two path contexts. None of them utilizes the semantic similarities of the nodes on the two path contexts. To consider the semantic similarities of the nodes, we have developed an *ordered maximum-weighted bipartite matching* algorithm based on dynamic programming as follows.

Definition V.2: (Optimal ordered maximum-weighted bipartite matching) Let *G* be a weighted bipartite graph with two *ordered* sets of vertices $U = \{u_1, u_2, ..., u_m\}$ and $V = \{v_1, v_2, ..., v_n\}$, and the set of edge *E* and the weight matrix *W*. The core algorithm, *OMBM* (*U, V*), finds the optimal matching *M* between *U* and *V* by recursively partitioning the problem into smaller sub-problems until the solution becomes trivial. Note again that *W* is calculated by the previous matching stage (i.e., node-to-node matching stage). For a sequence $S=s_1s_2...s_d$, a subsequence shortened from the end is denoted $S_k=s_1s_2...s_k$, where $k \leq d$. We call S_k the *prefix* of *S*. The *prefixes* of *U* are U_1, U_2,..., U_m, and the *prefixes* of *V* are $V_1, V_2, ...V_n$. Let *OMBM* (U_i, V_j) be the function that finds the optimal matching of *prefixes* U_i and V_j. This can be solved by first reducing the original problem to three simpler sub-problems with shortened *prefixes* and by returning the solution for one of the sub-problems with maximum sum of weights:

1) u_i and v_j match each other. Then, the optimal matching for U_i and V_j can be formed by attaching edge e_{ij} to the optimal matching of two shorten sequences U_{i-1} and V_{j-1}, denoted (*OMBM* (U_{i-1}, V_{j-1}), e_{ij}).

2) u_i and v_j do not match each other. Then, either of them can be removed to shorten one of two the sequences and *OMBM* (U_i, V_j) is reduced to either *OMBM* (U_{i-1}, V_j) or *OMBM* (U_i, V_{j-1}).

Thus *OMBM* (U_i, V_j) can be computed by the following recursive function:

$$OMBM(U_i, V_j) = \begin{cases} \varnothing & \text{if } i = 0 \text{ or } j = 0 \\ max \begin{pmatrix} OMBM(U_{i-1}, V_j), \\ OMBM(U_i, V_{j-1}), \\ \left(OMBM(U_{i-1}, V_{j-1}), e_{ij}\right) \end{pmatrix} & \text{otherwise} \end{cases} . \quad \text{(V.2)}$$

where the function *max* returns the optimal matching among the three matchings from the sub-problems based on the similarity scores returned by *OMBM*; it returns empty if either U_i or V_j is reduced to *null* ($i = 0$ or $j = 0$).

The similarity score of *OMBM* (U_i, V_j), denoted Sim_{OBMB} (U_i, V_j), is the average weight (similarity score) of all edges (matching pairs) in the matching. It can be calculated by Eq. III.1 as follows:

$$Sim_{OMBM}\left(U_i, V_j\right) = \frac{2}{i+j} \sum_{e_{ij} \in OMBM\left(U_i, V_j\right)} w_{ij} . \quad \text{(V.3)}$$

The optimal matching M of two sets of ordered vertices U and V, $|U| = m$, $|V| = n$, is then computed as:

$$M = OMBM\left(U, V\right) = OMBM\left(U_m, V_n\right). \quad \text{(V.4)}$$

By Eq. III.1, the similarity score of M, denoted $Sim_{OBMB}(U, V)$ can be calculated as follows:

$$Sim_{OMBM}\left(U, V\right) = \frac{2}{|U| + |V|} \sum_{e_{ij} \in M} w_{ij} . \quad \text{(V.5)}$$

The example below shows how the optimal matching and similarity score between two simple path contexts is calculated by Eqs. V.4 and V.5.

Example V.1: Consider two path contexts "*PO/BillTo/Zip*" and "*PurchaseOrder/Customer/Address/Postal*". Let *P1* and *P2* be the ordered sets of nodes on these two paths:

$$P1 = \{n_{PO}, n_{BillTo}, n_{Zip}\} \text{ and } P2 = \{n_{PurchaseOrder}, n_{Customer}, n_{Address}, n_{Postal}\}.$$

Suppose that the similarity scores among all node pairs between the two sets *P1* and *P2* are as follows:

$$sim^N\left(n_{PO}, n_{PurchaseOrder}\right) = 1.0,\ sim^N\left(n_{BillTo}, n_{Customer}\right) = 0.6,$$

$$sim^N\left(n_{BillTo}, n_{Address}\right) = 0.4, \text{ and } sim^N\left(n_{Zip}, n_{Postal}\right) = 1.0.$$

The similarities between all other pairs are 0. By Eqs. V.3 and V.4, *OMBM* (*P1*, *P2*) of Eq. V.2 returns the optimal matching for the nodes between the two paths as follows:

$$M = \{(n_{PO}, n_{PurchaseOrder}), (n_{BillTo}, n_{Customer}), (n_{Zip}, n_{Postal})\}.$$

By Eq. V.5, the similarity score is

$$Sim_{OMBM}\left(P1, P2\right) = \left(1.0 + 0.6 + 1.0\right) \times \left(2 / \left(3 + 4\right)\right) = 0.74.$$

To efficiently execute the algorithm, we use a bottom-up approach [Bellman 2003]. The algorithm is as follows:

Algorithm *OMBM-A*(*U*,*V*,*W*)

```
1. for i from 1 to |U|
2.   for j from 1 to |V|
3.     A[i,j]:= maximum of A[i-1,j], A[i,j-1],
                                  and A[i-1,j-1]+W[i,j];
4. Sim:= 2*A[|U|,|V|]/(|U|+|V|);
5. return {A, Sim};
```

Figure V.4. Bottom-up dynamic programming algorithm for ordered maximum-weighted bipartite matching.

This algorithm starts from the simplest matching between U_1 and V_1 and continues to more complex matching problems. The calculated similarity scores for the optimal matchings (average weights by Eq. V.5) are stored in a similarity matrix $A[i,j]$ of the two dimensional array in Figure V.4. The bottom-up approach helps to avoid repeated calculations of smaller problems, whose similarity scores are stored in the similarity matrix $A[]$. The complexity is only $O(|U||V|)$, i.e., linear to the size of the similarity matrix $A[]$.

Based on the similarity matrix $A[]$ calculated by *OMBM-A*, Figure V.5 shows an algorithm to generate the pair-wise matching between two nodes in U and V. This pair-wise matching forms the optimal path-to-path matching.

Algorithm *OMBM*(*U*,*V*,*W*)

```
1.  M:= ∅;
2.  {A,Sim}:= OMBM-A(U,V,W);
3.  i:= |U|, j:= |V|;
4.  while i > 0 and j > 0
5.    if A[i,j] equal to A[i-1,j] then i--;
6.      else if A[i,j] equal to A[i,j-1] then j--;
7.      else M:= M∪{e_{i,j}}, i--, j--;
8.    end if;
```

```
9.  end while;
10. return {M, Sim};
```

Figure V.5. Dynamic programming algorithm for ordered maximum-weighted bipartite matching.

How this bottom-up dynamic programming algorithm works is illustrated below using the same example defined in Example V.1.

Example V.2: Consider *P1* and *P2* for the two path contexts defined in Example V.1. The weight matrix *W* is initialized by the similarity scores between all pairs of nodes as shown below:

Table V.1. An Example of Weight Matrix

W		3	2	1
		n_{PO}	n_{BillTo}	n_{Zip}
4	$n_{PurchaseOrder}$	0.0	0.0	1.0
3	$n_{Customer}$	0.6	0.0	0.0
2	$n_{Address}$	0.4	0.0	0.0
1	n_{Postal}	0.0	1.0	0.0

The array *A*[] for calculating the matching similarity scores can be represented as follows.

Table V.2. An Example of Matching Similarity Score Table

A	3	2	1	0
4	1.0+1.6=2.6	1.6	1.0	0.0
3	1.6	0.6+1.0=1.6	1.0	0.0
2	1.6	0.4+1.0=1.4	1.0	0.0
1	1.0	1.0	1.0	0.0
0	0.0	0.0	0.0	0.0

Note that values in the array are not normalized. According to the algorithm *OMBM-A* in Figure V.4, the value of $A[i,j]$ is obtained from the maximum of the three values: $A[i\text{-}1,j]$, $A[i,j\text{-}1]$, and $A[i\text{-}1,j\text{-}1] + W[i,j]$, where $1 \le i \le 3$ and $1 \le j \le 4$. The initial values of $A[]$ set to zero.

The calculation starts from the simplest matching array $A[1,1]$. The algorithm compares three values: $A[1,0] = 0$, $A[0,1] = 0$, and $A[0,0] + W[1,1] = 1.0$ and the maximum score 1.0 is chosen. To find the optimal matching by the *ordered maximum weighted bipartite matching algorithm*, look at the first entry $A[3,4]$. It is calculated by the maximum value among three matching scores: $A[2,4] = 1.6$, $A[3,3] = 1.6$, and $A[2,3] + W[3,4] = 1.6 + 1.0 = 2.6$. The maximum value is 2.6, telling us the normalized similarity by the average length of two paths is $2.6 \times (2/(3+4)) = 0.74$, which is actually the same as what was calculated in Example V.1.

According to algorithm *OMBM* in Figure V.5, the optimal matching result can be obtained by following the traces to reach the first entity $A[3,4]$. As highlighted in Table V.2, the entities used to calculate $A[3,4]$ are $A[3,4]$, $A[2,3]$, $A[1,2]$, and $A[1,1]$. Then, the entities added their similarity scores are selected as matching: $A[3,4]$, $A[2,3]$, and $A[1,1]$, which lead to the optimal matching

$$M = \{(n_{PO}, n_{PurchaseOrder}), (n_{BillTo}, n_{Customer}), (n_{Zip}, n_{Postal})\}.$$

Algorithm *OMBM-A* and *OMBM* are further enhanced by considering the differences in importance for the individual nodes measured by their information contents. We collect each node's frequency-of-occurrence in the schema trees and com-

pute the information contents by Eq. II.6. Figure V.6 shows the modified algorithm of *OMBM-A* shown in Figure V.4 in which weights w_{ij} are modified by the IC values of u_i and v_j.

Algorithm *OMBM-A-IC*(*U*,*V*,*W*)

```
1.  for i from 1 to |U|
2.    ic_sum:= ic_sum+ic(u_i);
3.  end for;
4.  for j from 1 to |V|
5.    ic_sum:= ic_sum+ic(v_j);
6.  end for;
7.  for i from 1 to |U|
8.    for j from 1 to |V|
9.      ic_w:= W[u_i,v_j]*(ic(u_i)+ic(v_j));
10.     A[i,j]:= maximum of A[i-1,j], A[i,j-1],
                                  and A[i-1,j-1]+ic_w;
11.   end for;
12. end for;
13. Sim:= A[|U|,|V|]/ic_sum;
14. return {A, Sim};
```

Figure V.6. Algorithm enhanced by information contents.

Algorithm *OMBM-A-IC* modifies the original weights w_{ij} by the IC values of u_i and v_j (line 9), and calculate new similarity weighted by the IC values. The modified similarity is then normalized by the sum of IC values of all u_i and v_j (line 13). This tends to give more weights to higher-level nodes because lower-level nodes are usually generic entities that appear widely as descendants of higher-level nodes and thus have lower IC values. In addition, it also considers the differences in importance of nodes at the same level. The *OMBM-A-IC* can be alternatively applied to consider the differences in importance of nodes for *OMBM*. The complexity of this algorithm is still $O(|U||V|)$.

V.4. Overall Schema Matching Algorithm

Figure V.7 shows the algorithm for overall schema matching and explains how each stage obtains the weight matrix by calling the optimization algorithm of the previous stage.

Algorithm *A-matching*(*T1, T2*)

```
1.  return T2T-matching(T1,T2);
```

Function T2T-matching(*T1, T2*)

```
2.  for i from 1 to |T1|
3.    for j from 1 to |T2|
4.      t2t-smatix[i,j]:= P2P-matching (path of T1's ith atom,
                                        path of T2's jth atom);
5.    end for;
6.  end for;
7.  return MBM-greedy (T1's atoms, T2's atoms, t2t-smatix);
```

Function P2P-matching(*P1, P2*)

```
8.  for i from 1 to |P1|
9.    for j from 1 to |P2|
10.     p2p-smatix[i,j]:= N2N-matching (P1's ith node,
                                        P2's jth node);
11.   end for;
12. end for;
13. return OMWM-IC (P1's nodes, P2's nodes, p2p-smatix);
```

Function N2N-matching(*N1, N2*)

```
14. for i from 1 to |N1|
15.   for j from 1 to |N2|
16.     n2n-smatix[i,j]:= W2W-matching (N1's ith word,
                                        N2's jth word);
17.   end for;
18. end for;
19. return MWM-greedy (N1's words, N2's words, n2n-smatix);
```

Function W2W-matching(*W1, W2*)

```
20. if wordnet definitions for W1 and W2 exists then
```

```
21.    for i from 1 to |W1|
22.      for j from 1 to |W2|
23.        w2w-smatix[i,j]:= word-sense-sim (W1's i^th sense,
                                              W2's j^th sense);
24.      end for;
25.    end for;
26.    return MWM-greedy (W1's senses, W2's senses,
                                            w2w-smatix);
27.  else
28.    return word-desc-sim (W1, W2);
29.  end if;
```

Figure V.7. Overall schema matching algorithm.

The algorithm considers matching between two schema trees as matching between two sets of atomic nodes (i.e., *a-matching*) with their respective *path-contexts*. Each path consists of a sequence of nodes along the path from the root to the leaf of the schema tree. Each node represents either an element or an attribute named by a label of English word or concatenation of words or their abbreviations. To compute semantics similarities between two words, we analyze optimal pair-wise matching between multiple meaning (senses) of the two words.

The word-to-word matching algorithm uses two semantic similarity measure functions: *word-sense-sim* based on WordNet taxonomy and *word-desc-sim* based on textual description. In WordNet, nouns are organized into taxonomies in which each node has a set of synonyms (a synset), each of which representing a single sense [Miller 1995]. If a word has multiple senses (meaning), it will appear in multiple synsets at various locations in taxonomy. To compute the semantic similarity between two words (two sets of senses), we use the *MBM-greedy* algorithm with the input of two

set of senses for words *W1* and *W2*, respectively, and the similarities between the two senses are calculated by Eq. II.6.

If a word does not exist in WordNet, we extract the textual description of a given word from the internet and then use string-similarity measures, such as the cosine similarity [Sneath 1957], to calculate the similarity between the textual descriptions of the two words.

V.5. Experiments and Results

A prototype system is implemented. The system implements *MBM* and *OMBM* algorithms to compute the optimal *a-matching* between two XML schemas.

V.5.1. Experimental data

To test and evaluate the proposed approach, we used five real world XML schemas for purchase orders (i.e., CIDX, Apertum, Excel, Norris, and Paragon) from [Biztalk 2010 and Aumüller 2005]. Table V.3 summarizes the characteristics of those XML schemas.

Table V.3. Characteristics of PO XML Schemas

Schemas	CIDX	Apertum	Excel	Norris	Paragon
max depth	4	5	4	4	6
# of nodes	40	145	55	65	80
# of atomic nodes	33	116	42	54	68

V.5.2. Results analysis

In the experiment, as it was suggested in [Aumüller 2005], we compute the tree-to-tree similarity of *a-matching* for all ten pairs of five XML schemas. Then for each schema, we accept a matching to any of the other four if the similarity score is above a fixed threshold 0.6. To evaluate the quality of our match result, we used several performance metrics including "*Precision*", "*Recall*", "*F-measure*", and "*Overall*" [Aumüller 2005 and Makhoul 1999], against the results from manual matching [Aumüller 2005]. These measures are then compared with the performances of other approaches with the same setting [Madhavan 2001; Thang 2008; and Aumüller 2005]. Note that the "*Overall*" metric, proposed by [Aumüller 2005] to estimate the post-match efforts, varies in [-1,1] and the other metrics vary in [0,1].

The performances of our results in "*Precision*", "*Recall*", "*F-measure*", and "*Overall*" are 0.85, 0.85, 0.85, and 0.69, respectively. To increase "*Precision*", we used a relative threshold which is chosen as the similarity of the matching with the largest gap to the next best matching, among matching candidates with similarities ranging from 0.5 to 0.6. Figure V.8 shows the performance analysis of the matching result that our solution produced.

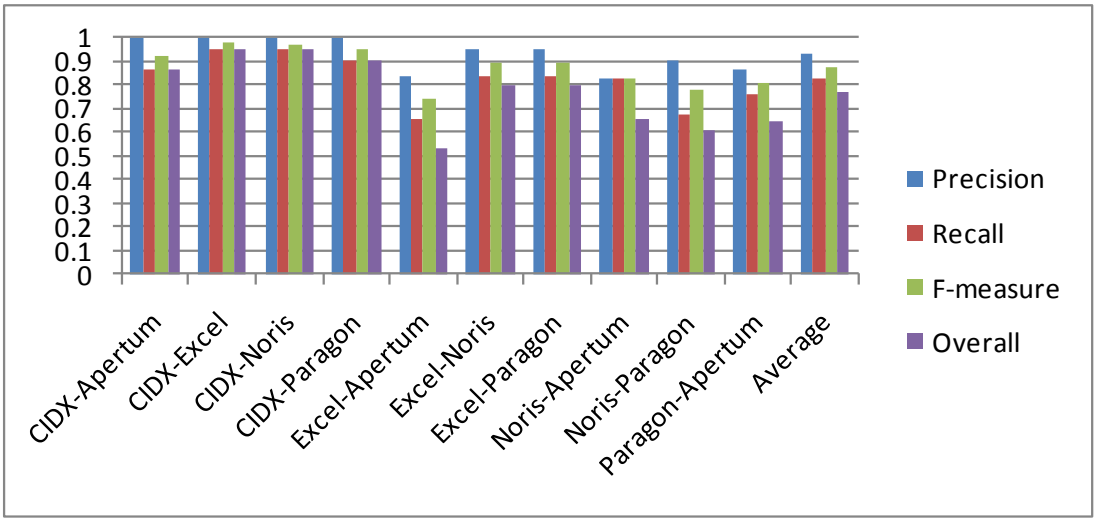
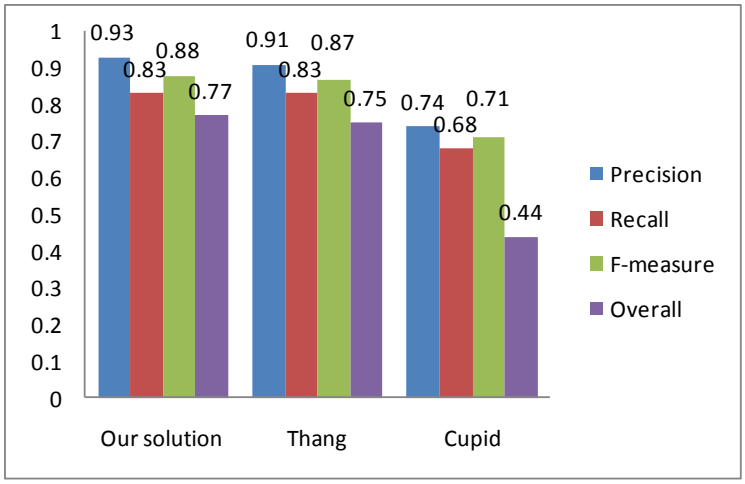


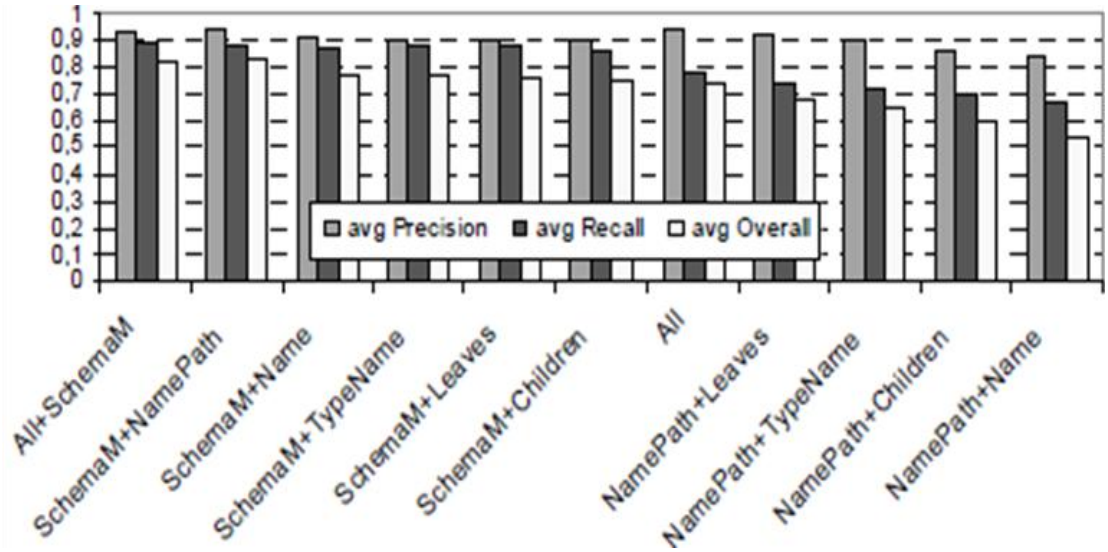
Figure V.8. Performance analysis #1.

The experiment results show that our matching performances of average *Precision*, *Recall*, *F-measure*, and *Overall* are 0.93, 0.83, 0.88, and 0.77, respectively. Compared to the previous results that used a fixed threshold, the *Recall* slightly decreased, while the *Precision* significantly increased. The relative threshold also helps to increase *F-measure* and *Overall*. For comparison purposes, the average scores of performance metrics derived by other methods are given in Figure V.9.

The first comparison, as illustrated in Figure V.9 (a), is with Thang (2008) who proposed an XML schema matching solution that combines linguistic, data type, and path-context similarity measures. He also implemented the Cupid [Madhavan 2001] algorithm for comparison purpose. We compared our result to both algorithms. In general, all performance metrics of our approach are slightly better than Thang's and significantly better than Cupid's.



(a) Thang and Cupid



(b) COMA: matcher combinations

Figure V.9. Performance analysis #2.

The second comparison is with COMA (COmbination MAtch) [Aumüller 2005], which used various ways to combine different matchers. Since COMA only provides performance graphs without the specific scores as shown in Figure V.9 (b), it is difficult to compare the performances with our result precisely. However, comparison between Figure V.8 and Figure V.9 (b) shows that our result is, in general, at least equal

to or slightly better than COMA's results even if some of their matchers used the manual matching called *SchemaM* [Aumüller 2005].

V.6. Chapter Summary

In this chapter, we have described a solution to identify semantic-based optimal XML document schema matching using mathematical programming. This solution identifies the optimal matching between two XML schemas on the assumption that the *a-matching* problem can be globally optimized by reducing it to simpler problems, such as path-to-path, node-to-node, and word-to-word matching. We have implemented a prototype system for our solution and conducted experiments with actual industry XML schemas. We compared our result to some other XML schema matching approaches. The results were positive. The average matching performances of "*Precision*", "*Recall*", "*F-measure*", and "*Overall*" were 0.93, 0.83, 0.88, and 0.77, respectively, which are better than or at least equal to other approaches' performances.

Although our approach primarily targets the XML schema matching problem, the solution can be also applied to other matching problems - such as XML instance matching if the instances can be represented as labeled trees (see Chapter VI). Our solution is limited by the assumptions that only *1-to-1* matching is considered and that schema designers correctly use the English terminologies when labeling the elements/attributes in the schemas. These limitations call for further research. Other directions of research include methods to improve the performance by utilizing domain specific terminology and taxonomy, ontology with formally defined concept semantics, and user feedback.

CHAPTER VI

XML MATCHING APPLICATION – SUPPLIER DISCOVERY

This chapter introduces an XML matching application to address the issue of discovery of suppliers. The suppliers' manufacturing capability profiles and customers' requirements are typically represented based on semi-structured descriptions in the form of textual information of their manufacturing products, processes, tools, materials, and resources. In this chapter, we employ a new form architecture based on XML schema to help capturing requirements and capabilities in a better structured way. Matchings between the collected requirements and capabilities are analyzed by our optimal XML schema matching algorithm.

VI.1. Supplier Discovery Overview

In today's dynamic manufacturing industry, the discovery of manufacturing suppliers – henceforth, *supplier discovery* – is essential for building a flexible network of suppliers in the supply chain [Christopher 2004]. To facilitate supplier discovery, several electronic marketplaces (e-marketplaces), such as Thomasnet[28],

[28] http://www.thomasnet.com/

mfg.com[29], and GlobalSpec[30], have been established. These e-marketplaces are commerce sites on public internet that allow large communities of customers and suppliers to connect and trade with each other [Ariba 2000].

In general, the supplier discovery function in e-marketplaces involves two steps. The first step is to collect supplier capabilities and customer requirements – henceforth *collect function*. In traditional e-marketplaces, form-based user interfaces are typically used for the *collect function* [Dumas 2004 and Noia 2005]. However, those forms are mostly fixed and pre-defined, so that they are not flexible enough to capture a variety of requirements and capabilities. Therefore, some information is entered in an unstructured way such as free texts. Different suppliers (or customers) often use different terminologies and structures with their own semantics to represent their own capabilities (or requirements). The requirements and capabilities collected by the traditional forms are mostly semi-structured by some tables and database schemas, or even unstructured as textual description and with un-unified semantics.

The second step is to find suppliers whose capabilities are of the greatest relevance to requirements specified by the customer – henceforth, *search function*. The traditional e-marketplaces typically provide keyword search, directory search, and database search capabilities. Several approaches have been proposed to enhance these search functions (e.g., semantic-based search) [Ameri 2006; Guarino 1995; and Jang

[29] http://www.mfg.com/

[30] http://www.globalspec.com/

2008]. They mostly apply to the structured data models such as XML[31], RDF[32], and OWL[33]. These advanced search approaches cannot be applied to unstructured or semi-structured information collected by traditional forms. Therefore, it is necessary to first enhance the "form" architecture that collects requirements and capabilities in a better structured way.

In this chapter, we propose new form architecture called eXtensible Dynamic Form (XDF) to help capture requirements and capabilities in a better structured way. Using XDF, we collect structural information of requirements and capabilities as XML instances. For identifying similarities between pairs of XML instances for requirements and capabilities, we extend the optimal XML schema matching algorithm proposed in Chapter V.

VI.2. eXtensible Dynamic Form (XDF) Architecture

For better collection and organization of supplier capabilities and customer requirements, XDF architecture allows users not only to extend the base form with new form components for their own contents but also to reuse the existing form components through intelligent search. It helps to represent the requirements and capabilities in a better structured way. Figure VI.1 shows the overview architecture of XDF.

---

[31] XML – eXtensible Markup Language

[32] RDF – Resource Description Framework

[33] OWL – Web Ontology Language

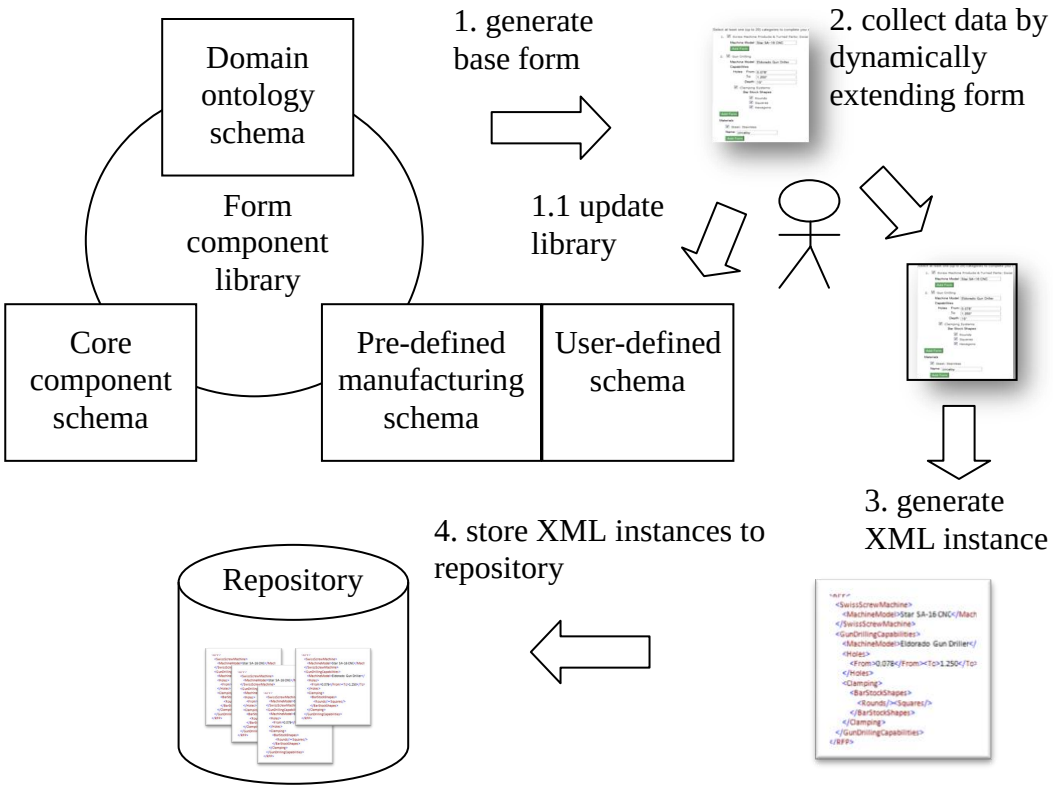


Figure VI.1. Architecture of XDF.

The basic architecture of XDF relies on a *form component library* to provide a collection of reusable form components, each of which can be attached as sub-structure to base form or other form components. The form components are defined by several XML schemas, including domain ontology, core component, and pre-defined manufacturing schemas.

The domain ontology is a formal representation of knowledge as a set of concepts within a specific domain, and the relationships among those concepts [Guarino 1995]. Several ontology approaches have been developed for manufacturing domain [Ameri 2006; Jang 2008; and Kulvatunyou 2005], especially to facilitate supplier discovery. They mostly provide a formal way to capture and represent semantic informa-

tion based on their ontology. However, most of the ontology developed so far is too small to apply to the real industry.

To create a basic set of form components, XDF utilizes a manufacturing domain ontology proposed by Ameri and Dutta (A&D) [Ameri 2006]. Based on the concepts and their relationships defined in the domain ontology, we created an XML schema model, called *domain ontology schema*, which represents the basic terminologies and structures of form components. The domain ontology is small and only defines the manufacturing domain specific concepts (e.g., equipments, materials, certifications and so on). To support more terminologies and concepts widely used in e-Business industry, XDF also utilizes OAG Common Core Components (CCC) XML schema [OAGIS 2002]. This schema defines the common or "general" components that basically can be used across several business domains (e.g., Address, Party, and ID).

We also investigated several supplier capability profiles obtained from Thomasnet.com. Many terminologies and concepts shared among these suppliers are not defined by both domain ontology and common core component schemas. This is because the domain ontology schema is too small to cover all the manufacturing concepts and common core component schema only defines the "general" components. For those terminologies and concepts that are mainly manufacturing domain specific but not covered by ontology, we created a new XML schema called *pre-defined manufacturing schema*.

As shown in Figure VI.1, the *collect function* of XDF is as follows. First, XDF generates a base form based on the *form component library*. Users (suppliers or cus-

tomers) may create new form components, which will be encoded and stored as *user-defined schema*. Secondly, the users fill out the base form with their data (i.e., requirements or capabilities) by dynamically extending the base form. Thirdly, XDF automatically generates XML instances based on the users' input data. Last, the generated XML instances are stored into a supplier capability repository to be used later for *search function*.

VI.3. Search Algorithms for Supplier Discovery

The search function is typically based on an algorithm that computes a similarity between requirement and capability information. The search result is then a ranked list of suppliers, whose capabilities are of the greatest similarity to requirements specified by a customer. We focus on an XML-based search method, but also employ keyword-based and ontology-based search methods for comparison purpose. These methods are described in the following subsections.

VI.3.1. XML-based search

To analyze XML instances encoded for requirement and capability, we have extended our optimal XML schema matching algorithm proposed in Chapter V. The XML instances can be represented as labeled trees similarly to XML schemas (see Figure I.2). However, instances often include descriptive text data for instance values which can be represented as labels in atomic nodes of the labeled trees. For example, "high definition large format Hewlett Packard 1050C plotter" is an instance value for an "Equipment" element. WordNet-based approach used in Chapter V is not effective

to analyze this descriptive text data. Among many text-based similarity metrics (see Chapter II.1.1), we use the *n*-gram (specifically, *tri*-gram) similarity metric [Kondrak 2005] which is simple to implement and produces slightly better results than Cosine coefficient [van Rijsbergen 1979] and Jaccard similarity [Sneath 1957]. We have also further investigated IC-based approach by Eq. II.6 for weighting the similarities between two phrases according to their importance. Instead of getting IC of words from WordNet and schema itself, we used the number of the web pages retrieved by Google's search as IC which could represent phrase importance. Instead of combining multiple ICs of words in the phrase, it actually provides more precise statistics for the phrase. In addition, Google allows specifying web sites for domain specific search scope.

The modified algorithm computes the semantic similarity between pairs of XML instances for requirement and capability. The search result is a ranked list of suppliers, whose capabilities are of the greatest semantic similarity to requirements specified by a customer.

VI.3.2. Keyword-based search

For keyword-based search, we use an exact string matching method that simply checks if the search keywords for customer requirements exist in the textual description of the supplier capabilities. The ranking of the returned set is also based on a number of keywords found in the capability descriptions. The higher the number of the matched keywords in descriptions of capabilities that a certain supplier can provide, the higher the rank of the supplier is.

VI.3.3. Ontology-based search

For ontology-based search, we use an ontology-based matching algorithm proposed by Ameri and Dutta (A&D) which connects customers and suppliers based on semantic similarities between the customer's requirements and suppliers' capabilities [Ameri 2006]. The algorithm relies on semantic definitions found in the manufacturing ontology rather than on syntactic descriptions of requirements and capabilities. It is required that both supplier capabilities and customer requirements are described formally using Manufacturing Service Description Language (MSDL), which is a formal language based on Description Logic developed particularly for matchmaking purposes [Baader 2003].

In addition to the logical reasoning, the quantification of similarity is supported in the A&D approach by several methods that calculate semantic distances among manufacturing concepts. The Information Content-based method and Feature-based method [Lin 1988; Resnik 1999; and Tversky 1977] are two of such methods. To optimize similarity calculation for the domain of manufacturing, different calculation methods were applied to different concepts. Specifically, the similarity scores for industry and product concepts in the A&D ontology are determined using an IC-based approach by Eq. II.6. On the other hand, the similarities for process or material concepts in the A&D ontology are computed using a feature-based method as follows:

$$Sim(A,B) = \frac{\eta_{A\cap B}}{\eta_{A\cap B} + u\eta_{A-B} + v\eta_{B-A}}, \qquad \text{(VI.1)}$$

where $\eta_{A\cap B}$ is the number of features common to both the query and the service profile, η_{A-B} is the number of features that are in the query but not in the service profile, and η_{B-A} is the number of features that are in the service profile but not in the query class.

Once semantic similarities among individual concepts are computed, the overall similarity of a service profile (SP) to a given query (Q) is calculated as the sum of similarities of the actor (the suppler) and similarities of the service.

VI.4. Experiments and Results

We implemented several prototype systems for XDF and four search methods. A variety of experiments have been conducted with the real industry data.

VI.4.1. Performance measures for evaluation

We measure the performances of *keyword-based* and *ontology*-based search methods, and they are compared with the performances of *XML-based* search methods. The performances of these approaches are measured by the *Recall* and *Precision* metrics (see Eqs. II.7 and II.8). Because our search methods return the *Ranked List of Suppliers* rather than one supplier as the search result, we employed new metric, called "Top-*k Recall*", that computes the fraction of the number of relevant suppliers retrieved by search method among *k* relevant answers. In this experiment, we assume that the number of the relevant answers is the same as the number of suppliers retrieved by search method. Thus, the *Precision* and *Recall* are the same. For Top 4 *Re-*

call, for example, top 4 relevant answers are chosen from the list of suppliers ranked by human experts, and top 4 suppliers are retrieved by the algorithm as the result of supplier discovery. If only 2 of 4 suppliers exist in the relevant answers, both *Recall* and *Precision* are the same 0.5.

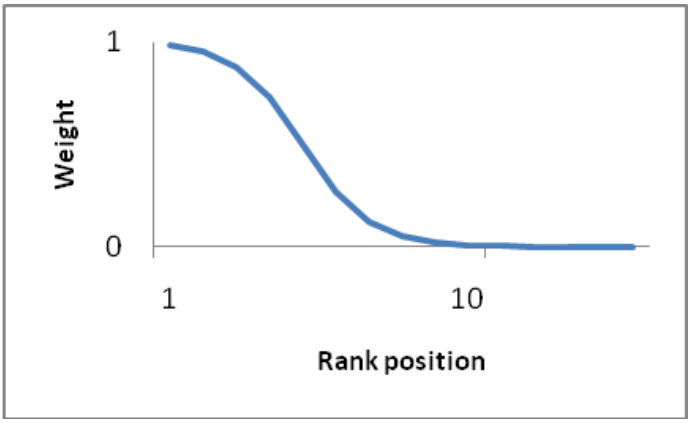


Figure VI.2. The adjusted weight function for the *DCG* metric.

As another performance metric for this experiment, we use the *DCG* performance metric by Eq. II.13. The original weight function for *DCG* has been refined in our experiment to reflect the characteristics of supplier discovery where top 10 suppliers in general are of substantial importance with their ranking order being of moderate significance to the result. The weight function, illustrated in Figure VI.2, can be formulated as

$$f(p) = \frac{1}{1+e^{p-5}}, \tag{VI.2}$$

where p is the rank position of a supplier in the search result.

Therefore, the *DCG* adjusted by Eq. VI.3 accumulated at a particular rank position *p*, called $aDCG_p$, can be calculated as follows:

$$aDCG_P = \sum_{i=1}^{p} rel_i \cdot \left(\frac{1}{1+e^{i-5}} \right), \tag{VI.3}$$

where rel_i is the graded relevance of the result at position *i*.

The $aDCG_P$ can be alternatively applied to calculate the *normalized DCG* (*nDCG*) by Eq. II.13.

VI.4.2. Experimental data

Initially, the supplier capabilities and customer requirements information were collected in the form of textual descriptions. This information is then encoded into XML instances by using XDF architecture. The proposed XML-based search method identifies matchings between these XML instances (i.e., requirements and capabilities). The search result is a list of suppliers ranked by similarities between supplier capabilities and customer requirements.

We chose 30 sample suppliers, which have capabilities in machining processes, randomly from manufacturers registered in the e-marketplace, Thomastnet.com. First, raw data that describe capabilities were collected from the e-marketplace and then refined by the data collected from the manufacturers' own web sites. Initially, the collected raw data of supplier capability profiles consists of a set of keywords. These words are directly used in the keyword-based search. The data is then manually en-

coded into ontological descriptions following the semantic search assumptions of the selected approach. We also encode the data as XML instances by XDF architecture.

Two sample customer requirements were also collected from the same e-marketplace. Raw data of customer requirements was technical specifications of certain products – called "center post" and "swing arm". The technical specifications were analyzed by human experts to extract a set of keywords to use it actually for query. Two sample queries for the experiments have 7 and 16 keywords, respectively. Some data was lost during the encoding for ontology (i.e., 4 of 7 and 5 of 16 keywords were lost), because the ontology only covered a small set of terminologies.

VI.4.3. Results analysis

Based on the 30 sample suppliers and 2 queries, we have continued experiments and evaluations of our work with help of domain experts at DSN Innovations Corp[34]. We compare our XML-based search method to two other methods: keyword-based and ontology-based methods with the same queries and same set of supplier samples. For the fair comparison with ontology-based method that lost some data, we established two experiments: 1) with the same sets of data as ontology-based matching approach uses (hereafter *partialXML*) – i.e., 3 and 11 keywords for two queries, respectively, and 2) with full sets of data which is the same as keyword-based search method uses (hereafter *fullXML*) – i.e., 7 and 16 keywords for two queries, respectively.

[34] http://www.dsninnovations.org/

Thus, four search methods (i.e., *keyword-based*, *ontology-based*, *partialXML-based*, and *fullXML-based*) based on two queries (i.e., "center post" and "swing arm") were executed to discover suppliers whose capabilities satisfy customer's requirements. The result of each search method is a ranked list of suppliers. To evaluate the search results, domain experts working in the field of supplier discovery were requested to analyze the requirements and capabilities. They manually produced a ranked list of matched (discovered) suppliers. Then, the discovery result of each search method was compared with the discovery results from human experts. Certain performance measures, such as Top-*k Recall* (the size of *k* is specified by human experts) and *nDCG*, were calculated by using similarity scores from suppliers based on the given queries and their rankings. The overall performances are as follows:

Table VI.1. Performances of Search Methods for Query #1

	Keyword	*Ontology*	*PartialXML*	*FullXML*
Top 4 *Recall*	0.500	0.500	0.250	0.250
Top 8 *Recall*	0.688	0.750	0.688	0.750
nDCG	0.664	0.703	0.692	0.727

Table VI.2. Performances of Search Methods for Query #2

	Keyword	*Ontology*	*PartialXML*	*FullXML*
Top 3 *Recall*	0.667	0.667	0.333	1.000
Top 6 *Recall*	0.500	0.500	0.500	0.500
Top 10 *Recall*	0.600	0.500	0.600	0.600
nDCG	0.837	0.902	0.848	0.920

For the fair comparison, we first compare the *fullXML-based* search with the *keyword-based* search, which use the same number of keywords for two queries (7 and 16, respectively). The overall performance measures show that the *fullXML-based* search performed better than the *keyword-based* search. Furthermore, *nDCG* performances show that the gap between performances of two search methods increased where more complex queries are used for the search – i.e., the *nDCG* gap of Table VI.2 (0.920 – 0.837 = 0.083) is bigger than the *nDCG* gap of Table VI.1 (0.727 – 0.664 = 0.063). This is because the XML-based method can more effectively analyze the complex data in a structured way than keyword-based method that utilizes the textual descriptions.

The second comparison is done between the *partialXML-based* search and the *ontology-based* search. Both also use the same number of keywords for two queries (3 and 11, respectively). The overall performances show that the *ontology-based* search performs better than the *partialXML-based* search. This is because *ontology-based* search can infer better logical relationships and similarity based on a formally defined semantics. However, if we compare the *ontology-based* search to the *fullXML-based* search, the performances show that the *fullXML-based search* performs better. This is because the *fullXML-based* search utilizes more information captured by XDF than that used in ontology-based search.

Our experiments have certain limitations in terms of the small size of the samples and manually encoded experimental data. The experimental results are not sufficient to show statistically significant differences. Based on the performance mea-

surements and experimental analysis, however, we can conclude that *XML-based* search method (1) shows significant promise to perform better than keyword-based search and (2) shows promise to perform at the same level or better as the ontology-based search when using all the available information, which sometimes may not be accessible by the ontology-based approach. The experiments also show that our optimal XML schema matching algorithm can be effectively used to enhance the search function to facilitate the supplier discovery.

CHAPTER VII

CONCLUSION

VII.1. Summary of Contributions

The objective of this dissertation is to develop effective approaches to XML schema matching, particularly applicable to XML schema integration and data transformation between heterogeneous e-Business systems. For that our research supports two different tasks: *integration task* between two *different component schemas*; and *transformation task* between two business documents which confirm to *different document schemas*.

For the *integration task*, we propose an innovative XML schema matching approach, called layered approach, which produces the best matching candidates between global type components of two component schemas, using their layer specific semantic similarities. For the *transformation task*, we propose another innovative XML schema matching approach, called optimization approach, which produces the *best sets* of matching pairs for all atomic nodes between two document schemas, based on semantics from both their linguistic and structural information.

Our approaches address three challenging problems in the schema matching. First, the existing approaches have often failed to sufficiently investigate and utilize semantics imbedded in the hierarchical structure of the XML schema which the schema designer intended to. Secondly, due to synonyms and polysemies found in natural languages, the meaning of a data node in the schema cannot be determined

solely by the words in its label. Last but not least, it is difficult to correctly identify the *best set* of matching pairs for all data nodes between two schemas.

We evaluate our approaches with the state of the art evaluation metrics and sample schemas obtained from several e-Business standard organizations and e-Business system vendors. Comparative analysis is conducted to validate the proposed approaches with a variety of experiences. In the layered approach, a series of experiments have been conducted with encouraging results. The system found a match to the human experts' matching in 35 of 49 cases in a real world application. The experiment results show promises to assist experts in accomplishing the integration tasks more efficiently. We also show that our approach can be made more efficient by grid computing in a service-oriented architecture.

In the optimization approach, we compared our result to some other XML schema matching approaches. The results were also encouraging. The average matching performances of *Precision*, *Recall*, *F-measure*, and *Overall* were 0.93, 0.83, 0.88, and 0.77, which are better than or at least equal to other approaches' performances. We also extend this technique to supplier discovery, a practically important problem in e-marketplaces. For that purpose, we proposed an eXtensible Dynamic Form (XDF) architecture to help capturing requirements and capabilities in a better structured way. The experiment results based on XDF and our XML matching approach show that the XML-based data collection and matching can be more efficient than traditional search or even ontology based search.

VII.2. Future Work

We have made significant advances in understanding and developing solutions for XML schema matching problems in various ways to support e-Business systems to be interoperable; however, still substantial work remains in order to achieve the goal of a comprehensive matching solution.

First, determining the best weights for combining individual similarity measures is a challenge in the proposed *c-matching*. Some machine learning techniques are under consideration, including regression and neural networks. Secondly, the label similarity measure is only used for inner-layers of layered approach. Our experiments show that labels at higher level are more important than the lower ones. There is also evidence that the atomic layer becomes more important when the structure of the element is shallow. It should be investigated, how to better incorporate the structural information into semantic analysis. Thirdly, the proposed *a-matching* only considers the *1-to-1* pair-wise matching. Other matching cardinalities, such as *1-to-many*, *many-to-1*, and *many-to-many*, should be also addressed to fully support the matching tasks. Last, a parallel and distributed computing technology, called MPJ, is used to improve the computation performance. Also, Hadoop and Globus Toolkit provide better functionalities than MPJ, for parallel and distributed computing such as security, resource and data management, communication, and fault detection. This calls for further examination of parallel and distributed computing technologies.

These researches described above will help to increase the performance of the matching algorithm. Furthermore, we state a few more work for future to better sup-

port the integrations and interoperability among the heterogeneous e-Business systems.

1) Our research only focuses on the matching activity between two XML schemas for the successful B2B systems integrations. However, to fully support B2B systems integrations, other activities, such as mapping, extending, and reusing, beyond matching should be also investigated.

2) The semantic resources used in this research are thesauri and dictionary such as WordNet. These resources are not domain-specific. For better matching of domain-specific schemas, it is necessary to utilize the domain-specific dictionaries. However, these domain-specific dictionaries require a substantial effort to be built up in a consistent way. Our matching approaches identify the matchings among words (terminologies) or word senses while analyzing the matching between two different schemas used by the same or similar domain industry. Those matchings can help to identify the domain-specific terminologies and their relationships. Furthermore, the matching results can be used as the initial resources to build up the domain-specific dictionaries. This calls for further investigation.

3) An XML-based form architecture and data matching applications are proposed to address the supplier discovery problems in the e-manufacturing domain. There are also many other domains that rely on schema matching such as data warehouse, e-government, e-health, and semantic query processing.

Our schema matching solutions can be also applied to these domains. This calls for further investigation.

REFERENCES

[AIAG 2008] AIAG (Accessed 2008), Automotive Industry Action Group. Available at http://www.aiag.org

[Aikins 1981] Aikins, Janice, Rodney Brooks, William Clancey, *et al*. (1981), "Natural Language Processing Systems," In *The Handbook of Artificial Intelligence*, Vol. I, Barr, Avron and Edward A. Feigenbaum (eds.), pp. 283-321. Stanford/Los Altos, CA: HeurisTech Press/William Kaufmann, Inc.

[Alan 2001] Kotok, A., and Webber, D. (2001), "ebXML: The New Global Standard," *New Riders Publishing Thousand Oaks*, CA, USA, ISBN: 0735711178.

[Ameri 2006] Ameri, F., and Dutta, D. (2006), "An upper ontology for manufacturing service description," *ASME 2006 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Philadelphia, September 10-13, 2006.

[Ariba 2000] Ariba (2000), "B2B marketplaces in the new economy," *Research report*. Available at http://www.ariba.com/com_plat/white_paper_form.cfm

[Asanovic 2006] Asanovic, K., Bodik, R., Catanzaro, B.C., Gebis, J.J., Husbands, P., Keutzer, K., Patterson, D.A., Plishker, W.L., Shalf, J., Williams, S.W., and Yelick, K.A. (2006), "The landscape of parallel computing research: a view from Berkeley," *Technical Report No. UCB/EECS-2006-183, EECS Department, University of California, Berkeley*.

[Aumüller 2005] Aumüller, D., Do, H.H., Massmann, S., and Rahm, E. (2005), "Schema and ontology matching with COMA++," In *Proceedings of the International Conference on Management of Data (SIGMOD)*, Software Demonstration.

[Baader 2003] Baader, F., and Nutt, W., (2003) "Basic Description Logics." In F. Baader, D. Calvanese, D. McGuiness, D. Nardi and P. F. Patel-Schneider (eds.), *The*

Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, Cambridge, pp. 43–95

[Bellahsene 2011] Bellahsene, Z., Bonifati, A., Rahm, E. (2011), "Schema Matching and Mapping," Springer, ISBN: 978-3-642-16517-7.

[Bellman 2003] Bellman, R.E. (1957), "Dynamic Programming," *Princeton University Press*, Princeton, NJ, Republished 2003: Dover, ISBN 0486428095.

[BizTalk 2010] BizTalk Server (Accessed 2010). Available at http://www.microsoft.com/biztalk/

[Borthaku 2007] Borthaku, D. (2007), "The Hadoop distributed file system: architecture and design." Available at http://hadoop.apache.org/common/docs/r0.17.2/hdfs_design.html

[Boukottaya 2005] Boukottaya, A., and Vanoirbeek, C. (2005), "Schema Matching for Transforming Structured Documents," In *DocEng*, pp. 2-4, DOI: 10.1145/1096601.1096629.

[Bussler 2001] Bussler, C. (2001), "B2B protocol standards and their role in semantic B2B integration engines," *Bull Tech Comm Data Eng*, 24 (1), pp. 3–11.

[Carmel 2002] Carmel, D., Maarek, Y., Mass, Y., Efraty, N., and Landau, G. (August 2002), "An Extension of the Vector Space Model for Querying XML documents via XML fragments," In *ACM SIGIR 2002 Workshop on XML and Information Retrieval*, Tampere, Finland.

[Carpenter 2000] Carpenter, B., Getov, V., Judd, G., Skjellum, A., and Fox, G. (2000), "MPJ: MPI-like message passing for Java," *Concurrency: Practice and Experience*, 12 (11), pp. 1019-1038, DOI: 10.1.1.35.9869.

[Christopher 2004] Christopher, M., and Peck, H. (2004), "Building the resilient supply chain," *International Journal of Logistics Management*, 15 (2), pp 1-13.

[Cover 1991] Cover, T.M., and Thomas, J. A. (1991), "Elements of information theory," *Wiley series in telecommunications*, New York, ISBN: 0-471-24195-4.

[Dean 2004] Dean, J., and Ghemawat, S. (Dec. 2004), "MapReduce: Simplified Data Processing on Large Clusters," In *Proceedings of OSDI '04: 6th Symposium on Operating System Design and Implemention*, San Francisco, CA.

[Do 2003] Do, H.H., Melnik, S., and Rahm, E. (2003), "Comparison of Schema Matching Evaluations," *Lecture Notes in Computer Science*, 2593, pp. 221-237. DOI: 10.1.1.11.4792.

[Doan 2001] Doan, A.H., Domingos, P., and Halevy, A. (2001), "Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach," In *Proceedings of ACM SIGMOD 2001 Conference*, pp 509-520.

[Douglas 1999] Douglas, W.B. (1999), "Introduction to Graph Theory," (2nd Edition), *Prentice Hall*, Chapter 3, ISBN: 0-13-014400-2.

[Dulmage 1958] Dulmage, A.L., and Mendelsohn, N.S. (1958), "Coverings of bipartite graphs," *Canadian Journal of Mathematics*, 10, pp. 517–534.

[Dumas 2004] Dumas, M., Benatallah, B., Russell, N., and Spork, M. (Spring 2004), "A configurable matchmaking framework for electronic marketplaces," *Electronic Commerce Research and Applications*, 3 (1), pp. 95-106.

[ebXMLCC 2008] (Accessed 2008), ebXML Core Components. Available at http://www.ebxml.org

[Foster 1997] Foster, I., and Kesselman, C. (1997), "Globus: a metacomputing infrastructure toolkit," *The International Journal of Supercomputer Applications and High Performance Computing*, 11 (2), pp. 115-128. DOI: 10.1177/109434209701100205.

[Giunchiglia 2004] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, "S-Match: an algorithm and an implementation of semantic matching," Proc. of the European Semantic Web Symposium (ESWS), 2004, pp. 61–75, DOI: 10.1007/978-3-540-25956-5_5.

[Guarino 1995] Guarino, N. (1995), "Formal Ontology: Conceptual Analysis and Knowledge Representation," *International Journal of Human-Computer Studies*, 43(2/3), pp. 625-640.

[He 2004] He, B., and Chang, K.C.C. (2004), "A holistic paradigm for large scale schema matching," *SIGMOD Record*, 33 (4), pp. 20-25, DOI: 10.1.1.58.7651.

[HL7 SAIF 2010] Health Level 7 (April 2010), HL7 Services-Aware Interoperability Framework (SAIF).

[Hoare 1962] Hoare, C.A.R. (1962), "Quicksort," *Computer Journal*, 5 (1), pp. 10-15.

[Hovy 1999] Hovy, E.H., and Lin, C.Y. (1999), "Automated Text Summarization in SUMMARIST," in M. Maybury and I. Mani (eds.), *Advances in Automatic Text Summarization*, Cambridge: MIT Press.

[IV&I 2008] IV&I (Accessed 2008), AIAG Inventory visibility & interoperability. Available at http://www.aiag.org/committees/IVPE.cfm

[Jang 2008] Jang, J., Jeong, B., Kulvatunyou, B., Chang, J., and Cho, H. (2008), "Discovering and integrating distributed manufacturing services with semantic manufacturing capability profiles," *International Journal of Computer Integrated Manufacturing*, 21 (6), pp. 631–646.

[Jarvelin 2002] Jarvelin,K., and Kekalainen J. (2002), "Cumulated gain-based evalution of IR techniques," *ACM Transactions on Information Systems*, 20 (4), pp. 422-446.

[Kondrak 2005] Kondrak, G. (2005), "N-gram similarity and distance," In *Proceedings of International Conference on String Processing and Information Retrieval*, pp. 115-126.

[Kulvatunyou 2005] Kulvatunyou, B., Cho, H., and Son, Y.J. (2005), "A semantic web service framework to support intelligent distributed manufacturing," *Interna-*

tional Journal of Knowledge-based and Intelligent Engineering Systems, 9, pp. 107–127.

[Langenberg 2005] Langenberg, T. (2005), "Standardization and Expectations," *Germany: Springer-Verlag Berlin and Heidelberg GmbH & Co. KG*, ISBN-10: 3540281126, ISBN-13: 978-3540281122.

[Lerner 2000] Lerner, B.S. (2000), "A model for compound type changes encountered in schema evolution," *ACM Transactions on Database Systems*, 25 (1), pp. 83-127, DOI: 10.1.1.105.1542.

[Lin 1998] Lin, D. (1998), "An Information-theoretic definition of similarity," In *Proceedings of the 15th International Conference on Machine Learning*, pp. 296-304, DOI: 10.1.1.55.1832.

[Lin 2000] Lin, C.Y., and Hovy, E.H. (2000), "The Automated Acquisition of Topic Signatures for Text Summarization," In *Proceedings of the COLING Conference. Strasbourg*, France, DOI: 10.3115/990820.990892.

[Madhavan 2001] Madhavan, J., Bernstein, P.A., and Rahm, E. (2001), "Generic schema matching with Cupid," In *Proceedings of the 27th International Conference on Very Large Data Bases*, pp. 49-58. DOI: 10.1.1.17.4650.

[Makhoul 1999] Makhoul, J., Kubala, F., Schwartz, R., and Weischedel, R. (February 1999), "Performance measures for information extraction," In *Proceedings of DARPA Broadcast News Workshop*, Herndon, VA.

[Markov 2007] Markov, Z., and Larose, D.T. (2007), "Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage," *Wiley*, ISBN: 978-0-471-66655-4.

[Meadows 2004] Meadows, B., and Seaburg, L. (September 2004), "Universal Business Language 1.0," *Organization for the Advancement of Structured Information Standards (OASIS)*, Committee Draft.

[Medjahed 2003] Medjahed, B., Benatallah, B., Bouguettaya, A., Ngu, A., and Elmagarmid, A. (May 2003), "Business-to-business interactions: issues and enabling technologies," *VLDB Journal*, 12 (1), pp. 59–85, DOI: 10.1007/s00778-003-0087-z.

[Melnik 2002] Melnik, S., Garcia-Molina, H., and Rahm, E. (2002), "Similarity flooding - a versatile graph matching algorithm," In *Proceeding of 18th International Conference of Data Engineering*, pp. 117-128, DOI: 10.1.1.61.4266.

[Miller 1995] Miller, G.A. (1995), "WORDNET: a lexical database for English," *Communications of ACM*, 38 (11), pp. 39-41, DOI: 10.1145/219717.219748.

[Milo 1998] Milo, T., and Zohar, S. (1998), "Using schema matching to simplify heterogeneous data translation," In *Proceeding of the 24th International Conference on Very Large Data Bases*, pp. 122-133, DOI: 10.1.1.30.2620.

[Mong 2002] Mong, L.L., Liang, Y.H., Wynne, H., and Xia, Y. (November 2002), "XClust: Clustering XML Schemas for Effective Integration," In *Proceedings in 11th ACM International Conference on Information and Knowledge Management (CIKM)*, McLean, Virginia, DOI: 10.1145/584792.584841.

[Murphy 2008] Murphy, C.N., and Yates, J. (2008), "The International Organization for Standardization (ISO) : Global Governance Through Voluntary Consensus," *New York: Routledge*, ISBN-10: 0415774292, ISBN-13: 978-0415774291.

[Noia 2005] Noia, T.D., Sciascio, E.D., Donini, F.M., and Pinto, A. (2005), "Ontology-based natural language parser for e-marketplaces," In *18th Intl. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, *vol. 3533 of Lecture Notes in Artificial Intelligence*.

[Noy 2001] Noy, N., and Musen, M. (2001), "Anchor-PROMPT: using non-local context for semantic matching," In *Proceedings of the workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 63–70.

[OAGIS 2002] OAGIS (2002), "Open Applications Group Interface Specification V9.0." Available at http://www.openapplications.org/

[Papadimitriou 1981] Papadimitriou, C.H. (1981), "On the complexity of integer programming," *J. ACM*, 28, pp. 765–768.

[Peng 2006] Peng, Y. (2006), "On Semantic Similarity Measures", *Technical Report from Syllogism*, Com to NIST.

[Qin 2009] Qin, P., Lu, Z., Yan, Y., and Wu, F. (2009), "A New Measure of Word Semantic Similarity Based on WordNet Hierarchy and DAG Theory," In *Proceedings of International Conference on Web Information Systems and Mining*, pp. 181-185, DOI: 10.1109/WISM.2009.44.

[Rada 1989] Rada, R., Mili, H., Bicknell, E., and Blettner, M. (1989), "Development and application of a metric on semantic nets," *IEEE Transaction on Systems, Man, and Cybernetics*, 19 (1), pp. 17-30, DOI: 10.1109/21.24528.

[Rahm 2001] Rahm, E., and Bernstein, P.A. (2001), "A survey of approaches to automatic schema matching," *VLDB Journal*, 10 (4), pp. 334-350, DOI: 10.1007/s007780100057.

[Resnik 1995] Resnik, P. (1995), "Using information content to evaluate semantic similarity in a taxonomy," In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 448-453, DOI: 10.1.1.55.5277.

[Resnik 1999] Resnik, P. (1999), "Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language," *Journal of Artificial Intelligence Research*, 11, pp. 95-130.

[van Rijsbergen 1979] van Rijsbergen, C.J. (1979), "Information retrieval," (2nd Edition), *London: Butterworths*, ISBN-10: 0408709294, ISBN-13: 978-0408709293.

[Saleem 2008] Saleem, K., Bellahsene, Z., and Hunt. E. (2008), "PORSCHE: performance oriented schema mediation," *Information Systems*, 33 (2), pp. 637-657. DOI: 10.1016/j.is.2008.01.010.

[Shim 2000] Shim, S.Y., Pendyala, V.S., Sundaram, M., and Gao, J.Z. (Oct. 2000), "Business-to-business e-commerce frameworks," *IEEE Computer*, 33 (10), pp. 40-47, DOI: 10.1109/2.876291.

[Shvaiko 2005] Shvaiko, P., and Euzenat, J. (2005), "A survey of schema-based matching approaches," *Journal on Data Semantics IV*, *LNCS 3730*, pp. 146-171, DOI: 10.1007/11603412_5.

[Sneath 1957] Sneath, P.H.A. (1957), "The application of computers to taxonomy," *Journal of General Microbiology*, 17 (1), pp. 201–226.

[STAR 2008] STAR (Accessed 2008), Standard for Technology in Automotive Retail. Available at http://www.starstandard.org

[TBG 2008] TBG (Accessed 2008), UN/CEFACT International Trade and Business Processes Group (TBG). Available at http://www.disa.org

[Thang 2008] Thang, H.O., and Nam, V.S. (2008), "XML Schema Automatic Matching Solution," *International Journal of Computer Systems Science and Engineering*, 4 (1), pp. 68-74.

[Tversky 1977] Tversky, A. (1977), "Features of similarity," *Psychological Review*, 84 (4), pp. 327-352.

[W3C 1998a] W3C Recommendation, Extensible Markup Language (XML) 1.0 (1998). Available at http://www.w3.org/TR/REC-XML

[W3C 2001a] W3C Recommendation, XML Schema Part 0: Primer (2001). Available at http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/

[W3C 2001b] W3C Recommendation, XML Schema Part 1: Structures (2001). Available at http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[W3C 2001c] W3C Recommendation, XML Schema Part 2: Datatypes (2001). Available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[Yang 2005] Yang, D., and Powers, D.M.W. (2005), "Measuring semantic similarity in the taxonomy of WordNet," In *Proceedings of the 28th Australasian Computer Science Conference*, pp. 315-322.

APPENDICES

VII.3. Appendix A – Recommended Naming Rule of XML Schema

1) Each element or attribute XML name MUST have one and only one fully qualified XPath (FQXP).

   For example: Communication/Address/StreetName

2) Element, attribute and type names MUST be composed of words in the English language, using the primary English spellings provided in the WordNet 3.0 (WordNet 3.0 online service: http://wordnetweb.princeton.edu/perl/webwn).

3) Either Lower camel case (LCC) or Upper camel case (UCC) MUST be used for naming attributes, elements and types.

4) Element, attribute and type names MUST be in a singular form unless the concept itself is plural.

   Example of Singular and Plural concept forms:

   Singular – Allowed: <xsd:element name=”GoodsQuantity” …>

   Plural – Not Allowed: <xsd:element name=”ItemsQuantity” …>

5) Element, attribute and type names MUST be drawn from the following set: a – z and A – Z.

   Example of Non-Letter Characters – Not Allowed

   <xsd:element name=”LanguageCode8” …>

6) XML element, attribute and type names constructed from dictionary entry names MUST NOT include periods, spaces, or other separators; or characters not allowed by W3C XML 1.0 for XML names.

Example of Spaces in Name – Not Allowed <xsd:element name=”Customized_ Language. Code:8” …>

7) XML element, attribute and type names MUST NOT use acronyms, abbreviations, or other word truncations except those included in the UN/CEFACT controlled vocabulary or listed in Appendix B.

Example Acronyms and Abbreviations

ID is an allowed abbreviation: <xsd:element name=”ID”>

Cd is not an approved abbreviation : <xsd:element name=”ReasonCd”>

8) Acronyms and abbreviations at the beginning of an attribute declaration MUST appear in all lower case. All other acronyms and abbreviation usage in an attribute declaration must appear in upper case.

9) Acronyms MUST appear in all upper case for all element declarations and type definitions.

VII.4. Appendix B –Abbreviations and Acronyms Accepted by XML Schema Matching

VII.4.1. Acronyms

- BOD – Business Object Document
- BOM – Bill of Material
- DUNS – Data Universal Numbering System
- EFT – Electronic Funds Transfer
- GL – General Ledger
- HR – Human Resources
- HTML – Hyper Text Markup Language
- SCE – Supply Chain Execution
- UOM – Unit of Measure
- URI – Uniform Resource Identifier
- URL – Uniform Resource Locator
- WIP – Work In Process

VII.4.2. Abbreviations

- Class – Classification
- Doc – Document
- Enum – Enumeration
- ID – Identifier
- Ind – Indicator
- Max – Maximum
- Min – Minimum
- Ship – Shipment
- Sync – Synchronize

VII.4.3. Non-Oxford

- ABC Classification
- Subentity
- Subline

VII.5. Appendix C – Stop Words Removed by XML Schema Matching

This stop word list is obtained from Onix Text Retrieval Toolkit[35]. This contains 429 words as follows:

a, about, above, across, after, again, against, all, almost, alone, along, already, also, although, always, among, an, and, another, any, anybody, anyone, anything, anywhere, are, area, areas, around, as, ask, asked, asking, asks, at, away, b, back, backed, backing, backs, be, became, because, become, becomes, been, before, began, behind, being, beings, best, better, between, big, both, but, by, c, came, can, cannot, case, cases, certain, certainly, clear, clearly, come, could, d, did, differ, different, differently, do, does, done, down, down, downed, downing, downs, during, e, each, early, either, end, ended, ending, ends, enough, even, evenly, ever, every, everybody, everyone, everything, everywhere, f, face, faces, fact, facts, far, felt, few, find, finds, first, for, four, from, full, fully, further, furthered, furthering, furthers, g, gave, general, generally, get, gets, give, given, gives, go, going, good, goods, got, great, greater, greatest, group, grouped, grouping, groups, h, had, has, have, having, he, her, here, herself, high, high, high, higher, highest, him, himself, his, how, however, i, if, important, in, interest, interested, interesting, interests, into, is, it, its, itself, j, just, k, keep, keeps, kind, knew, know, known, knows, l, large, largely, last, later, latest, least, less, let, lets, like, likely, long, longer, longest, m, made, make, making, man, many, may, me, member, members, men, might, more, most, mostly, mr, mrs, much, must, my, myself, n, necessary, need, needed, needing, needs, never, new, new, newer, newest, next, no, nobody, non, noone, not, nothing, now, nowhere, number, numbers, o, of, off, often, old, older, oldest, on, once, one, only, open, opened, opening, opens, or, order, ordered, ordering, orders, other, others, our, out, over, p, part, parted, parting,

[35] http://www.lextek.com/manuals/onix/stopwords1.html

parts, per, perhaps, place, places, point, pointed, pointing, points, possible, present, presented, presenting, presents, problem, problems, put, puts, q, quite, r, rather, really, right, right, room, rooms, s, said, same, saw, say, says, second, seconds, see, seem, seemed, seeming, seems, sees, several, shall, she, should, show, showed, showing, shows, side, sides, since, small, smaller, smallest, so, some, somebody, someone, something, somewhere, state, states, still, still, such, sure, t, take, taken, than, that, the, their, them, then, there, therefore, these, they, thing, things, think, thinks, this, those, though, thought, thoughts, three, through, thus, to, today, together, too, took, toward, turn, turned, turning, turns, two, u, under, until, up, upon, us, use, used, uses, v, very, w, want, wanted, wanting, wants, was, way, ways, we, well, wells, went, were, what, when, where, whether, which, while, who, whole, whose, why, will, with, within, without, work, worked, working, works, would, x, y, year, years, yet, you, young, younger, youngest, your, yours, z