Ontological and Conceptual Structures for Tendering Automation

Ahmad Kayed
Robert M. Colomb

Computer Science and Electric Engineering,
University of Queensland
Brisbane, Australia
Email: kayed, colomb@csee.uq.edu.au

Abstract

The aim of method presented in this paper is to build infrastructure for business-to-business electronic commerce (EC). To narrow the scope we are focusing on a specific business process i.e. the tendering process. We provide ontological structures for tendering system, which can be reused. This will help in using knowledge to support any business process. Using formal structures has advantages over other approaches (e.g. EDI). Here, we need only to agree about the common ontology. This is more flexible and can be stored in knowledge-based system. This will help in matching and reasoning. In this paper, a framework for ontological based tendering system is introduced. The components of the ontologies are clarified. Conceptual Graph (CG) formalisms to build tendering ontologies are identified. Finally we demonstrate how the ontological structures solve many problems in the online tendering domain.

Keywords

Ontology, Conceptual Structures, Software agents, Business-to-Business E-Commerce, Tendering process.

INTRODUCTION

Technology development represents a powerful driving force for the establishment of new methods in managing and organizing public procurement processes (Blomberg and Lennartsson, 1997). Electronic Tendering may contribute to increase efficiency and effectiveness of the procurement process in terms of costs, quality, performance and time for both buyers and sellers. Efficiency and effectiveness will be increased by applying electronic tendering techniques in terms of cuts in manpower costs, reduced administrative and transaction costs, improvements in tender quality, strengthened tender preparation capacity, simplified public market access, competitiveness, and high integration capability with internal and external systems (Blomberg and Lennartsson, 1997) (Slone, 1992).

Using the Internet as the underlying platform for tendering automation involves many problems. Examples of these problems are: security, authentication, heterogeneity, interoperability, and ontology problems. Internet users need tools to search for information across heterogeneous systems and to match potential data sources with their needs. Consumers also need to search for information using terms from domains they are familiar with (Ontologies) (Adam et al., 1998). Automating any business process needs to reuse and distribute information among different parties. This raises two problems: the need of a common vocabulary (ontology) and the need of common protocol and management model (standards).

This paper contributes in this direction. We are building online tendering system focusing on helping buyers to write their tender, solving the ontological problems, and giving more potential to e-mediators. Our system is composed of three components: Agent-based component for tender forming, knowledge-based component to store tendering information, and mediator component to perform matchmaking and bid filtration. These components depend on conceptual structures, which are kept in different types of ontologies. The main aim of this paper is to define the ontological structures needed for the tendering process. To give the flavor of these structures we first give a brief view of our system then we detail these structures. The reader encouraged to read (Kayed and Colomb, 1999) (Kayed, 2000a) or visit (Kayed, 2000b) for more details. Section two identifies the roles of ontological and conceptual structures in automating the tendering process. Section three introduces the ontological conceptual structures for tendering system. Section four demonstrates our solution and section five concludes the paper.

THE ROLES OF CONCEPTUAL STRUCTURES

A knowledge representation system of some kind is necessary to deal with the complexity of the public tendering process, to support semantic matching, software agent communication, e-mediator, and reason about tendering information.

Using natural language to model tendering makes any process associated with tendering automation extremely difficult. We need to define structures and their contents that will store the tendering information. Since we are interested in storing the information in a knowledge base, the structures should be modeled in logical or formal way. These structures should contain what is called a context (Sowa, 1995) (Sowa, 1997) (Kashyap and Sheth, 1996), which will help in matching and reasoning. The context provides a mapping mechanism between different terminologies through the ontology. Using formal structures has advantages over the standardized approach (e.g. EDIFACT messages). The EDI approach needs a pre-agreement about everything, but here we just need to agree about the common ontology. The ontology contains abstract concepts that will form the primitives to construct a tender or a bid. This is more flexible and can be stored in knowledge base. The ontology will make it easy to build tools to transfer from a friendly user interface (like the Web) to a logical structure (knowledge base).
In the tendering process domain, ontology is needed to solve many heterogeneity problems (Kayed and Colomb, 1999). Buyers/sellers in a specific domain can use a common ontology to describe their needs/offers. The huge amount of information on the Internet prevents buyers and sellers from finding relevant information that meets their needs. If the tendering information is committed to a common ontology, this will make easy to find the relevant information for all parties.

In the tendering domain, different activities may need different types of matching. As an example, in the invitation activity the buyers advertise general information about the tender, while the seller profiles are more specific. In the contrary, when a seller is looking for "good" tenders, the tenders are more specific than the sellers' profiles. Different activities need different types of matching. Formal ontologies define the roles, the concepts, and the relations between concepts. This allows us to define many compatibility measures to check the similarity between concepts. This will facilitate different type of matching among buyers and sellers. A formal ontology with different type of relation can solve the different level of abstractions in the call for bid invitation.

The procurement process requires mediation between buyers and sellers. In the e-commerce environment, software agents will play this role. In a community governed by software agents, ontologies become a central issue for the development of any agent-based systems. Ontological-based tendering system will help in testing the feasibility of the ontological approach, which will contribute in building a new generation of business to business e-commerce.
Our solution will deploy the mediator concepts to build a shared ontology. The mediator will be responsible for maintaining different types of ontologies and performing different types of matching. This will facilitate the automation of many activities such as tender forming, buyer and seller matching, bid evaluation and other activities.

Ontology

The term Ontology has its roots in philosophy which has been defined as a particular theory about the nature of being or the kinds of existence (Merriam-Webster, 1999). In the computer science community, ontology has become an important issue. Many research areas study and use ontology in various ways, fields such as Artificial Intelligence (AI), knowledge-based systems, language engineering, multi-database systems, agent-based systems, information systems, etc (Guarino, 1998).

Ontology is a convention about the meaning of things. Ontology is a way of categorizing objects such that they are semantically meaningful. The Ontology in the AI and database sense is an elaborate conceptual schema of generic domain concepts and relations with constraint or axioms, and with a formal lexicon or concept-dictionary (Lehmann, 1995). Guarino (Guarino, 1998) defines the ontology as an explicit, partial account of a conceptualization. Gruber (Gruber, 1995) defines it as an explicit specification of a conceptualization.
Different types of ontology can be defined according to their level of generality. In a large community a unified Top-level ontology can be used to describe very general concepts (Guarino, 1998). The vocabulary related to a generic domain or activity can be describe in the domain and task ontology. At the application level a related application ontology can use a concept from both the task and domain ontology (Guarino, 1998).

There are axiomatic-based and taxonomic-based ontologies. Axiomatic-based defines small primitive concepts and axioms that define the relation among these concepts. Taxonomic-based ontologies can be constructed using a large taxonomy (e.g. Cyc) (Elkan and Greiner, 1993) which contains about 100,000 concepts (Noy and Hafner, 1997) or they can be structured around a number of smaller taxonomies such as Toronto Virtual Enterprise (TOVE) (Fadel et al., 1994) (Shadolt et al., 1996).

Wand and Weber used ontology in different way. They defined a meta-ontology to evaluate information system modeling techniques (Wand and Weber, 1990). They defined what so called Bunge-Wand-Weber (BWW) model [Weber, 1997]. BWW model defines about 30 ontological constructs. According to these constructs the completeness and efficiency of a modeling technique can be measured. If there is one-to-one relation between these constructs and a modeling technique like ER then we can say that this technique is complete or efficient. We will use Bunge (Bunge, 1977) properties in designing our Domain Ontology. We will leave the evaluation of CG for future work.

Conceptual Graphs and Ontologies

Conceptual Graphs (CGs) are a method of knowledge representation developed by Sowa (Sowa, 1984) based on Charles Peirce's Existential Graphs and semantic networks of artificial intelligence (Sowa, 1995). According to Sowa (Sowa, 1984), CGs have a direct mapping to and from natural language and a graphic notation designed for human readability. Conceptual graphs have all the expressive power of logic but more intuitive and readable. Many popular graphic notations and structures ranging from type hierarchies to entity-relationship or state transition diagrams can be viewed as special cases of CGs (Way, 1994). CGs are semantically equivalent graphic representation for first order logic (FOL) like Knowledge Interchange Format (KIF).
CGs can model many organizational concepts like roles or actors (Sowa, 1998), process (Mineau, 1999), events, procedures, state, situations, and context (Sowa, 1995). As well there are many tools to represent CG in XML formats (Martin and Eklund, 1999). In our project, we have used CGs as our implementation language. We are aware of some problems of CGs to model ontology. Mineau (Mineau, 1993) argues that CG can be used easily to represent an ontology. He argues that with some work in CG it is more suitable than Ontolingua (Gruber, 1993). Ontolingua is a formal ontology description language. Ontolingua consists of a KIF parser, tools for analyzing ontologies, and a set of translators for converting Ontolingua sources into forms acceptable to implemented knowledge representation systems.

The one-to-one mapping between KIF and CG makes it possible to implement ontology in CGs. Ontology can be implanted in CG by using a second order CG. The type definition in CGs can be used to replace the Define-class construct in Ontolingua. The second order Graph, the lambda expression, and the canonical basis are sufficient constructs to implement ontologies in CGs.
There are two ways of defining things: by stating necessary and sufficient conditions or by giving a few examples and saying that everything similar to these belongs to that definition ((Sowa, 1984), pp104). The former way is logically easier to handle than the latter. CGs supports both mechanisms. In CG the Type definition is sufficient and necessary condition for any definition, while prototype function (which has the same semantic as type function) is not.
In our project we use both. Some situations need a precise definition, in this case we use the type definition. Many situations do not, for there we use the prototype definition. Usually, for ontological conceptual structures we use the type definition while for templates we use the prototype definition. Using the prototype definitions is flexible and helps us in modifying the ontology. Any new structure is defined by the prototype definition. All the specialized definition will be kept then a generalized structure will be defined in the type definition.

Three things influence us to use CGs. First the CG components (the relation and concepts catalogs, the type hierarchy, the canonical basis) are suitable for decomposing the ontology to those components. Second the context (situation where the CG is assertion) is good to represent the lifting axioms and to define different type of matching. Third the one-to-one mapping between CG and KIF (KIF was the original Ontolingua language).
In the following subsection we will introduce the tendering ontologies and show how we used the CGs to build these ontologies.

TENDERING ONTOLOGICAL STRUCTURES

Before we introduce our tendering ontological structures we will discuss the theory that influence us in designing our models.

Designing theoretical basis

Many ontology designers do not distinguish between word (symbol) and concept. Concept is a particular conceptualization of an element of the domain of discourse, and each concepts can be denoted by one or more word (Campbell and Shapiro, 1998). Guarino (Guarino, 1998) defines a conceptualization as a set of conceptual relations on a domain space, where a domain space is a set of possible states of affairs within a specific domain". Ogden and Richards (Ogden and Richards, 1946) established the meaning triangle as a means for expressing the relationships among symbols, concepts, and referents (Tepfenhart, 1998). In the lower left corner of the triangle is the symbol, which corresponds to the linguistic element of a word. The lower right corner is the referent, which is related to the object. The top of the triangle is the concept, which serves to link the symbol and the referent. The direct link between symbol and referent is actually a virtual link. The symbol invokes in the mind of an individual the concept. Alternatively, one may view the link as the symbol expresses the concept. The relation between the concept and the referent is more complex. The referent is observed and expressed as a percept. The percept is then interpreted as a concept (Tepfenhart, 1998).

As a result, developing ontology and grounding the semantics of concepts in the physical word is a matter of studying objects and how we observe them. So we have two approaches that help in designing conceptual structure. The first depends on the percepts and the second depends on the linguistics where the semantic of a concept is based on the result view invoked by the symbol.
Existing modelling tools tend to be neutral with respect to the ontological assumptions of the users. These assumptions are often hidden and tied to a particular task. This leads to difficulties in integration and reusing existing knowledge-based system. These system need to be rebuilt from scratch when a new problem is addressed (Guarino and Poli, 1995). We understand the ontology as an enhanced step in knowledge representation. To make knowledge-based systems more scalable we need to implement the ontology as abstract knowledge which can be reused. We agree with Mineau (Mineau, 1993) that the ontology should contain both the concepts of the domain and a set of knowledge formation operators which allow the concepts to be expanded, and which control how the objects of the domain could be represented from these concepts.

World is made up of things and information is a representation of some phenomenon in this world (Weber, 1997). We know about things in the world via their properties by the models of things that we create. Bunge (Bunge, 1977) defined many types of properties for things. In information system six of them are important (Weber, 1997). The main six properties are (Weber, 1997): properties in general and in particular, intrinsic and mutual properties, and hereditary and emergent properties. Bunge's properties should be explicitly defined in the ontology. We classify concepts around common classes. A class of things has common properties that belong to this class. To say that two classes are similar, we compare their properties. These properties help us to differentiate between the "type" and "role" concepts. As example: Person is a type of organism while student is a role of person, the different here is in the intrinsic (native) property. Usually "type" subsumed the "role" concepts but not vice-versa.

Tendering ontologies

Depending on the previous discussion, in our project we divided the ontology into three parts: collections of concepts, collections of conceptual structures, and collections of formal contexts. These all form our ontology (see figure 1). The collections of concepts help us to build tools for

translation and integration from one domain to another. The Concepts part consist of three sub-parts. Those are: the catalog vocabularies, the relation vocabularies, and the hierarchical relation between concepts (the type function in CG). The Conceptual Structures (CS) represents the basic element for the tendering system. Software agents use these CSs to communicate and interact. Buyers, sellers, and mediator use these structures to describe their need, offers, responses, or queries (the canonical basis in CGs). The formal context will provide the mechanisms of defining the similarities between concepts. The formal contexts contain three parts: the intentions graph (the graph in which the graph will be asserted), the lifting axioms, and the relation (type-of, is-a, part-of, etc.). The lifting axioms help us in reusing the ontologies and knowledge.

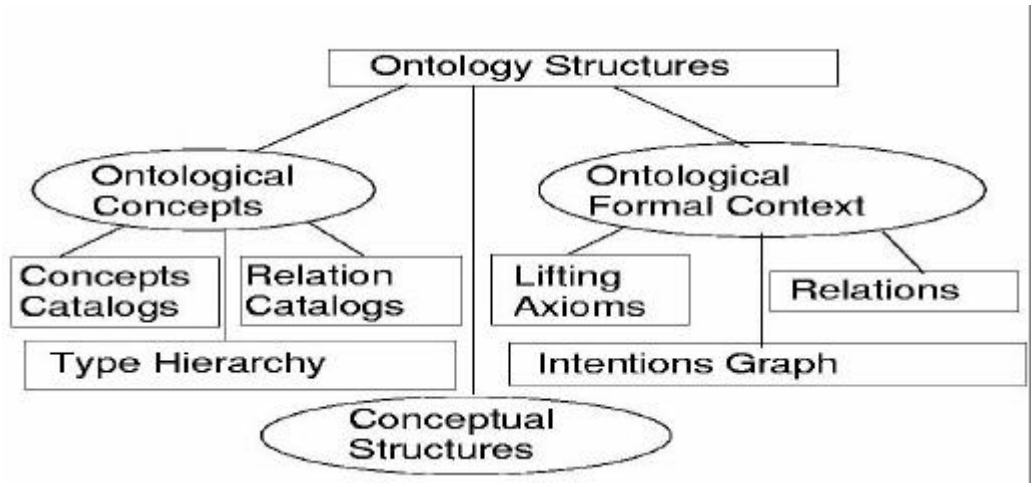

Figure 1: Ontology Structure

Our approach of decomposing the ontological constructions into three parts helps us building the tendering ontologies. Building ontology is still an art, not a science (Fernandez et al., 1996), despite some attempts to define a methodology for ontology construction (Gomez-Perez and Rojas-Amaya, 1999) (Fernandez et al., 1999) (Uschold, 1996).

Decomposition of the ontology facilitates the implementation of large-scale knowledge bases. It maintains efficiency for high-performance knowledge representation systems. (Hendler and Stoffel, 1999) divide their ontology into structural assertion (e.g. Is-a) and non-structural assertion (all other concepts in the KB). They claim that the number of structural assertions is less that the number of non- structural assertions. Depending on that they kept the structural assertions in a cache memory where the non-structural assertions are kept in secondary memory.

Following our framework (Kayed and Colomb, 1999), we need four types of ontologies: meta-ontology, abstract domain ontology, domain ontology, and tendering ontology (see figure 2). In the following we will describe each one.

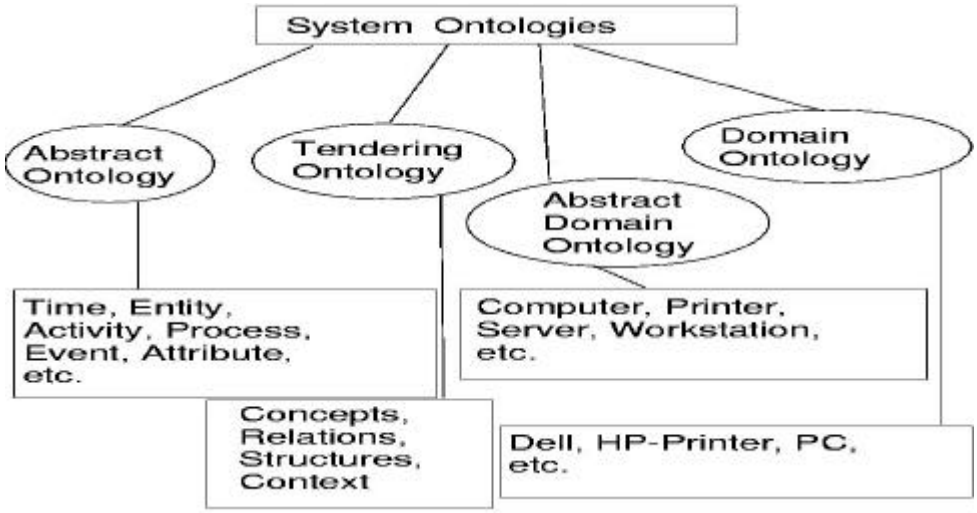

Figure 2: System Ontologies

The Meta-Ontology defines (describes) very general concepts for other ontologies. The meta-Ontology helps to query the domain ontologies and to translate from and to the domain ontologies. This is a very abstract ontology and we build its components from other generic ontologies like (Farquhar et al., 1997), (Elkan and Greiner, 1993), and (Uschold et al., 1998). We reuse the definition of time (Date, Days, Years, Hours) from the ontology server (Farquhar et al., 1997). We take the basic unit measures from Cyc ontology server (Elkan and Greiner, 1993). Cyc is common-sense knowledge base ontology used to relate concepts from different models. We also redefine some organizational concepts like Entity, Buyer, Seller, Agent, Activity, Process, etc. from the Enterprise ontology (Uschold et al., 1998).

The Abstract Domain Ontology contains *Classes* which are abstract description of objects in a domain. The class has Class-ID, Class-Properties, Class-Synonyms, Class-Type, Relation, Sub-Class, and Axioms. A relation is a link between classes, and axioms are rules that govern the behavior of the classes. The abstract domain ontology represents a container of abstract data types for sellers' catalogs. In this sense we should distinguish between the catalogs and the ontology. Ontology may contain a PC as a concept, which has RAM and CPU as other concepts. Catalog may contain Pentium 3 with 32 MB RAM. In CGs sense, this can be translated to [PC:Dell]? (Part-Of)? [CPU-Type: Pentium 3]? (part-Of)? [Memory: RAM] ? (measure)? [Memory-Unit: 32 MB]

The Domain Ontology is a collection of vocabularies mapped to concepts in the Abstract Domain Ontology (ADO). Since the ADO is a schema for the sellers' catalogs, we should define mapping between these abstract concepts (in the ADO) and the catalog values. This is how we know that Dell computer is a PC concept. We can keep this ontology in the mediator side or each seller can create it locally. Normally, this ontology is huge and constructed from the catalogs. If there are some values are not mapped to the abstract domain ontology, the mediator may add new concepts, which are relevant to that value. Mediator also can provide [unknown] concept, which mapped these values, then later add new concepts for these values.

The Tendering Ontology represents the core ontology in our system. The basic part of it is the Tendering Conceptual Structures (TCSs). We divide them into three models: buyer, seller, and mediator models. The buyer model is divided into advertising model, query model, and policy model. The advertising model again can be divided into tender invitation, terms, objects (services), specification, and returned forms. It is beyond the scope of this paper to describe everything in this ontology. Figure 3 shows the hierarchy of this ontology. Appendix 1 summarizes some examples of the catalogs concepts and relations. For the sake of space, we illustrate our tendering ontologies by detailing two of the most important TCSs. Those are the Tendering Invitation Structure (TIS) and the sellers' profile structure (SPS).

Tendering Invitation Structure (TIS) is to inform the tenderers of the scope of the procurement. The tender invitation provides basic information on the procurement and guidance to the tenderers on the participation. We derived the component of TIS from UN EDIFACT ISO 9735 request for quote message (Blomberg and Lennartsson, 1997).
We define TIS as a nested CG, containing information about the scope of the procurement, the address and conditions of contracting entity, nature of contract, duration/completion of contract, eligibility, award criteria, rules on participation, and objects specifications.

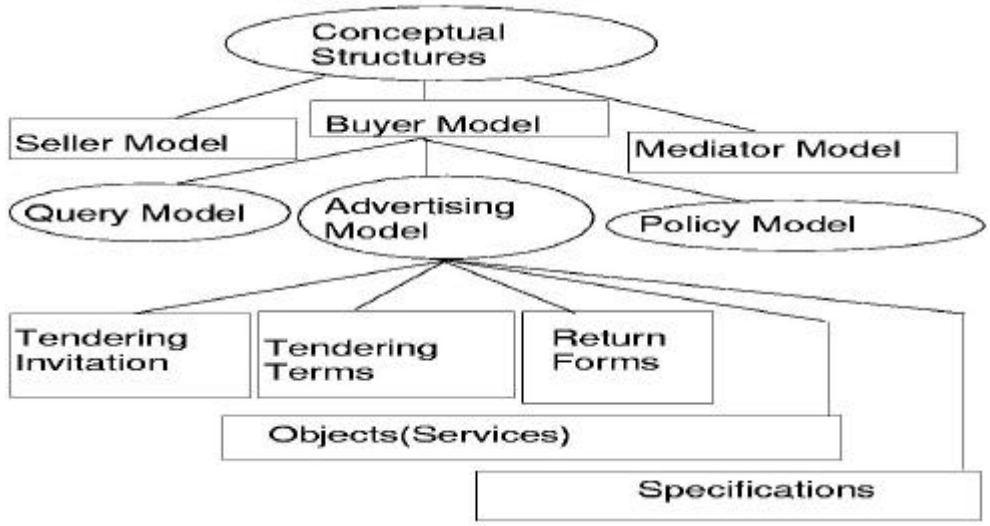

Figure 3: Tendering Conceptual Structures

Formally TIS is defined using the type function of CG as follows:

```
Type TIS(x) is
[TCS:*x] -
        (ATTR) ?  [Contracting Entity]-
                (ATTR)? [Address]
                (ATTR)? [Name]
        (ATTR)? [Contracting Duration]
        (ATTR)? [Eligibility]
        (ATTR)? [Award Criteria]
        (ATTR)? [Participation Rules]
        (ATTR)? [Services])-
```

(ATTR)? [Identification]
(ATTR)? [Description]
(ATTR)? [Quantity]
(Measure)? [Unit]
(Relevant Date)? [Production Date]
(Relevant Date)? [Expiry date]
... etc.

Sellers register with a certain mediator by sending their profiles. We define the SPS as a nested CG, containing information about sellers, their domain of interests, the services they offer, the value-added services such as extended warranties, brand reputation, fast delivery times, etc. Formally SPS is defined in CG as follows:

Type SPS(x) is a
[TCS:*x]-
(ATTR) ? [Contracting Entity]-
(ATTR) ? [Address]
(ATTR) ? [Name]
(ATTR) ? [Domain Interests]
(ATTR) ? [Services]
(ATTR) ? [Value Added Services: *{extended warranties, brand reputation, fast delivery times }]

In this example the concepts are not normalized i.e.(concept can be divided into more primitives concept like the address can be divided into country, street, etc.). The Tendering Conceptual Structure (TCS) is a primitive concept defined in the concept catalog. In this case the TCS is a super type of TIS (TIS < TCS). The type definition in C.G declares a new concept from primitive and predefined concepts and relations. Type definition contains two parts: the genus (the body of the definition) and differentiae (the label associated with that definition) (Sowa, 1984). Each concept in the body of a type definition should be defined in the concept catalog or in previous type definitions; also each relation should be defined in the same manner. To illustrate this, the contracting duration (CD) has been defined by the type definition as follows:

Type CD(x) is
[Time:*x]-
(Part-Of) ? [Tendering Date]
(Part-Of) ? [Start Date]
(Part-Of) ? [End Date]

Building ontology in this way help us to reuse existing ontologies. For example, we reuse the definition of time (Time, Date, Days, Years, Hours) form the ontology server (Farquhar et al., 1997). The one to one mapping between KIF and CG helps to reuse these definitions. Also we reuse the legal entity of (Uschold et al., 1998) as a genus concept and redefine it to cope with our application. We can use the context to remove and define the similarity between concepts in different ontologies, which is beyond the scope of this paper.

MOTIVATING EXAMPLE

E-mediator receives users' structures (buyers or sellers) and checks their semantics and correctness. To check that, E-mediator checks if the structures are canonically derived from the ontologies. Here we apply the algorithm of (Mugnier and Chein, 1993). This algorithm decides whether a conceptual graph is canonical relative to a given canonical basis. The complexity of this algorithm is polynomial related to the complexity of computing a projection between two conceptual graphs. When the canonical basis is a set of trees, it is polynomial. We use the editor proposed in (Pollitt et al., 1998)to edit the canonical basis graphs.
Buyer agent contacts the mediator through formal structures that are committed to the ontology which should be defined in previous step. Mediator checks the profile repository, and depending on the buyers' strategies determines the address of sellers that match their needs. When mediator receives the user's structure (e.g. TIS), mediator tries to find a projection between the ontological structure and the user structure. In the case of using type definition, one-to-one mapping between both structures is necessary. In the case of prototype mapping, each element in the user structure should be mapped to any element in mediator ontology.

Sowa ((Sowa, 1984), pp94) states that a graph w is said to be canonical derivable from a set of graphs A if one of the following is true:

- w is a member of A
- w may be derived by applying the canonical formation rule to graphs u and v that are themselves canonically derivable from A.

The canonical formation rules are copy, Restrict, Join, and simplify. To illustrate this, suppose Mr. K. wants to buy 600 PC. A seller wants to add his profile to the mediator repository. Mr. K. wants to advertise his need with a certain mediator. He consults mediator ontology; he browses the structure ontology and finds that TIS suits his needs. He fills the TIS as follows

[TCS: B1] -
(ATTR)? [Contracting Entity]-
(ATTR)? [Name: Mr. K.]
(ATTR)? [Contracting Duration]-
(Part-Of)? [Start Date:31-10-2000]
(Part-Of)? Lambda(T)[End Date:31-12-2000]
(ATTR)? [Services])-
(Part-Of) ? [Service1]-
(ATTR) ? [Identification: PC] ? (CTX) ? [I_4]
(ATTR) ? [Quantity:600]
(ATTR) ? [Description]-
(ATTR) ? [Warranty]? (> =)? [Year:@3]
(Do)? [installation]
(Do)? [Delivery]? (Before)? Delivery Time] ? (< =) ? [[T]+ [Month: @3]]
(ATTR)? [General Terms]? (Part-Of)? [Experience]? (>)? [Year:@3]

The seller S1 can submit his profile as follows

[TCS: S1]-
(ATTR) ? [Contracting Entity]-
(ATTR) ? [Name: Seller1]
(ATTR) ? [Domain Interests: {[Electronic], [Selling], [Installation]}]

(ATTR) ? [Services: {Computers, Printers }]
(ATTR) ? [Value Added Services: {[Extended warranties: @4 years], [Delivery: @[Day: @60]]}]

All terms in square brackets are defined in concepts catalog, previous type or prototype definition. We assume that these structures are semantically correct. After that e-mediator tries to match the TIS with the suitable profile. By applying the formulation rules, we can find out that the buyer TIS matches the seller profile. The matching here is in the sense of generalization and specialization. That means there is a projection from the buyers TIS onto the seller SPS. Think of e-mediator as a theorem prover who tries to prove the buyers' TIS with sellers' PSPs repository. Mediator may use some lifting axioms to prove any assertion in the TIS graph. As example, the assertion [Day: @60] < [Month: @3] can be easily proved by using the lifting axiom [Month: 1]? (=)? [Day: 30]. For more technical details about CG matching and indexing, readers encouraged to read Ellis work in (Ellis, 1995) (Ellis, 1993) (Ellis et al., 1994).
When buyers submit their TIS they should define contexts, which help the mediator, to find the appropriate sellers. In the CG sense the context is a situation where the graph is true. In our system we define the context as concepts measures labeled from I_1 to I_{12} (see table 1, some measures were adopted from (Hammer and McLeod, 1993)). We associate the context with a sub-graph via the CXT relationship. In the previous example, under I_4 (Kind-of) the buyer accepts any seller who sells computer since the [PC] is a kind-of [Computer]. In other situation (i.e. under I_1 (Identical)) this is not true.
We apply the context measures in graph level or in the concept level. In the concept level, the concept is true if and only if there is a relation in the ontology catalogs relates the two concepts. This relation must be equivalent or dominated the context relation (i.e. I_1 to I_{12}). In the graph level, the context is true if and only if the context is true for each concept.

Similarity Type	Meaning
(I_1) -Identical	Two concepts are the same
(I_2) -Equal	Two concepts are equivalent
(I_3) -Compatible	Two concepts are transformable
(I_4) -Kind-Of	Specialization of a concept
(I_5) -Association	Positive association between two concepts
(I_6) -Collection-Of	Collection of related concepts
(I_7) -Component-Object	Component part of a concept
(I_8) -Portion	A mass portion of a concept
(I_9) -Instance-Of	Instance of a concept
(I_{10})-Common	Common characteristics of a collection or concept
(I_{11})-Feature	Descriptive feature of a concept
(I_{12})-Has	Property belonging to instances or concepts

Table 1: Intention CG Contexts

CONCLUSIONS AND FUTURE WORK

We have defined the role of ontology in automating the tendering process. We have constructed our ontologies based on three components: the concepts, the structures, and the contexts. This decomposition facilitates the process of ontology building and reusing. We have described our system of tendering automation focusing on the role of ontology. We clarified how the ontology would help in defining semantic matching. We have shown how the expressive power of CG helps in building ontologies and conceptual structures.

We introduced the concepts of layered ontologies. At the top level we used very abstract ontology which contains abstract data types for the domain ontology. Defining variant levels of abstractions facilitates the transform of catalog to ontology. One of the exciting areas here is to define the relation between the catalogs, standards, and ontologies. Catalogs are not interoperable. Standard catalogs are but lack flexibility. The ontology is more flexible and provides interoperability between partners.

The lack of agreement of the definition of ontology in each domain, where each application organizes it to suit themselves, causes a problem in building a common ontology. Our approach in decomposing the ontology into abstract and domain ontology helps to solve this problem.
In future work, we will use the context to implement what we call soft-matching (Kayed and Colomb, 1999). Soft-matching depends on the multi-attribute utility theory (MAUT) (Hatush, 1996). The tender is divided into classes, which may contain sub-classes. The buyer provides a factor of importance (utility function) for each concept in each class in each level in the tender (the sum in each level should =100%). A context like I_1 is not totally true, it is true with the importance factor specified by the buyer. So the context will have the form [I_i, Percentage]. This fuzziness will capture the buyers' policies that will direct the agent in finding buyers' needs.

Since we are thinking of agent oriented design, the mediator represents a collection of software agents. Agency means that the agent can behave on behalf of the user, but this behavior is controlled by a given strategies or policies. In our future work, we will use the CG formalism to define how the agent could work autonomously, how the policies can be adaptable, whether the agent can change its own policy by learning, etc.

The CG literatures provide some tools to check if a CG is canonically derived from a canon basis or not (e.g. Mugnier et. al algorithm (Mugnier and Chein, 1993) and their CoGITaNT project (Genest and Salvat, 1998)). The reverse engineering of this process is not supported. At first glance, the idea of given CGs then create canon basis seems absurd. The idea of reverse engineering of CSs to canon basis makes sense in the ontology building process. If we think of ontology as enhanced step in knowledge representation, then the ontology will represent abstract knowledge that can be reused. In this case the reverse engineering of knowledge structure will be the basis for the ontology building process. An algorithm to build a minimal canon basis from given CGs is needed. We are building tools to reverse engineering of CSs to canon basis.

Sowa (Sowa, 2000) states that there are only five primitives needed to build any logical modeling languages (Existence, Co-reference, Relation, Conjunction, Negation). BWW model defines other five basic for any modeling technique (Things, Property, State, transformation, stable state).

For Property Guarino (Guarino, 1998) formulizes three ontological properties (identity, rigidity, and dependency). We are going to use BWW model to evaluate CG and compare these schools.

REFERENCES

Adam et al., (1998). *Electronic Commerce: Technical, Business, and Legal Issues*. Prentice Hall (ISBN: 0-13-949082-5.

Blomberg, P. and Lennartsson, S. (June 1997). *Technical assistance in Electronic Tendering Devlopment-FINAL REPORT Technical assistance in electronic procurement to EDI - EEG 12 Sub-group 1*. http://simaptest.infeurope.lu/EN/pub/src/main6.htm.

Bunge, M. (1977). *Treatise on Basic Philosophy: Volume 3: Ontology I: The Furniture of the World*. Dordrecht, Holland.

Campbell, A. and Shapiro, S. C. (1998). Algorithms for ontological mediation. In *In S. Harabagiu, Ed., Usage of WordNet in Natural Language Processing Systems: Proceedings of the Workshop, COLING-ACL, New Brunswick, NJ*, pages 102-107.

Elkan, C. and Greiner, R. (1993). Book review of building large knowledge-based systems: Representation and inference in the cyc project (D.B. Lenat and R.V. Guha). *Artificial Intelligence*, 61(1):41-52.

Ellis, G. (1993). Efficient retrieval from hierarchies of objects using lattice operations. *Lecture Notes in Computer Science*, 699:274-??

Ellis, G. (1995). *Ph.D. Thesis: Managing Complex Objects*. Computer Science Department, University of Queensland.

Ellis, G., Levinson, R. A., and Robinson, P. J. (1994). Managing complex objects in peirce. *International Journal of Human-Computer Studies*, 41(1,2):109-148.

Fadel, G., F., Fox, M., and Gruninger, M. (1994). A generic enterprise resource ontology. In *Proceedings of the Third Workshop on Enabling Technologies- Infrastructures for Collaborative Enterprises , West Virginia University*.

Farquhar, A., Fikes, R., and Rice, J. (1997). The ontolingua server: A tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707-727.

Fernandez, M., Gomez-Perez, A., and Juristo, N. (1996). *Methodology: From Ontological Art Towards Ontological Engineering*. Workshop on Ontological Engineering. ECAI'96 PP 41-51.

Fernandez, M., Gomez-Perez, A., and Sierra, J. (1999). Building a chemical ontology using methodology and the ontology design environment. *IEEE Intelligent Systems*, 14,1:37-46.

Genest, D. and Salvat, E. (1998). A platform allowing typed nested graphs: How CoGITo became CoGITaNT. *Lecture Notes in Computer Science*, 1453:154-??

Gomez-Perez, A. and Rojas-Amaya, D. (1999). Ontological reengineering for reuse. *Lecture Notes in Computer Science*, 1621:139-149

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199-220.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5,6):907-928.

Guarino, N. (1998). Formal ontology and information systems. In *Proc. of the 1st International Conference, Trento, Italy*.

Guarino, N. and Poli, R. (1995). Editorial: The role of formal ontology in the information technology. *International Journal of Human-Computer Studies*, 43(5,6):623-624.

Hammer, J. and McLeod, D. (1993). An approach to resolving semantic heterogeneity in a federation of autonomous, heterogeneous database systems. *Journal for Intelligent and Cooperative Information Systems*, 2(1):51-83.

Hatush, Z. (1996). *Ph.D. Thesis: Contractor selection using multi-attribute utility theory*. School of Construction Management and Property, Queensland University of Technology-Australia.

Hendler, J. and Stoffel, K. (1999). Back-end technology for high-perfoemance knowledge-representation systems. *IEEE Intelligent Systems*, 14,3:63-69.

Kashyap, V. and Sheth, A. P. (1996). Semantic and schematic similarities between database objects: A context-based approach. *The VLDB Journal*, 5(4):276-304.

Kayed, A. (June 2000a). *Business to Business Electronic Commerce: The Electronic Tendering*. Doctoral Consortium,2000 Pacific Asia Conference on Information Systems (PACIS2000), Hong Kong.

Kayed, A. (2000b). *Home Page*. http://www.csee.uq.edu.au/ kayed/.

Kayed, A. and Colomb, R. M. (1999). Infrastructure for electronic tendering interoperability. In *The Australian Workshop on AI in Electronic Commerce, conjunction with the Australian Joint Conference on Artificial Intelligence (AI'99) Sydney, Australia, ISBN 0643065520*, pages 87-102.

Lehmann, F. (1995). *Machine-Negotiated, Ontology-Based EDI In Lecture Notes in Computer Science: Electronic Commerce, Current Research Issues and Application by Nabil R. Adam and Yelena Yesha*. Springer 1028.

Martin, P. and Eklund, P. (1999). Embedding knowledge in Web documents: CGs versus XML-based metadata languages. *Lecture Notes in Computer Science*, 1640:230-240

Merriam-Webster, I. (1999). *WWWebster Dictionary*. http://www.m-w.com/dictionary.

Mineau, G. W. (1993). The term definition operators of ontolingua and of the conceptual graph formalism: a comparison. In Mineau, Guy W.; Moulin, Bernard; Sowa, J. F., editor, *Proceedings of the First International Conference on Conceptual Structures for Knowledge Representation (ICCS '93)*, volume 699 of *LNAI*, pages 90-105, Quebec City, Canada. Springer Verlag.

Mineau, G. W. (1999). Constraints on processes: Essential elements for the validation and execution of processes. *Lecture Notes in Computer Science*, 1640:66-??

Mugnier, M. L. and Chein, M. (1993). Characterization and algorithmic recognition of canonical conceptual graphs. *Lecture Notes in Computer Science*, 699:294-301.

Noy, N. and Hafner, C. (1997). The state of the art in ontology design. *AI Magazine*, Fall:53-74.

Ogden, C. and Richards, I. A. (1946). *The meaning of meaning*. Harcourt, Brace, and World, New York, NY, 1946.

Pollitt, S., Burrow, A., and Eklund, P. W. (1998). WebKB-GE - A visual editor for canonical conceptual graphs. *Lecture Notes in Computer Science*, 1453:111-118.

Shadolt, N., Cottam, H., and Milton, N. (1996). *Ontologies for Business Process Re-Engineering*. University of Nottingham, Artificial Intelligence Group, Department of Psychology, University Park, Nottingham NG7 2RD. United Kingdom.

Slone, A. (1992). Electronic data interchange and structured message usage. *Computer Standards and Interfaces*, 14,5-6:411-414.

Sowa, J. F. (1984). *Conceptual Structures: Information Processing in Minds and Machines*. Addison-Wesley, Reading, Mass.

Sowa, J. F. (1995). Syntax, semantics, and pragmatics of contexts. *Lecture Notes in Computer Science*, 954:1-??

Sowa, J. F. (1997). A Peircean foundations for the theory of context. In Lukose, D., Delugach, H., Keeler, M., Searle, L., and Sowa, J., editors, *Proceedings of the 5th International Conference on Conceptual Structures*, volume 1257 of *LNAI*, pages 41-64, Berlin. Springer.

Sowa, J. F. (1998). Conceptual graph standard and extensions. *Lecture Notes in Computer Science*, 1453:3-45

Sowa, J. F. (2000). Ontology, metadata, and Semiotics. *Lecture Notes in Artificial Intelligence*, 1867:56-83.

Tepfenhart, W. M. (1998). Ontologies and conceptual structures. *Lecture Notes in Computer Science*, 1453:334-344

Uschold, M. (1996). Building ontologies: Towards a unified methodology. In *16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems*, Cambridge, UK.

Uschold, M., King, M., Moralee, S., and Zorgios, Y. (1998). The enterprise ontology. *The Knowledge Engineering Review*, 13(Special Issue on Putting Ontologies to Use).

Wand, Y. and Weber, R. (1990). An ontological model of an information system. *IEEE Transactions on Software Engineering (SE), ; ACM CR 9107-0564*, 16(11).

Way, E. C. (1994). Conceptual graphs: Past, present, and future. In Tepfenhart, W. M., Dick, J. P., and Sowa, J. F., editors, *Proceedings of the 2nd International Conference on Conceptual Structures : Current Practices*, volume 835 of *LNAI*, pages 11-30, Berlin. Springer.

Weber, R. (1997). *Ontological Foundations of Information Systems*. Cooper and Lybrand, Accounting Research Methodology, Monograph No. 4.

ACKNOWLEDGMENT

The authors acknowledge the department of CSEE at the University of Queensland for financial support for this project. Also we acknowledge anonymous referees who contributed to improving the ideas and readability of this paper.

APPENDIX 1

Entity, Contracting Entity, Buyers, Sellers, Process, Activity, Name, Address, Street, Service, Items, Contracting entity Classification, Classical, Government entity, Name, Address, Organisation, Organisation code, Postal code, Place, City, Country, Official registration number, Registration number, VAT number, Contact information, Contact person, Telephone, Telefax, Electronic mail, Nature of contract, Purchase, Rent, Lease, Hire, Duration, completion of contract, Contract date, Contract start date, Contract end date, Number of months, Weeks, Days, Eligibility, Lowest price, Most economically advantageous, tender Rules, Final date for receipt of tenders, etc.

COPYRIGHT