Issues in Ontology-based Information Integration

Zhan Cui, Dean Jones and Paul O'Brien
Intelligent Business Systems Research Group
Intelligent Systems Lab
BTexact Technology

Abstract

Solving queries to support e-commerce transactions can involve retrieving and integrating information from multiple information resources. Often, users don't care which resources are used to answer their query. In such situations, the ideal solution would be to hide from the user the details of the resources involved in solving a particular query. An example would be providing seamless access to a set of heterogeneous electronic product catalogues. There are many problems that must be addressed before such a solution can be provided. In this paper, we discuss a number of these problems, indicate how we have addressed these and go on to describe the proof-of-concept demonstration system we have developed.

1. Introduction

There are a number of obstacles to completely open e-commerce over the Internet. One of the major problems is the vast amount of information that is available and our ability to make sense of it. For example, how do we identify whom to do business with? How do we know that a supplier's products are what we are looking for? It is only once we know what people are saying that we can start to identify who is worth talking to. In this article, we will discuss a number of related issues and describe the way we have begun to address some of them.

The problems of interoperability between interacting computer systems have been well documented. A good classification of the different kinds of interoperability problems can be found in [Sheth, 98] who identifies the system, syntactic, structural and semantic levels of heterogeneity. The system level includes incompatible hardware and operating systems; the syntactic level refers to different languages and data representations; the structural level includes different data models and the semantic level refers to the meaning of terms using in the interchange. A good example of semantic heterogeneity is the use of synonyms, where different terms are used to refer to the same concept. There are many more types of semantic heterogeneity and they have been classified in [Visser *et al.*, 1998]

Many technologies have been developed to tackle these types of heterogeneity. The first three categories have been addressed using technologies such as CORBA, DCOM and various middleware products. Recently XML has gained acceptance as a way of providing a common syntax for exchanging heterogeneous information. A number of schema-level specifications (usually as a Document Type Definition or an XML Schema) have recently been proposed as standards for use in e-commerce, including ebXML, BizTalk and RosettaNet. Although such schema-level specifications can successfully be used to specify an agreed set of labels with which to exchange product information, it is wrong to assume that these solutions also solve the problems of semantic heterogeneity. Firstly, there are many such schema-level specifications and it cannot be assumed that they will all be based on consistent use of terminology. Secondly, it does not ensure consistent use of terminology in the data contained in different files that use the same set of labels. The problem of semantic heterogeneity will still exist in a world where all data is exchanged using XML structured according to standard schema-level specifications.

A solution to the problems of semantic heterogeneity should equip heterogeneous and autonomous software systems with the ability to share and exchange information in a semantically consistent way. This can of course be achieved in many ways, each of which might be the most appropriate given some set of circumstances. One solution is for developers to write code which translates between the terminologies of pairs of systems. Where the requirement is for a small number of systems to interoperate, this may be a useful solution. However, this solution does not scale as the development costs increase as more systems are added and the degree of semantic heterogeneity increases.

Our solution to the problem of semantic heterogeneity is to formally specify the meaning of the terminology of each system and to define a translation between each system terminologies and an intermediate terminology. We specify the system and intermediate terminologies using *formal* *ontologies* and we specify the translation between them using *ontology mappings*. A formal ontology consists of definitions of terms. It usually includes concepts with associated attributes, relationships and constraints defined between the concepts and entities that are instances of concepts.

We provide software support for the definition and validation of formal ontologies and ontology mappings, allowing us to resolve semantic mismatches between terminologies according to the current context (*e.g.* such as the application.) In the next section we discuss a number of issues relating to the use of ontologies in enabling semantic interoperability. We then describe how we have addressed some of these issues using a system called DOME (Domain Ontology Management Environment) which includes a set of tools for creating and mapping between ontologies, for browsing and customising ontologies and for constructing concept-based queries.

2. Issues in Resolving Semantic Heterogeneity

In this section we describe some of the problems involved in achieving semantic interoperability between heterogeneous systems.

2.1 Developing ontologies

In any reasonably realistic e-commerce scenario involving interoperability between systems, semantic heterogeneity is a significant problem and will continue to be so in the future. A solution to this problem based on the use of formal ontologies will need to accommodate different types of ontologies for different purposes. For example, we may have *resource ontologies*, which define the terminology used by specific information resources. We may also have *personal ontologies*, which define the terminology of a user or some group of users. Another type is *shared ontologies*, which are used as the common terminology between a number of different systems.

The problem of developing ontologies has been well-studied and a number of methodologies have been proposed. A comparative analysis of these can be found in [Jones *et al.*, 1998].) One of the major conclusions of this study was that the best approach to take in developing an ontology is usually determined by the eventual purpose of the ontology. For example, if we wish to specify a resource ontology, it is probably best to adopt a bottom-up approach, defining the actual terms used by the resource and then generalising from these. However, in developing a shared ontology it will be extremely difficult to adopt a bottom-up approach starting with each system, especially where there are a large number of such systems. Here, it is most effective to adopt a top-down approach, defining the most general concepts in the domain first.

2.2 Mapping Between Ontologies

In order to resolve the problems of semantic mismatches discussed above, we will often need to translate between different terminologies. While it would be ideal to be able to automatically infer the mappings required to perform such translations, this is not always possible. While the formal definitions in an ontology are the best specification of the meaning of terms that we currently have available, they cannot capture the full meaning. Therefore, there must be some human intervention in the process of identifying correspondences between different ontologies. Although machines are unlikely to derive mappings, it is possible for them to make useful suggestions for possible correspondences and to validate human-specified correspondences.

Creating mappings is a major engineering work where re-use is desirable. Declaratively-specifying mappings allows the ontology engineer to modify and reuse mappings. Such mappings require a mediator system that is capable of interpreting them in order to translate between different ontologies. It would also be useful to include a library of mappings and conversion functions as there are many standard transformations which could be reused *e.g.* converting kilos to pounds, etc.

Mapping between ontologies is not an exact science. Certain semantic mismatches cannot be resolved exactly but may involve some loss of information *e.g.* when translating from a colour system based on RGB values to one which uses terms such as 'red', 'blue', etc. Whether or not the loss of information is an issue varies between applications. In some domains, precision of information is more important than in others. For example, in e-commerce, imperfect information is generally unacceptable, whereas it is widely accepted that internet search engines will return many irrelevant results.

2.3 Ontologies and Resource Information

It is generally acknowledged that we have more information than we know what to do with. This proliferation of data means that often, for any information query we might have, there are a variety of resources available that store data about the same domain and which are of varying quality. A distributed query engine needs to decide which of the many available resources to use in finding the solution to a query. In addition to finding the resources that have the required information, it may also be necessary to decide between different resources that have the same information available. In order for a distributed query engine to understand what information is available, the resources need to make descriptions of their contents available in a meaningful way. If the terms using in such a description are formally defined in an ontology, the query engine has access to the meaning of the terms in the description. This allows the query engine to make fully informed decisions about which resources are relevant to resolving a particular query.

There are a number of pragmatic issues in locating the resources that will be used to answer a query. For example, a particular user may - for whatever reason - prefer one resource over another as the source of some information.

Such personal preferences can be taken into account by the distributed query engine if a personal profile of a user's preferences is maintained. The query engine can make better informed decisions if the definitions of the terms used in such a profile are available to it in the form of a user *ontology*, which defines the terminology of a user or user-group.

2.4 Ontologies and Database Schemas

Ontologies and database schemas are closely related and people often have trouble deciding which is which. There is often no tangible difference, no way of identifying which representation is a schema and which is an ontology. This is especially true for schemas represented using a semantic data model. The main difference is one of purpose. An ontology is developed in order to define the meaning of the terms used in some domain whereas a schema is developed in order to model some data. Although there is often some correspondence between a data model and the meaning of the terms used, this is not necessarily the case. Both schemas and ontologies play key roles in heterogeneous information integration because both semantics and data structures are important.

For example, the terminology used in schemas is often not the best way to describe the content of a resource to people or machines. If we use the terms defined in a resource ontology to describe the contents of a resource, queries that are sent to the resource will also use these terms. In order to answer such queries, there needs to be a relationship defined between the ontology and the resource schema. Again, declarative mappings that can be interpreted by some mediator system are useful here. The structural information provided by schemas will enable the construction of executable queries such as SQL queries.

This is related to the discussion earlier about XML, where a database schema is analogous to an XML schema or DTD. As pointed out above, using XML is insufficient for determining the semantics of resources. A schema, whether specified using XML or some database schema language, needs an associated formal ontology in order to make the semantics of the resource clear. When the meaning of data and schemas is made explicit using an ontology, programs can be designed that exploit those semantics.

2.5 Entity Correspondence

Ontologies are used in e-commerce environments where data is scattered across heterogeneous distributed systems. In order for the consumer to have access to the maximum amount of available information, we want to be able to retrieve information from various systems and to integrate it. For example, we might want to integrate information from a supplier's product catalogue with customer reviews produced independently.

To gather all the information relevant to an entities, the correspondence between entities across resources must be established. For example, the academic records and criminal records of a person are likely to be stored in separated data resources. However, the way in which different resources identify individuals varies. For example, in relational databases entities are identified using key attributes. There is no guarantee that different relational databases use the same key attributes. Even when the same key attribute is used, different terms may be used to denote the attributes. How our systems can determine whether entities from different resources are the same or not is crucial to fusing information. Standard schemas do not provide a full solution here since many systems (*e.g.* KBSs, object-oriented databases) often do not have key attributes at all.

3. DOME Overview

The DOME project has been researching and developing ontology-based techniques to support the building of a "one-stop knowledge shop" for corporate information. We have developed a methodology, a set of tools and an architecture to enable enterprise-wide information management for data re-use and knowledge sharing. The system retrieves information from multiple resources to answer user queries and presents the results in a consistent way that is meaningful to the user. This section gives an overview of the DOME prototype system and some implementation details. Further details of DOME can be found in [Cui *et al.*, 2001]. Figure 1 shows the architecture of the DOME prototype.

The DOME prototype consists of a number of interacting components: an ontology server which is responsible for managing the definitions of terms, a mapping server which manages the relationships between ontologies, an engineering client with tools for developing and administrating a DOME system, a user client to support querying the knowledge shop, and a query engine for decomposing queries fusing the results to sub-queries. The prototype is implemented as an Enterprise JavaBean which provides two APIs - one for developers and one for users and applications.

3.1 Engineering client

A developer who wishes to set up a DOME system interacts with an engineering client which provides support in the development of the knowledge shop. This includes tools for the semi-automated extraction of ontologies from legacy systems [Yang *et al.*, 1999], for defining ontologies, for defining mappings between ontologies and between resource ontologies and database schemas.

We have developed a methodology that combines top-down and bottom-up ontology development approaches. This allows the engineer to select the best approach to take in developing an ontology. The top-down process starts with domain analysis to identify key concepts by consulting corporate data standards, information models, or generic ontologies such as Cyc or WordNet. Following that, the engineer defines competency questions [Gruninger and Fox, 1995.] The top down process results in the shared ontologies

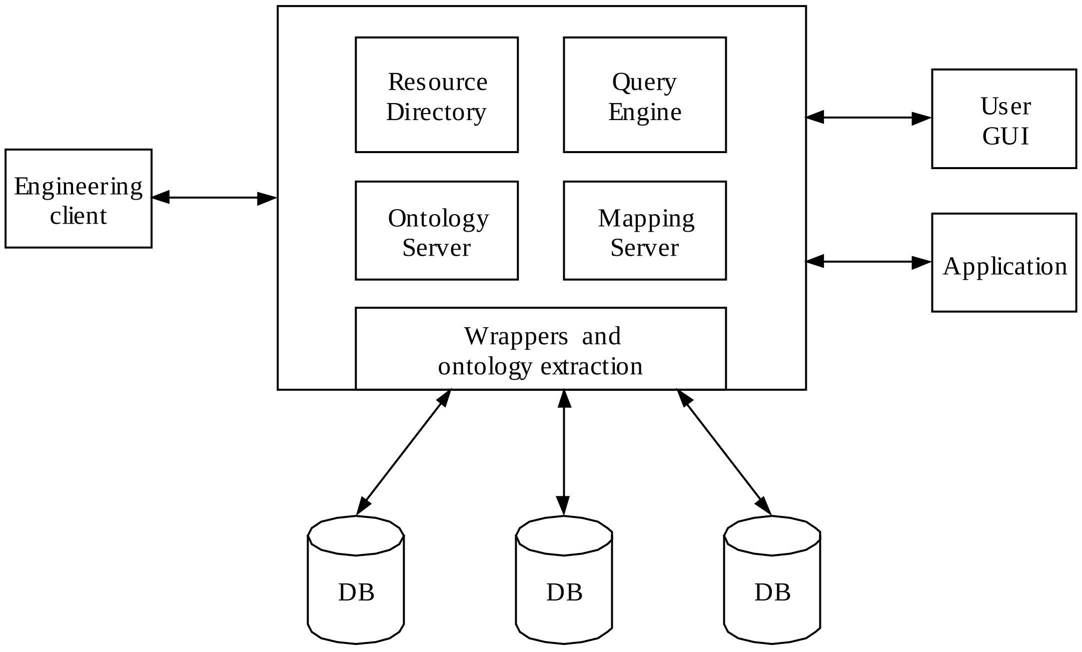

Figure 1: The DOME architecture

mentioned above. The bottom-up process starts with the underlying data sources. The extraction tool is applied to database schemas and application programs to produce initial ontologies which are further refined to become resource ontologies. We also provide for the development of *application ontologies*, which define the terminology of a user-group or client application. Application ontologies are defined by specialising the definitions in a shared ontology. Once the ontologies have been defined, they are stored in the ontology server.

The engineer also needs to define mappings between the resource ontologies and the shared ontology for a particular application. The rest of the ontology engineering task is to define mappings between the resource and shared ontologies using ontology mappings. Although we do not infer the mappings automatically, we can utilise ontologies to check the mappings for consistency. The engineer also needs to define mappings between the database schemas and the resource ontologies.

3.2 Ontology server

The ontology server stores the ontologies that are defined using the engineering client and allows access to the three kinds of ontologies in a DOME network: *shared*, *resource* and *application* ontologies. Shared ontologies contain definitions of general terms that are common across and between enterprises. A resource ontology contains definitions of terms used by a particular resource. These ontologies are stored in the DOME ontology server which implements ontologies using the description logic CLASSIC [Brachman *et al.*, 1992]. CLASSIC is used to both store ontologies and to make inferences. Access to the ontology server is through Open Knowledge Base Connectivity (OKBC) interface, which is a *de facto* standard for accessing knowledge bases [Chaudhri *et al.*, 1998].

3.3 User client

We provide users with tools to access the knowledge shop. We have defined a simple API that allows a user client or an application to querying the distributed information space. The user client also provides facilities for loading and browsing specific ontologies in the knowledge shop to view what is available in the whole information space. Queries are passed to DOME as strings which conform to an XML schema which defines the syntax of the DOME query language. This is similar to SQL but doesn't require that we specify on which attributes to make joins between concepts since this will be identified automatically by the query engine. Queries are formed using the terminology defined in an application ontology and the results which are returned are represented using the same terminology, hence hiding the details of the different systems, their distribution, structure, syntax or semantics from the user.

3.4 Mapping Server

The mapping server stores the mappings between ontologies which are defined by the engineer in setting up a DOME network. The mapping server also stores generic conversion functions which can be utilised by the engineer when defining a mapping from one ontology to another. These mappings are specified using a declarative syntax, which allows the mappings to be straightforwardly modified and reused. The query engine queries the mapping server when it needs to translate between ontologies in solving a query.

3.5 Wrappers

Most interaction between a resource and the DOME network occurs via wrappers. A wrapper performs translations of queries expressed in the DOME query syntax and terminology of the resource ontology to queries expressed in the syntax of the resource query language and the terminology of the resource schema. They also perform any translations required to put the results into the terminology of the resource ontology. Although they are configured for particular resources, DOME wrappers are generic across resources of the same type *e.g.* wrappers of SQL databases utilise the same code.

3.6 Resource Directory

When a resource is connected to a DOME network, its wrapper will inform the DOME directory about its existence and pass to the resource directory a description of the contents of the resource, expressed in terms of the relevant resource ontology. This ensures that the query engine is able to identify what information is available without having to access the schema of the resource. When a wrapper is - for whatever reason - no longer able to provide information from a resource, it will inform the resource directory which is then able to discount that resource from any future query solving.

3.7 Query engine

Upon receiving a query, the DOME query engine first needs to decide which resources are relevant to that query. It obtains a list of currently available and relevant resources by consulting the directory. Based on this information, the query engine decomposes the query into sub-queries. The query engine ensures that the decomposition is performed in such a way that the results to the sub-queries, once they are received from the resources, can then be integrated. It then translates queries from the ontology of the query to that of the relevant resource and will send the sub-queries to the resources. Once the results are received, the query engine will integrate the results.

4. DOME Demonstrator

We have developed a demonstrator for the DOME prototype based on a database marketing scenario. Database marketing involves targeting marketing information from customer information stored in databases. Typically, queries are *ad hoc*, that is, it is difficult to pre-define a set of typical queries. Also, customer information is necessarily split across many different databases *e.g.* a customer may have multiple products, the records for which are stored in different databases. This requires that queries often need to join information from a wide variety of databases. As the databases we used are developed independently and serve different applications, DOME has to search for resources which hold data about customers that it is possible to integrate. The resources that are used varies from query to query. The databases also have different levels of data quality - there are incorrect entries, missing records, etc. As DOME allows mappings to be specified between the shared and resource ontologies, we have some control over which resources are utilised for data that is available from multiple databases. By only defining mappings between the shared ontology and the parts of the resource ontology for which the resource is a trusted sources of information, we can limit the parts of a resource that is used to solve queries.

5. Conclusions

Semantic interoperation is one of the main obstacles to free and full electronic commerce. Understanding what is available is a necessary prerequisite to a successful business transaction. We have described a number of issues involved in supporting the interoperation of computer systems at the semantic level. We have also described the architecture of the DOME system that we have developed to illustrate our approach to overcoming some of these problems. We believe that the proof-of-concept demonstrator we have developed supports the utility of ontologies in integrating heterogeneous information resources for applications such as e-commerce. DOME provides functionality to (i) support a system engineer in providing an integrated view of networked heterogeneous databases, (ii) allow a user to select and browse definitions of terminologies and to pose queries in their chosen vocabulary and (iii) answer user queries based on the information available. This work is ongoing and there are a number of areas currently being explored. For example, an increasing number of resources that use some form of XML technology are becoming available and we are currently developing components that will allow data retrieved from such resources to be integrated with data retrieved from other kinds of resources such as relational databases. We believe strongly that even in a world where there are many such resources, there will still be a role for formal ontologies in enabling semantic interoperability.

References

[Brachman *et al.*, 1992] R.J. Brachman, A. Borgida, D.L. McGuinness, P.F. Patel-Schneider and L. Alperin Resnick. The CLASSIC Knowledge Representation System, or KL-ONE: The Next Generation. *Proceedings of the 1992 International Conference on Fifth Generation Computer Systems*, Tokyo, Japan, June 1992.

[Chaudhri *et al.*, 1998] V.K. Chaudhri, A. Farquhar, R. Fikes, P.D. Karp, and J.P. Rice. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. *Proceedings of AAAI-98*, pages 600-607, Madison, WI, 1998

[Cui *et al*., 2001] Z. Cui, M.D.J. Cox and D.M. Jones. An Environment for Managing Enterprise Domain Ontologies. M. Rossi and K. Siau (eds.) *Information Modelling in the New Millennium*, Idea Group Publishing, London.

[Gruninger and Fox, 1995] M. Gruninger, and M.S. Fox. Methodology for the Design and Evaluation of Ontologies. *IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, 1995

[Jones *et al.*, 1998] D.M. Jones, T.J.M. Bench-Capon and P.R.S. Visser. Methodologies for Ontology Development. *Proceedings IT&KNOWS Conference of the 15th IFIP World Computer Congress*, Budapest, Chapman-Hall.

[Sheth, 1998] A.P. Sheth. Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. M. F. Goodchild, M. J. Egenhofer, R. Fegeas, and C. A. Kottman (eds.) *Interoperating Geographic Information Systems*, Kluwer.

[Visser *et al.*, 1998] P.R.S. Visser, D.M. Jones, T.J.M. Bench-Capon and M.J.R. Shave. Assessing Heterogeneity by Classifying Ontology Mismatches. *Proceedings International Conference on Formal Ontology in Information Systems - FOIS'98*, IOS Press.

[Yang *et al.*, 1999] H. Yang, Z. Cui and P.D. O'Brien. Extracting Ontologies from Legacy Systems for Understanding and Re-engineering. *Proceedings of 23rd IEEE International Conference on Computer Software and Applications*, Pheonix, AZ, October 1999.