Domain Ontology Management Environment

Zhan Cui and Paul O’Brien
Intelligent Business Systems Research
BT Advanced Communications Research Center
Adastral Park, IPSWICH IP5 3RE, UK
zhan.cui@bt.com, paul.d.obrien@bt.com

Abstract

Service-oriented business-to-business e-commerce requires dynamic and open-interoperable information systems. Although most large organizations have stored in databases information such as products, services and customers, and XML/DTD allows these to be published over the Internet, sharing information among these systems have been prevented by semantic heterogeneity. True electronic commerce will not come until the semantics of the terms used to model these information systems could be captured and processed by computers.

To develop a machine process-able ontology (vocabulary) is intrinsically hard. The semantics of a term varies from one context to another. We believe ontology engineering will be a major effort of any future application development. In this paper we describe our work on building a Domain Ontology Management Environment (DOME). DOME is developing techniques for ontology-based information contents description and a suite of tools for domain ontology management. Information contents description is extending traditional metadata to an ontology. This makes it possible to dynamically find relevant data sources based on contents and to integrate them as need arises.

1. Introduction

Service-oriented business-to-business e-commerce requires dynamic and open-interoperable information systems. Business services in e-commerce are often made of sub-services and tasks that normally belong to autonomous participants. Inevitably the underlying information systems to support e-business are distributed and autonomous. The success of e-commerce requires an infrastructure to support effective information exchange.

Today’s Internet has provided an effective, dynamic and open infrastructure to publish and to access vast quantity of text and data. Information can be easily published by anyone, browsed from anywhere and at anytime, and downloaded to support decision-makings. The recent initiative of W3C on XML/RDF Schema [1] is set to develop a unified mechanism to publish not only free texts and ad hoc form data, but also structured data and information on the Internet. This will probably form the basis for future on-line electronic commerce just as databases do for computerised applications. Eventually technologies such as XML/Schemas will allow companies to easily publish their existing databases over the Internet. In spite of this the effective use of distributed information has been seriously hindered by semantic heterogeneity. For example, people from different places often use different terms to refer to the same or similar products. Worst still the same term may be used to refer to different products. The success of e-commerce depends on effective collaborative information systems that help with access to information, support decision making and aid in task execution [2]. True electronic commerce will not come until the semantics of the terms used in these information systems can be captured and processed by computers.

The semantics of diverse information sources are captured by their ontologies, i.e., the terms and relationships between them. In tightly coupled applications, the intended meaning of a term is often implicit, thus relying on developer’s mutual agreement. In a distributed environment mutual agreement is hard to come by if not impossible. Thus it is crucial for the domain model and the vocabulary to be represented in such a way that enables programs to reuse them as they were originally intended with minimum human intervention during their execution. That is the task undertaken by ontology research that has now attracted the attention from both academia and industry.

To develop a machine process-able ontology (vocabulary) is intrinsically hard. The semantics of a term varies from one context to another. Ideally we need an approach that reduces the problem of knowing the contents and structure of many information resources to the problem of knowing the contents of easily understood, domain specific ontologies, which a user familiar with the
domain is likely to know or understand easily. We believe ontology engineering will be a major effort of any future application development. Support environments need to be built to assist ontology designers with defining, updating, mapping ontologies and their reuses. We envisage that the environment maintains an ontology repository that can be accessed by ontology designers and application programs. During the design phase, tools are available to explore and reuse the terms from the repository. When new terms need to be added, the environment will check they are consistent with what in the repository. During the run time, application programs can access the repository through a set of APIs to query the structures of the entities and their relationships with other terms or to convert one term to other. The environment should also have a set of tools to extract ontology from legacy systems because large amount of company's information is locked in their legacy databases. As companies move to electronic marketplace, these data have to be brought to support electronic commerce because they have embodied essential company information, represented large investment and become quite reliable over the years.

This paper describes our work of building such an environment called DOME - Domain Ontology Management Environment. The rest of the paper is structured as follows. Section 2 discusses different types of semantic heterogeneity that are relevant in the e-commerce context. Then we introduce the concept of ontology and how DOME views ontology. Section 4 describes the DOME ontology management methodology. Section 5 discusses how ontology developed in DOME can be used as information contents description. The related work is discussed in section 6. Finally section 7 concludes the paper.

2. Semantic heterogeneity

Heterogeneity in e-commerce is inevitable because the concerned systems are often developed by autonomous participants. However, heterogeneity is not new. It has been studied in the database integration context for many years. Amit Sheth [3] has discussed many of the issues of interoperability, and the focuses of the past decades. He classified heterogeneity into four categories: system, syntax, structure and semantic. The system heterogeneity includes hardware and operating systems; the syntax heterogeneity includes different languages and data representations; the structure heterogeneity includes different data models; and semantic heterogeneity includes the semantics of user's information request and those of information sources.

Visser, et al [4] considered heterogeneity from the ontology perspective. They classified the mismatches into two categories: conceptualisation and explication. The conceptualisation mismatches include different classifications, aggregations and attribute assignments and value types. Explication mismatches concern with the naming of concepts and definition form variations.

Many of the heterogeneity will overcome if XML/DTD is going to be adopted for data publishing. However the semantic heterogeneity remains. Semantic heterogeneity has been used to refer to the heterogeneity of both the mismatch of conceptualisation of reality and the mismatch of the modeled realities. Although the perception and conceptualisation of the reality is different from one to another, the real-world is composed of things having properties that are inherent and exist objectively whether or not they are observed or recorded [5]. Thus the conceptualisation of the reality has to be comprehended by reference to the real-world.

Wittgenstein [6] stated:

*Only the sentence has meaning; a name has meaning only in the context of a sentence.*

*A name means an object. The object is its meaning.*

It seems that any conceptualisation should have a set of primitive concepts whose meaning can only be understood only by reference to the real world objects. Therefore we assume the different conceptualisations of the real world can only be addressed and mapped by humans. The semantic mismatches of the modeled realities are caused by the choice of terms and a fixed set of attributes. In the modeling process, we have committed to model particular properties of the entities. The main difference is that the former is independent of languages while the latter is dependent of languages. According to this distinction, we classify the semantic heterogeneity as follows.

**Semantically equivalent concepts**

- Different terms are used to refer the same concept by two models. These terms are often called synonyms. However, synonyms in their common usage do not necessarily denote semantically equivalent concepts.
- Different properties are modeled by two systems. For example, for the same product, one catalogue has included its colour but the other has not. This heterogeneity is not a bad thing.
- Property type mismatches. For example, the concept length may be in meter or mile.

**Semantically unrelated concepts**

- Conflicting term - the same term may be chosen by two systems to denote completely different concepts. For example, apple is used to denote fruit or computer
Semantically related concepts

- Generalisation and specification. One system has only the concept of fruit, but the other has the concepts of apple, orange, etc. Another example is that student in one system refers to all students, but the other only to PhD students.
- Definable terms or abstraction - A term may be missing from one ontology, but which can be defined in other terms in the ontology.
- Overlapping concepts. For example, kids in one ontology means persons aged between 5 to 12 years old, but in the other means persons aged between 3 and 10 years old, and in yet another ontology, young persons means persons aged between 10 and 30 years old.
- Different conceptualisation. For example, one ontology classifies person as male, female. The other person as employed, unemployed.

3. DOME ontology

Semantic heterogeneity could be resolved by mutual agreements, manual reconciliation and standards. In a tightly coupled application, this can be achieved. However, this is difficult to achieve if not impossible in an open electronic commerce environment because standards need to accommodate the requirements of many different tasks that often require different properties of the entities to be modeled. Furthermore, applications may have slightly different views or constrain certain properties due to efficiency considerations, local customs, and so on. For example, there is no need to store the area code or country code if the telephone numbers in a database are all within the same area. Abbreviations are often more appropriate for communications within a particular user community.

 Ontology has been seen as a way to resolve semantic heterogeneity by specifying explicitly the semantics of the terms used in information systems. Ironically, there is no agreed definition of what ontology is in computer science. The most frequently quoted ontology definition is the one given by Gruber [7]. He states that

An ontology is an explicit specification of a conceptualisation.

The problem with this definition lies in what a conceptualisation is. Guarino [8] has extensively discussed the vagueness of this term and proposed his definitions. According to Guarino [9],

An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e., its ontological commitment to a particular conceptualisation of the world.

Logical theory is a formal system designed for inferences. Its meaning depends on the assignments of its non-logical symbols (terms) to objects in a domain of discourse. Although we could use the possible world semantics to capture the intended meanings, there is no efficient implementation to manipulate a logical theory. Furthermore, most of the applications deal with only the physical world we live in. Thus we should exploit our basic understanding of the actual world and use it as the basis to underpin the meanings of terms in our modeled reality.

In DOME, we assume that domain conceptualisation has been agreed among the user community. They have shared understanding of the reality or the domain. They may have used different terms to model the reality. Some may model some aspects of the reality, others may have modeled other aspects. Aspects of the reality include attributes of domain entities, and relationships and constraints between domain entities. Some aspects are primitives that must be mapped by human users as discussed in the previous section. We expect that this mapping is straightforward because of our shared understanding of the domain. Some support tools may be built to assist this mapping task using some heuristics.

Ontology in DOME consists of terms denoting concepts, their relationships and constraints. A concept is characterised by attributes that are necessary conditions which all objects of the concept must satisfy. This definition is intentionally similar to relational database schema definitions as well as object-oriented database classes because DOME is focusing on constructing ontologies from structured data sources. The representation language of DOME can be found in the Appendix.

We distinguish two classes of ontology terms: primitive terms and defined terms. The semantics of primitives are not specified. When ontology mapping is required, primitives have to be mapped manually. For this purpose we have used a mapping database and attribute semantic clusters to support this task. Once primitives have been mapped, defined terms can be eventually mapped automatically through a rewrite-engine using the mapping database.

Ontology in DOME forms a specialisation hierarchy with lower level terms having close links with ER models and database schemas. Figure 1 shows the relationships of ontology, ER models and database schemas. Ontology is seen as domain oriented concepts. It includes abstract concepts and specifies domain-level constraints that can be used for knowledge-level reasoning. Ontology is suited to represent high-level information requirements. Schemas and classes are data-level concepts that are implementation dependent. They are designed to optimise procedural operations. Constraints at this level are operational constraints. Many domain constraints are not
explicitly represented at this level. Terms of ontology are used to define database schemas. One ontology can be used to define different schemas.

Figure 1: The relationship between ontology and the underlying data sources

In addition, DOME ontologies form a hierarchy as shown in Figure 2. This is similar to the ontology clustering ideas discussed in Visser and Cui [10] and Visser and Tamma [11]. The parent ontology is inherited by child ontologies. Child ontologies understand concepts defined in their parent ontology even though some concepts may have been modified. Parent ontology is the minimum shared understanding of its child ontologies. We expect that every resource or application will have an ontology. Similar resources or applications will have similar ontologies. Thus they share a common parent ontology. Ontologies closer in the hierarchy will share more knowledge than distant ontologies. The mappings between closer ontologies are expected to be straightforward and simple. The ontology hierarchy should be similar to the classification of domains and subdomains.

Figure 2: Ontology hierarchy

4. DOME ontology management

In the previous section, we discussed the DOME’s view of ontology. In this section we describe the DOME ontology management system. DOME is a project to build an ontology management environment, focusing on deriving ontologies from legacy information systems.

DOME recognises that the central problem in the construction of e-commerce services is the lack of methods and tools to support the integration of process models and information systems from multiple organisations into shared virtual enterprise processes. The goal of DOME project is to provide the languages, software tools, and knowledge bases to enable correct, first-time designs of agile enterprise processes that have predictable performance and are rapidly realisable.

4.1 DOME architecture

DOME is to provide an environment where ontology developers can use the tools for reverse engineering data sources to an ontology and users or software agents can use the ontologies developed to integrate the information support systems dynamically. Figure 3 shows the main components of DOME architecture. These are:

- **Data sources**: these are legacy information systems for which ontologies need to be developed. Currently DOME deals only with structured databases and their application programs. However, this can be extended to include semi-structured data sources such as webpages represented as XML/DTD [12].

- **Ontology designers**: ontology designers are domain experts who use the tools available to construct ontologies for their data sources.

Figure 3: DOME ontology management system components
XRA: it is an ontology extraction tool that uses the software reverse engineering approach to extract an initial ontology from given data sources and their application programs. XRA [13, 14] can take relational database schemas or object-oriented models and application programs to extract entities, relationships between entities and functions. This initial ontology will be refined by ontology designers. As database schemas are developed with respect to machine-level operations, many semantic relationships can only be discovered from application programs. The enriched entity relationships through concepts in application programs reflect the correct usage of the data sources that are often not documented.

General ontology: this is the ontology of the domain that is the basis of the co-operating agents. It is often derived from company data standards.

Ontology reasoner: ontology in DOME is implemented in the description logic implementation CLASSIC [19] which is used to check ontology consistency and to merge ontologies.

Mapping database: it records the mapping information between ontologies, and ontology and data sources. These mappings are primitive concept mappings. For derived concepts we are developing a rewrite engine on top of CLASSIC.

The interface that allows the users to query ontology repository is based on OKBC [15]. OKBC specifies a knowledge model and a set of operations based on this model (e.g., find a frame matching a name, enumerate the slots of a frame, delete a frame). OKBC is developed in a DARPA project and has been adopted by FIPA [16]. Mediators are intended to provide value added services using DOME ontologies. This has not yet been developed by DOME. However, their functions will be similar to the mediators of Kraft system [17].

4.2 DOME Tools

The tools being developed in DOME are:

Extraction tool uses software reverse engineering techniques to extract ontology from legacy systems by analysing database schemas and the associated application programs [14].

Ontology editor and browser are tools that allow users to visually manipulate ontologies. The editor and browser display ontology in a tree-like structure with nodes denoting concepts. The details of the concepts can be displayed by highlighting nodes. In the editor mode, definitions can be modified, nodes can be added or removed. Ontology relationships can also be displayed.

Ontology mapping tools include tools that manipulate mapping databases and a rewrite engine which can convert one term to another. The rewrite engine also uses rules defined in the mapping database. Attribute-based merging tool uses attribute semantic clustering ideas to automatically identify possible mappings between terms of two ontologies [18].

Ontology consistency checking and validation via sources. Ontology consistency checking is by translating ontology to CLASSIC [19]. If the terms are extracted from the underlying data sources through ontology extraction tool, objects could be retrieved and displayed to the designer to verify and confirm mapping rules. Sometimes it is much easy to spot mapping errors by looking at the actual data.

Ontology composition, i.e., merging two ontology. This will be similar to the idea of the ontology algebra along the line proposed by Wiederhold [20].

4.3 DOME ontology development process

The DOME ontology development process combines the top-down and bottom-up approaches [21]. Figure 4 shows the overview of the DOME ontology development process.

![Figure 4: Overview of DOME ontology development process](image-url)

The top-down process (from Standards Re-eng of Figure 4) starts with domain analysis to identify the key concepts of the domain. These key concepts can be obtained by re-engineering data standards which often exist in large organisations. After that competency questions [22] are defined, which can be used to guide ontology development and to measure the quality of
ontology obtained. This process produces a general domain ontology. As most of the data standards are flat in structure, lexicon-level ontologies such as WordNet [23] and CYC [24] will be required to organise and turn data standards into hierarchical ontologies. DOME is building tools to assist the use of WordNet, CYC, etc.

The bottom-up process (from Extraction of Figure 4) starts with the underlying data sources need to be reverse engineered. We are developing tools XRA to use software reverse engineering techniques to extract ontology from data sources. XRA takes relational database schemas and application programs to produce an initial ontology which includes entities, relations and functions. Application programs often contain new entities and relations that are not present in data schemas.

The rest of the task is to connect the bottom-up ontology with the top-level domain ontology through ontology editors. The discovered ontology is implemented in CLASSIC which can be used to check the consistency of the ontology and helps with merging ontologies.

5. Ontology-based contents description

In e-commerce, contents description serves two purposes: contents-based resource locating and data retrieval from the resource. Take a product catalogue as an example. The contents description of the catalogue must support to answer queries formulated over general abstract product description (such as fridge with capacity of 1.8 litres and green colour) which may or may not use the terms of the catalogue. Once the resource has been located, SQL-like queries can be issued to retrieve the required data from the source (such as models, prices and colors).

We propose that every resource is associated with an ontology and a set of DTDs derived from the ontology. The ontology defines the semantics of the terms used to model the resource. The set of DTDs is used to mark up the resource. The ontology supports contents-based resource locating while the DTDs give the syntax and tags to construct SQL-like queries. Given an ontology defined in DOME, DTDs are the terms of the ontology that denote data-level entities and their associated attributes. This is analogous to the role of database schemas in relational database. As with database schemas, XML/DTD also lacks high-level concepts that are required for communications. The abstract concepts in ontology are used for knowledge-level communications while DTDs are used for data retrieval. As DTDs are well structured, SQL-like queries can be developed.

DTDs are a collection of tags and their meanings can only be determined by querying their associated ontologies. Figure 5 illustrates the relationships between contents, their DTDs and ontology. Given a data source, ontology designers use the DOME ontology extraction tool to develop an ontology for it. A set of DTDs are derived, defined, and then used to encode the data source. When an application or agent requires data from the data source, it will construct a set of queries over the DTD encoded data source. If the application or agent does not understand the tags of the DTDs associated with the data source, it will query the ontology server. The ontology server is capable of mapping terms, returning relationships between terms and attributes associated with the terms.

When an application or agent needs to find a particular data source, it will construct queries over a shared ontology. DOME will use its mapping databases and term rewrite engine to find relevant ontologies that have associated DTDs and marked-up information sources.

6. Related work

Ontology has been studies by many projects both in academia and industry. Most of these projects have some ontology development support. However, few serious efforts exist to provide ontology building environments which we argued are valuable in bringing systems to support e-commerce.

Ontolingua [7, 25] of Knowledge Sharing Lab, University of Stanford is a large project focusing on ontology development. It has built sophisticated tools for
developing and maintaining frame-based ontologies. It focuses on formal ontology specifications, reuse and translation to different ontology implementation systems. It is not addressing problems related to legacy systems and tools to merge ontologies.

SKC [26] also from University of Stanford is another project focusing on ontology development. SKC is researching ontology composition by developing ontology algebra. If it succeeds, it would greatly ease ontology development. In DOME, we have used an incremental mapping database to merge ontologies.

Another environment to build ontology is DODDLE [27], a tool developed at the School of Information of the Shizuoka University. In order to build an ontology the user gives a (not structured) set of domain terms to DODDLE. These terms are then linked to a machine-readable dictionary (MRD) by the spell match resulting in a hierarchically structured set of all the nodes on the path from these terms to the root of a MRD. As a matched node (concept) from a MRD sometimes has one or more senses, the system supports the user in choosing the sense that is most suitable for the user’s purposes. Dome uses domain based lexicon ontology which the meaning is well constraint, more precise than MRD.

7. Conclusion

We have discussed the roles of ontology in e-commerce and various types of semantic heterogeneity of information sources. With the phenomenal growth of electronically available information and the emergent electronic commerce, ontology development environments will reduce system development time and facilitate system interoperability through the use of a common methodology, representation and incremental mapping databases. To this end, we have discussed the DOME project that is focusing on bringing legacy systems into virtual enterprises by developing ontologies using software reverse engineering techniques.

DOME is still in its development stage, most of the tools need to be further developed. In addition, we would like to study the various types of semantic heterogeneity and develop specific types of mapping functions for each type semantic heterogeneity.

8. Acknowledgement

The work on extracting ontology from legacy system is done with Hongji Yang of De Montford University. We have also benefited through discussions with Valentina Tamma of Liverpool University.

9. References


Appendix: DOME Ontology Definition

<Alfa-n~eric-string> :=
    {<Char> | <Upper-Case-Char> | <Digit>}*;
<Attribute-name> := <Alfa-n~eric-string>;
<Attribute-specification> :=
    Attribute: <Attribute-name>
    [Description: <Upper-case-string>]
    Value-type: <concept-name>
    [Super-attribute: <Attribute-specification>*]
    [Sub-attribute: <Attribute-specification>*]
    [Constraints: {<Constraint>,}+];
<Attribute-type> := <Enumerated-type> I 
    List of <Attribute-type> | <Concept-name>;
<Argument> := <Term> ;
<Axiom-specification> := Axiom: <Atomic-sentence> | 
    <Axiom-Specification> <Connective> <
    Axiom-Specification> | Not <Axiom-Specification> ;
<Atomic-sentence> :=
    <Term> <Relational-operator> <Term> ;
<Char> := "a" | . . . | "z" | "A" | . . . | "Z" ;
<Concept-name> := <Upper-case-string> ;
<Concept-or-Attribute-name> :=
    <Upper-case-string> "("<Upper-case-string> ")" ;
<Concept-specification> := Concept: <Concept-name>
    [Description: <Upper-case-string>]
    [Hyponym-of: 
        {<Concept-or-Attribute-name>,} ] child of
    [Hypernym-of: 
        {<Concept-or-Attribute-name>,} ]parent of
    [Type-of: {<Instance-specification>,} ]
    [{<Attribute-specification>}]
    [Constraints: {<Constraint>,}+];
<Connectives> := Implies I Equivalent I And I Or;
<Constant> := <String> | <Number> |
    <Alfa-n~eric-string> ;
<Constraint> := <Atomic-Sentence> ;
<Digit> := "0" | . . . | "9" ;
<Enumerated-type> := "{<Constant>,}" ;
<Function-body> := <Term> <Operator> <Term> ;
<Function-name> := <Upper-case-string> ;
<Function-type> := <Concept-name> ;
<Instance-specification> :=
    Instance: {<Constant>,} Of <Concept-name> ;
<Number> := {<Digit>} ;
<Ontology> := Begin-Ontology: <Ontology-name>
    Included-Ontologies: {<Ontology-name>,}
    [Description: <String>]
    End-Ontology: <ontology-name> ;
<Ontology-body> := {<Concept-specification>,}+ 
    {<Relation-specification>,}*
    {<Instance-specification>,}*
    {<Axiom-specification>,} ;
<Ontology-name> := <Upper-case-string> ;
<Operator> := + | - | * | / =
<Quantifier> := Exist | Forall ;
<Relation-name> := <String> ;
<Relation-specification> :=
    Relation: <Relation-name> : <Relation-type>
    [Description: <Upper-case-string>]
    Arguments: <Argument>, {<Argument>,}+ 
    [Subclass-of: {<Concept-name>,} ]
    [{<Attribute-specification>}]
    [Constraints: {<Constraint>,}+]
    [Function: <Function-name> : <Function-type>]
    [Description: <Upper-case-string>]
    [Arguments: {<Argument>,}+]
    Body: <Function-body> ;
<Relation-type> := one-to-one | one-to-many | many-to-one | many-to-many ;
<Relational-operator> := "=" | "<" | "=" | "<" | "=" | "<" | "=" |
    "=" ;
<String> := {<Char> | <Upper-case-char> | <Digit>} ;
<Term> := <Constant> | <Entity-name> |
    <Attribute-name> | <Relation-name> ;
<Upper-case-char> := "A" | . . . | "Z" ;
<Upper-case-string> := <Upper-case-char> | <String> ;