Bridging Semantic Gap

Pronab Ganguly*, Fethi A. Rabhi and Pradeep K. Ray

School of Information Systems, Technology and Management
The University of New South Wales, Kensington
*Tel: 61 2 9691 5762; Fax: 61 2 9691 9574
E-mail: pganguly@qantas.com.au

1. Intent

Software interoperation by semantics ensures that the requestor and the provider have a common understanding of the requested information. An example of semantic heterogeneity is the use of synonyms, such as employees or staff, which are used to refer to the same concept in different information systems. This type of software interoperation includes the semantics of the user queries and of information sources. The bridging gap pattern intends to bridge the semantic gap between the requestor and the provider .

2. Context

Semantic incompatibility often occurs when old data or procedures are used for new purposes not anticipated by their original developers or among new systems that are the product of independent development efforts. In both the cases, this is because the semantics and procedures and data are not explicit. Requesters cannot determine whether providers match their assumptions. The results of such mismatch can be catastrophic – wrong results, sometimes with hidden or delayed indication that they are wrong. For example, in the late 1980's, the Regan administration in USA began including military personnel in the base figure for calculating "unemployment". When those figures were combined with earlier figures that did not include the military, the "unemployment" rate appeared to drop by 0.3%. Another example is Ariane 5, where an exception occurred while converting one type of number into another type in the upgraded version of software where code was reused from previous version.

3. Problem

XML provides a common syntax to exchange heterogeneous information. Usually a Document Type Definition (DTD) or an Extensible Markup Language (XML) Schema is used as a standard mechanism to exchange information. But these schema-level specifications cannot resolve the issues related to semantic heterogeneity due to following reasons:

- There are many such schema-level specifications and they do not use the consistent set of terminology
- Data, captured in different files, having the same set of labels are not true representative of the consistent terminology.
- For a small number of systems, programs can be developed to translate terminologies between systems but scalability cannot be achieved.

4. Forces

The major forces involved are:

- Meta-level data capture the richness of meanings conveyed by the data and normally human intelligence resolves any differences associated with the meaning..
- To develop a computer readable meta-data is hard as the semantics of a term varies from context to context such as one information source may refer the term "apple" as a type of computer while another information source may refer the same term "apple" a type of fruit.
- Meta-data contains highest-level user/ business information requirement. You develop ER diagrams and database schema from the metadata. Schemas are implementation platform specific. Constrains at this level are implementation specific. Implementation specific schemas may not explicitly represent the constraints that are present in the meta-data level. Thus each meta-data may have different implementation specific schemas.
- Usually XML a Document Type Definition (DTD) or an Extensible Markup Language (XML) Schema is used as a standard mechanism to exchange information. It does not provide dynamic mapping and transformation between terms in a given context.
- Scalability – for a small system, programs can be developed to translate the terminologies, but that is not possible for a large system.

5. Solution

To address the above forces, draw on a formal Ontology – a shared model of the domain – for the vocabulary and formalism of the computational specifications. The Ontology helps to structure our concepts for effective computing. Ontology abstracts reality in order to understand and process it. This model is computer

readable. Thus Ontology is a formal, explicit specification of a shared conceptualization [DIE01].
As Ontologies are formal theories of about a certain domain, it requires a formal logical language to express them. Some of the ontological languages include [DIE01]:

- CycL and Knowledge Interchange Format (KIF)
- Ontolingua
- Frame Logic
- Description Logics

Ontolingua is designed with a clear logical semantics based on KIF to support the design and specification of Ontologies. Ontolingua definitions are Lisp-style forms, which contains a symbol with an argument list, a documentation string, and a set of KIF sentences labeled by keywords. An Ontolingua Ontology includes definitions of classes, relations, functions, distinguished objects and axioms that relate these terms. Document Type Definitions (DTDs) in XML are closest to ontological modeling with the following significant differences [DIE01]:

- Is-a relationship of classes is a central theme in ontology but this is absent in DTDs.
- DTD's do not have inherent mechanisms
- In Ontology, the ordering of attribute descriptions does not matter but in DTDs it does.

However mapping between ontology and DTD and vice versa are possible and realized [DIE01].

6. Example Implementation

Our proposed bridging gap pattern revolves around the use of a Domain Ontology Server and a DTD Mapper as described below:

- Given a data source, the domain experts build the domain ontology. The abstract concepts are used for higher level communications. Additionally, it is capable of mapping terms for a given context, relationships between terms and attributes associated with the terms. The scope of a domain , such as diabetes management in healthcare informatics, needs to be specified to manage the task. Within that domain, mappings such as relation between different units of measurement, relation between similar terms are specified.
- If the user application does not comprehend the default DTD, the user application of the default DTD submits its request to the domain ontology server. Domain ontology server resolves the issue and generates the relevant DTD with the help of DTD mapper. Then the ontology server guides the user application to the regenerated DTD.
- The application uses the regenerated DTD to access the information.
- These regenerated DTDs will constitute a repository for reuse

The diagram in figure 1 illustrates the structure of our pattern. It consists of following elements:

6.1 Domain Ontology Server - Given a data source, the domain experts build the domain ontology. The abstract concepts are used for higher level communications. Additionally, the ontology server is capable of mapping terms, relationships between terms and attributes associated with the terms.

6.2 DTD Mapper - Depending on the chosen ontological agreements, the mapper produces the relevant DTD. The mapper takes the resolved meta level information from ontology server and map that information into relevant DTD .

The interaction sequences are described below:

1. The ontology, with the help of DTD mapper, generates the default DTDs for a given data source.
2. When an application requires access to the data source, it will construct a set of queries through the default DTD encoding.
3. If the user application does not understand the tags of default DTD, it will submit a request to the Ontology server specifying the inadequacy of the default DTD.
4. The Ontology server is capable of mapping terms. It can also return relationship between terms and associated attributes.
5. Based on the request of the user application, the ontology server, with the help of DTD mapper, will generate a new DTD relevant to the user application and guide the user application to the regenerated DTD.
6. Over time, the ontology server and the DTD mapper will build a repository of DTDs for reuse.

7. Known Uses

7.1 Diabetes Management in Telehealthcare

Our main use of this pattern is in Telehealthcare where it is under implementation. The focus is on diabetes management. In diabetes management, the semantically equivalent concepts can be described as:

- Property mismatch such as different units for blood sugar e.g. mmol/l or mg/dl
- Different Properties – Measurement of glycohaemoglobin (GHb) provides a measurement of blood sugar level over time. There are four different major techniques and twenty different units [COL01]. Hence integration of results requires semantic integration.
- Different terms – "Intensive" and "Tight" glycemic control may be used to refer to the same concept but inner components may be different.

Depending on the application specific requests, the ontology will generate relevant DTD and request the application to commit to that DTD. The tagged DTD will provide the necessary information that will be used for integration.

7.2 Other Application Areas

Other application areas include business-to-business e-commerce, where dynamic information exchange is vital, following semantic issues may need to be resolved.

- Different Terms: One business application may refer to their personnel as employees whereas another business application may refer to their personnel as staffs.
- Different Properties: One business application may specify monthly for their employees' salary whereas the other application may specify them as annual salary.
- Property Mismatch: The salaries may be expressed in different currency units.

In other business-to-business e-commerce, following scenario may arise:

- For the same product, one business application may include colour in its catalogue whereas the other may not.

7.3 Related Patterns

This pattern is similar to agent based following patterns [FER00]:

- The Dependency Separation Pattern which reflects inter-application dependencies
- Interface connection pattern that reflects domain based communication mechanisms

But the above patterns do not explicitly address the issues related to semantic interoperation based on Ontology and DTD.

8. Resulting Context

This pattern uses DTD mapper to generate the relevant DTD from ontology. However instead of using DTD mapper, the Ontology can advise the user application to restructure its query based on default DTD [ZHA01]. In that case, the computing load will be transferred to user application side whereas in our model, the ontology server carries out the related tasks. If the computing load is transferred to the user application side, the application needs to reconstruct its query which involves additional and undesirable computational processing. However development of a comprehensive domain-specific Ontology and corresponding DTD mapper is complex task. The following patterns will work well with this pattern:

- Any pattern which can relate to DTD as well as to Ontology either directly or through an interface.
- In software agent oriented paradigm, as the agents will be able to interact directly with Ontology [PRO01], the DTD mapper and corresponding DTDs are not required.

9. References

[COL01] Colman.A et al; Glycemic Control; http://www.mja.com.au/public/issues/jul21/colman/colman.html. Accessed on 28/2/02

[DIE01] Dietel Fensel; Ontologies: A silver Bullet for Knowledge Management and Electronic Commerce;Springer-Verlag Berlin Heidelberg; 2001

[FER00] Fernandez, George & Zhao Liping: A Pattern Language for Federated Architecture; Proceedings of KoalaPLoP 2000: 24-26 May: Melbourne Australia, Technical Report TR-00-7, Department of Computer Science, RMIT, Australia,pp.21-31

[PRO01] Pronab Ganguly, Fethi A. Rabhi & P.Ray; The Semantic Interpreter Pattern;
Proceedings of KoalaPLoP 2001: Melbourne Australia,

[ROS99] Rosenthal A, Sciore E; Description, Conversion, and Planning for Semantic Interoperability;
http://www.mitre.org/pubs/data.mgt/papers/sem.pdf
Accessed on 29/12/1999

[ZHA01] Zhan Cui, Dean Hones and Paul O'Brien;Semantic B2B Integration: Issues in Ontology-based Approaches;

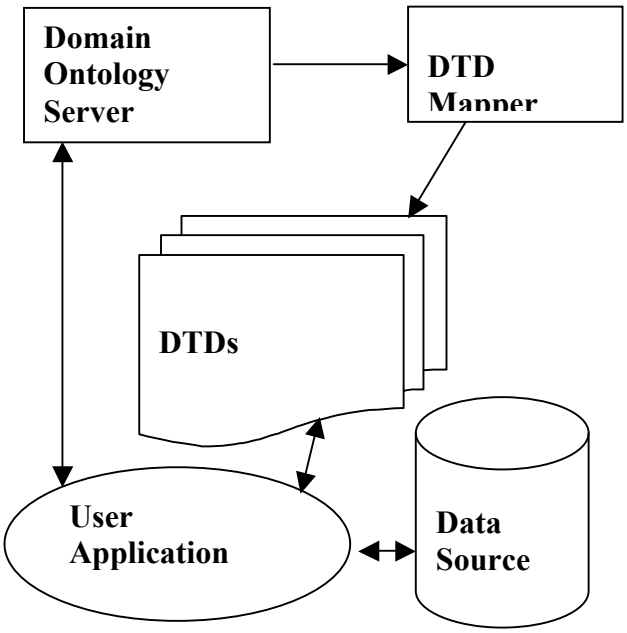

Figure 1: Structure of Ontology based system