Implementation of Ontology Based Business Registries to Support e-Commerce

Manjith Gunatilaka, Gihan Wikramanayake, Damitha Karunaratna
University of Colombo School of Computing
manjith@dmsswt.com, gnw@ucsc.cmb.ac.lk, ddk@ucsc.cmb.ac.lk

Abstract

The Business Registries are one of the key elements in the Business to Business (B2B) transaction process in e-Commerce. It acts as knowledge centers by offering services to different business parties to collaborate their business processes in an effective manner. Currently it is challenging to extract accurate information needed by a business party who is querying a registry on a particular industrial domain due to its inability to store business specific domain information effectively.
In this research paper we show how we improved the storage of a business domain specific knowledge by utilizing the implementation of Ontologies. For this we have selected "IT Outsource" as our reference implementation Ontology, which helps IT companies engaged in outsourcing business to setup their business repositories in an effective manner.
Once Ontology is represented in the Business Registry, business parties could to automate their search process by using Ontology based querying and automating Agent based search.

1. Introduction

The Electronic Business XML (ebXML) [1] and Universal Description, Discovery and Integration (UDDI) [2] are the commonest business infrastructure frameworks commercially available to perform B2B transaction process. All these frameworks are built with business registries and repositories as main components, which is used to advertise services each business party is involved.

1.1 Business Registries

The Registry stores services related meta information where as repository is a container which captures data. Currently these Registries implement knowledge of a business domain using their own coding schemes which is specific to its own implementation. These coding schemes restrict deductive inference capabilities in a registry and hence affect business parties collaborating business transactions in an interoperable manner.
The use of ontology based classification structures for the representation of classification relationships and its properties inside the registry is not used by any of these registries. We have used OASIS [3] open ebXML registry implementation (freeebXML) [4] as an example registry which is one of the popular open source ebXML framework specific registry. In real-world there are a number of business registries such as Adobe – eForm registry [5], SDMX – statistical registry [6], Apelon – Medical guideline registry [7], General Motors and Vokswagon B2B artifacts registry and IBM UDDI business test registry [8].

1.2 Ontology

Ontology is a description of a content and relationship expressed as a formal vocabulary. This can be used to classify domain specific information in different areas such as in education, engineering, natural sciences etc.
Most of the business information stored in repositories in Business to Business (B2B) infrastructure frameworks is difficult to extract due to unstructured domain classifications in business registries [9]. Therefore use of ontology as an information source to formalize domain knowledge and finding a way of mapping this information to registry content is a practical solution to enhance B2B communication. We have used OWL [10] ontology language to represent the web semantic content in the repository.

Rest of the paper is organized as follows: Section 2 introduces registry implementation model which is represented using UML diagrams. Thereafter in section 3 we briefly describe ontology, OWL ontology language and Description Logic (DL) which are used in ontologies in. Section 4 describes our reference ontology domain - "IT Outsourcing" along with its design and tools used to develop. Thereafter section 5 describes basics elements in OWL and how it is mapped with ebXML RIM model. It presents different stages in OWL to ebXML RIM mapping process. Section 6 describes "OWL-ebXML Onto" transformation engine we developed to implement ontologies into business registries and final section 7 concludes with a short summary and future enhancements in business registries.

2. Registry Implementation Model

The types of objects stored and how these objects are organized is defined in the ebXML Registry Information Model (RIM) [11]. The implementation of RIM is done in the form of a relational or object database schema. The registry objects can be categorized into two types namely classification related objects (Figure 1) and service related objects (Figure 2).

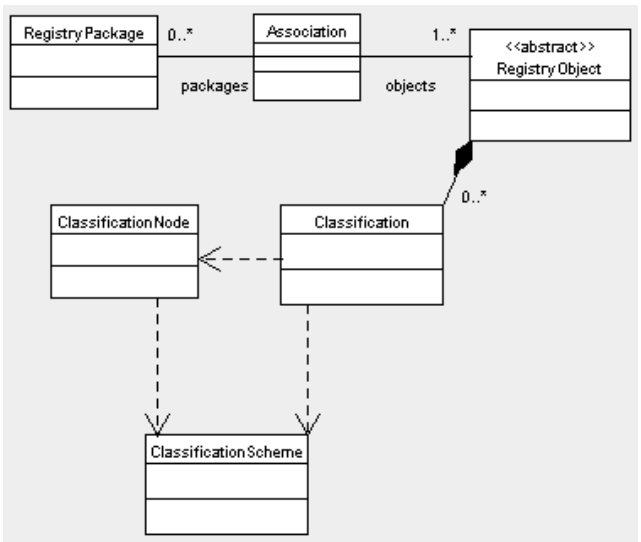

Figure 1: Classification related objects of RIM

All information stored in the registry is represented using an abstract class called *RegistryObject* which provides minimal meta data. These registry objects are classified or grouped into different classification schemes such as North American Industry Classification System (NAICS) [12], ISO3166 [13] or according to user define classification structures.

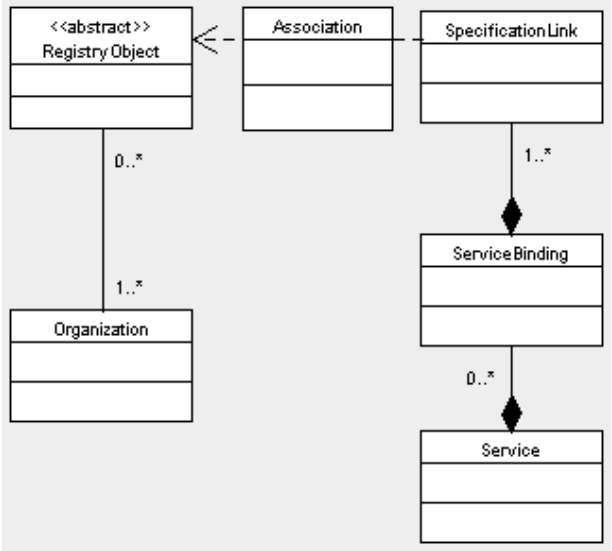

Figure 2: Service related registry objects of RIM

The classification schemes are further subdivided into classification nodes which define tree structures under *ClassificationScheme* registry object. These classification scheme instances and taxonomy values are identified using *Classification* registry object. Once these important registry objects instances are defined, grouping of logically related registry objects are specified using *RegistryPackage* instance.

The *Associaition* registry object instance is the most important element in the RIM which defines many-to-many associations between registry objects. These associations are classified into fourteen predefine association types such as "RelatedTo", "Uses", "HasMember", "EquivalentTo", "Contains" etc.. These types specify relationship between source and target registry objects.

The second type of RIM objects (*Organization*, *Service, ServiceBinding and SpecificationLink*) are used to specify registry services. The *Organization* registry object specifies available organizations in the registry and its published services are listed in the Service registry object (e.g. Web Services). The ServiceBinding registry object represents technical information on specific ways to access a specific interface published by a particular service. The SpecificationLink registry object lists down how to use the service.

3. Ontology and Semantic Data Interpretation

Ontology is an explicit specification of a conceptualization [14]. It represents concepts in a domain of discourse (Classes), properties of each concept describing various features and attributes of the concept (Slots) and restrictions on Slots (Facets).
There are two classifications of ontologies namely Thesaurus-like (formal) and Descriptive ontology. Thesarus-like ontology is first generation of ontologies which arrange terms into subsumption hierarchy and link them with other relationship to express synonym, composition, etc.. The Descriptive ontology define properties of concepts and their interrelationships which gives semantically rich representation of intended domain. We used descriptive ontology to represent our domain knowledge.

3.1 Description Logic

The Descriptive ontology is based on Description Logic (DL) which is the key concept behind knowledge representation. The DL is formalized on *Concepts* (i.e. Classes), *Roles* (Binary relationship between Classes and its cardinality) and *Constructors* (Table 1).

Constructor	DL Syntax
interSectionOf	$C1 \bigcap C2$
unionOf	$C1 \bigcup C2$
complementOf	$\neg C1$
oneOf (enumeration)	$\{x_1, x_2, \ldots x_n\}$
allValueFrom	$\forall P.C$
someValueFrom	$\exists P.C$
hasValue	$\exists P.\{X\}$
minCardinality	$\exists >= nP$
maxCardinality	$\exists <= nP$
cardinality	$\exists = nP$

Table 1: Constructors in DL

The Constructors define complex concepts. *Axioms* (Table 2) are used to name complex concepts and to state subsumption relationship between concepts. The Constructors include Union, Intersection, Negation, Existential restriction, Value Restrictions etc.. Axioms are categorized into two types, namely, Definition axiom and Inclusion axiom. Definition axioms introduce names for concepts and Inclusion axiom asserts subsumption relations.

Axioms	DL Syntax
subClassOf	$C1 \subseteq C2$
equivalentClasses	$C1 \equiv C2$
subProperty	$P1 \subseteq P2$
equivalentProperty	$P1 \equiv P2$
sameAs	$\{X1\} \equiv \{X2\}$
disjointWith	$C1 \subseteq \neg C2$
differentIndividualAs	$\{X1\} \subseteq \neg\{X2\}$
inverseAs	$P1 \equiv P2^{-1}$
transitiveProperty	$P+ \subseteq P$

Table 2: Axioms in DL

3.2 Ontology Language – OWL

The OWL is a web ontology language which is used to describe a relationship between classes. It uses DL as a main construct to specify information related to a specific domain. It is stored in a form of an XML/RDF (Resource Description Framework) format. The OWL is an extension to DAML (DARPA Agent Markup Language) + OIL (Our Ideas of a Language) [15] combine with additional areas other than DL is Frame based systems and web languages such as XML and RDFS.

OWL is divided into three sub languages, OWL Lite, OWL DL and OWL Full. The OWL Lite represents simple class hierarchies, where as OWL DL supports for maximum expressiveness which adds completeness and decidability based on DL. The OWL Full supports maximum expressiveness and freedom is given to users to define their own RDF formats [16].

4. The Reference Ontology

We have used IT Outsource domain as reference ontology. The knowledge acquisition in IT outsource domain was done by referring to leading outsourcing company web sites.

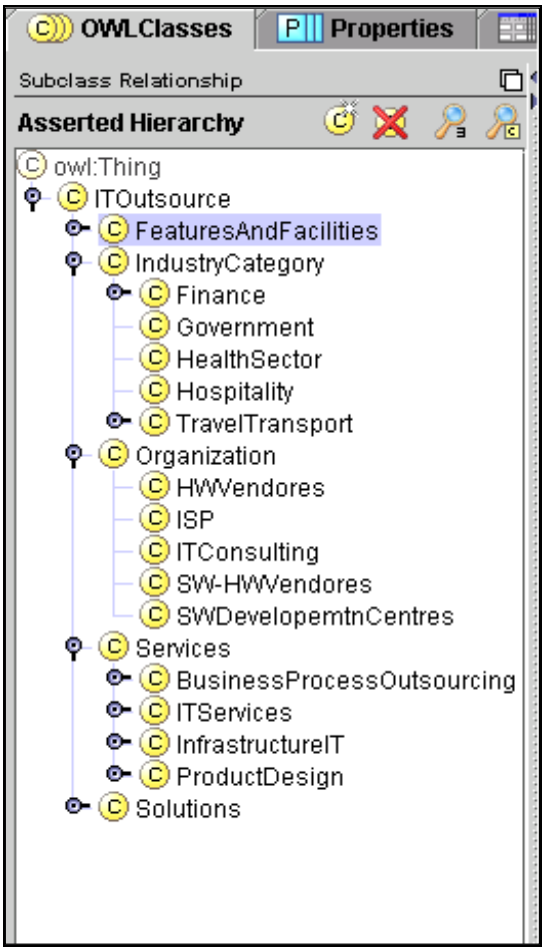

Figure 3: Reference ontology for IT Outsource domain

4.1 IT Outsourcing Domain

The “IT outsourcing" means hiring somebody outside your company to provide IT services. Outsourcing is most common for companies whose IT needs are well known in advance. It allows companies to focus on broader business issues while delegating operational details for handling by an outside expert. Basically any IT outsourcing company focus on different market segments (industry categories); specific solutions cater to those industries and type of services they offered. Other than these important elements, features and facilities provided by IT solutions are important elements in IT outsourcing domain. Any company

selects an IT outsource option because it brings benefits such as improve business focus, gain world class solutions, accelerate reengineering benefits to organizes their processes, share risk with another organization by delegation IT tasks and possible allocation of existing resources for more strategic activities.

4.2 Ontology Design

The knowledge gathered was represented to a reusable ontology using the Protégé [17] tool (Figure 3.0). The ontology is classified into five classification schemes, namely, different types of organizations involved in IT outsourcing, industry categories, services provided by different organizations, solutions these organizations provided and finally types of features/facilities available in these solutions. Each classification is further analyzed and sub categorized into different classification hierarchies. The relationship between these classification hierarchies and its nodes were identified and it was represented using DL. Properties were defined according to OWL classification (data and object properties) and necessary restrictions were enforced using asserts conditions given in the Protégé tool.

After completing the design of IT Outsource ontology, it was converted to OWL language (OWL XML format) using a conversion tool provided with Protégé. This automation process reduces additional effort necessary to convert the domain specific ontology to OWL.

5. OWL and ebXML RIM Object Mappings

Due to inadequacies in object registering and storing structures given in the existing registry implementation models, it is not possible to store semantic information using web semantic languages such as OWL. We have identified a practical way of mapping the existing structures to our IT outsourcing reference business registry, ebXML, using OWL specifications and hence overcome this problem.

5.1 Basic Elements in OWL

5.1.1 Classes

The basic OWL ontology elements consist of *Classes*, *Properties*, instances of classes and their relationships. These elements are mapped into ebXML registry objects to represent the domain knowledge inside the registry. The class and sub class relationship in OWL is represented using owl tags <owl:Class> and <rdfs:subClass>. The example code snippet given in 1.0 represents, “InsuranceSol” is a sub class of an “ITServiceSol” class.

```
<owl:Class rdf:ID="InsuranceSol">
  <rdfs:subClassOf>
    <owl:Class  rdf:about="#ITServiceSol"/>
  </rdfs:subClassOf>
</owl:Class>
```

Code snippet 1.0

5.1.2 Property

The second type of element in OWL is called the *Property* which defines the binary relationship between classes or between classes and RDF literal data types. There are two important property types, ObjectProperty and DatatypeProperty.
The ObjectProperty defines relationship between two classes (Binary relationship). The example code snippet 2.0 shows “erpSolutions” is an ObjectProperty which relates “ERP” and “ERPSol” classes. Its rdfs:domain and rdfs:range tags provides additional information to the relationship by indicating source and target classes which bound relationship end points.

```
<owl:ObjectProperty rdf:ID="erpSolutions">
  <rdfs:domain rdf:resource="#ERP"/>
  <rdfs:range rdf:resource="#ERPSol"/>
</owl:ObjectProperty>
```

Code snippet 2.0

The DatatypeProperty represents binary relationship between classes to data types. The example given in Code snippet 3.0 represents “costSolution” relationship has a data type property between “Solutions” and “String” data type.

```
<owl:DatatypeProperty rdf:ID="costSolution">
  <rdfs:domain rdf:resource="#Solutions"/>
  <rdfs:range  rdf:resource=
    http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

Code snippet 3.0

The scope of a class can be restricted using anonymous classes. This is achieved using Object or Datatype property restrictions. The code snippet given in 4.0 represents restriction applied for a class using a <owl:Restriction> tag. In this example “DatawarhousingSol” has been restricted using Datatype property which indicates it has a “costSolution” property and has a value type Boolean. The OWL defines set of predefine restriction properties, such as “owl:allValuesFrom”, “owl:someValueFrom”, “owl:cardinality”, “owl:minCardibality” and “owl:maxCardinality”.

5.1.3 Individual

The third important element in OWL is called “Individuals” which represents instances of a class. The example code snippet given in 5.0 represents

"Organization" class has an instance called "DMS" and it has two attributes, name of the organization and service industry.

```
<owl:Class rdf:ID="DatawarehousingSol">
  …..
  <owl:Restriction>
    <owl:hasValue rdf:datatype=
     http://www.w3.org/2001/XMLSchema#boolean>
     true
    </owl:hasValue>
    <owl:onProperty>
   <owl:DatatypeProperty rdf:about="#costSolution"/>
    </owl:onProperty>
  </owl:Restriction>
 ……..
  </owl:Class>
```

Code snippet 4.0

```
<Organization rdf:ID="DMS">
  <serviceIndustry rdf:resource="#InBoundTours"/>
  <nameOrganization rdf:datatype=
  "http://www.w3.org/2001/XMLSchema#string">
   DMS Software Technologies</nameOrganization>
</Organization>
```

Code snippet 5.0

The Object and Datatype properties can be further restricted using the complex property characteristics such as "TravsitiveProperty", "SymmetricProperty", "FunctionalProperty" and "InverseOf".

There are instances which need to specify the collection of class relationships in the ObjectProperty using advance type of class descriptions such as "unionOf", "intersectionOf" and "complementOf" properties in OWL. The example code snippet 6.0 represents how "ITServiceIndustry" property is restricted using domain class object as "ITService" and range is restricted to "Collection" rdf:parseType which comprises collection of classes such as "Finance" and "Government".

```
<owl:ObjectProperty rdf:ID="ITServiceIndustry">
  <rdfs:domain rdf:resource="#ITServices"/>
  <rdfs:range>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Finance"/>
     <owl:Class rdf:about="#Government"/>
     …..
    </owl:unionOf>
   </owl:Class>
  </rdfs:range>
 </owl:ObjectProperty>
```

Code snippet 6.0

5.2 Mapping Stages

The objective of the mapping stages is to implement a mapping mechanism that transforms domain knowledge represented in OWL to ebXML RIM. We have identified five mapping stages which will transform OWL to ebXML RIM specifications.

5.2.1 Mapping of Classification Hierarchies

The class and sub class relationship represented in the OWL (code snippet 1.0) is mapped into classification hierarchies using *ClassificationScheme*, *ClassificationNodes* and *Classificaion* registry objects (code snippet 7.0). The following code snippet shows how classifications of industry categories are submitted to the ebXML registry model.

```
<ClassificationScheme id="IndustryCategory-id"
 … >
<ClassificationNode id="Finance-id"
        parent=" IndustryCategory-id " code="Finance">
 <ClassificationNode id="Government-id"
      parent=" IndustryCategory-id " code="Government" />
 <ClassificationNode id="HealthSector-id"
       parent=" IndustryCategory-id " code=" HealthSector "
/>
```

Code snippet 7.0

5.2.2 Mapping of Properties

The Object and Datatype Property (Code snippet 2.0 and 3.0) binary relationship in OWL is converted to *Association* registry objects. Its source and target attributes represents two classes define in the OWL. The Association type can be specified according to predefine association types in ebXML RIM. The complex property characteristics such as *TransitiveProperty*, *FunctionalProperty*, *SymmetricProperty* etc. are defined as a new association type and special classification schema was created which is called "ComplexProperty" and its node representing all the complex properties in OWL.
The following code (code snippet 8.0) shows mapping of OWL code snippet 2.0 into ebXML registry object. Similar conversion is applied to OWL code snippet 3.0 by indicating target registry as a "String" ClassificationNode.

```
<Association id = "erpSolutions"
   associationType = "Contains"
   sourceObject = "ERP"
   targetObject = "ERPSol-id"
/>
```

Code snippet 8.0

The following example (code snippet 9.0) represents complex property mappings in the OWL. It defines "solType" as a *FunctionalProperty* by defining new Association type called "Functional" which defines solution type property that can either be based on "IndustryCategory" or "Service" classification scheme.

```
<Association id = "solType"
    associationType = " Functional "
    sourceObject = "IndustryCategory"
    targetObject = "Service"
/>
```

Code snippet 9.0

The important OWL element called *Restrictions* shown in the code snippet 4.0 can be mapped as a new type of Association called "Restriction". The restriction related properties such as "hasValue", "sameAs", "onProperty" etc. is mapped as Slot objects. Following code (code snippet 10.0) shows how Restriction can be mapped into ebXML registry objects.

```
<Association id = "DatawarehouseSolRestrict-id"
    associationType = " Restriction "
    sourceObject = "DatawarehousingSol "
    targetObject = " costSolution">

   <Slot name = "hasValue">
          <ValueList>
             <Value>true</Value>
          </ValueList>
   </Slot>
</Association>
```

Code snippet 10.0

5.2.3 Mapping of Organizations

The classification of *Organization* class in the reference IT Outsource ontology represents organization class hierarchy. The code snippet 5.0 is converted to ebXML registry object according to following code samples (code snippet 11.0).

```
<Organization id="dms-id"
                    primaryContact = "manjith-id">
    <NameOrganization>
       <LocalizedString lang="us-en" value="DMS/>
   </Name>
   <ServiceIndustry>
       <LocalizedString lang="us-en"
        value="Software  Development " />
  </ServiceIndustry>
 <Slot name = "Employee Count">
      <ValueList>
         <Value>50</Value>
      </ValueList>
  </Slot>
      <Address streetNumber="165"
                  street="Dharamapala Mw."
                  city="Colombo"
                  stateOrProvince="Western"
                  postalCode="0007"
      country="SL" />
  <TelephoneNumber countryCode="1"
                        areaCode="781"
                        number="442-0703"
                        phoneType="office"/>
          ....
</Organization>
```

Code snippet 11.0

5.2.4 Mapping of Collection Hierarchies

The collection of class relationships represented using rdf:parseType="Collection" and use advanced type of class descriptions "unionOf", "intersectionOf" and "complementOf" is converted into registry packages in

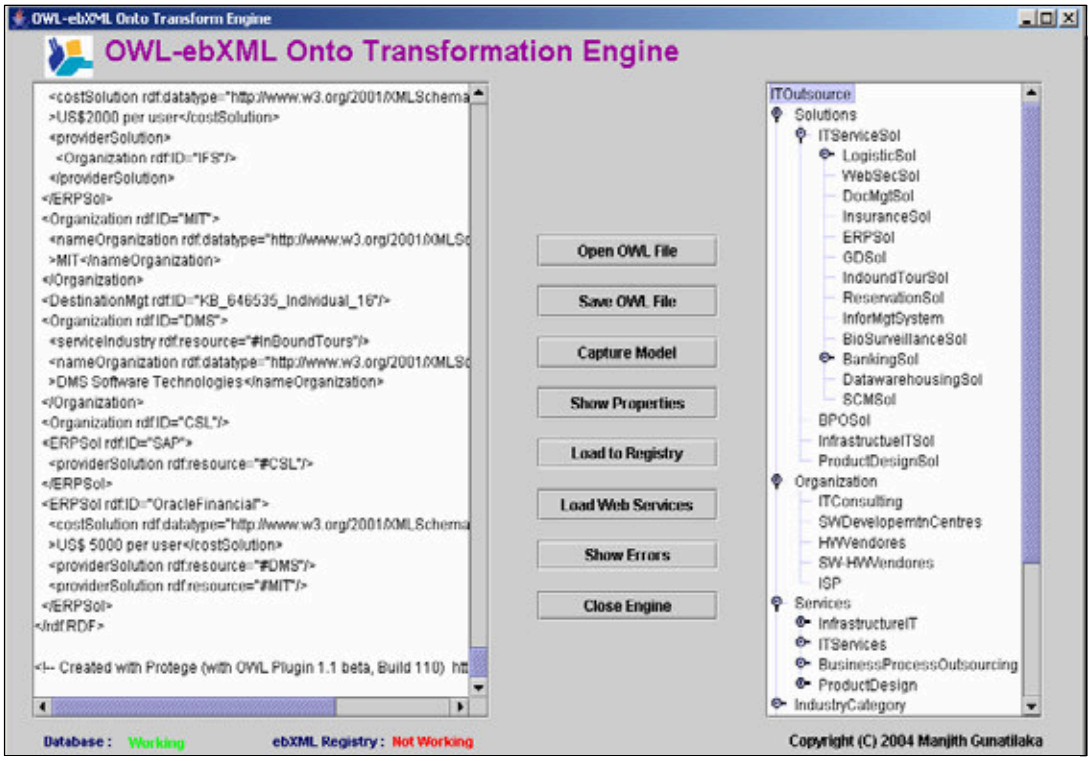

Figure 4: OWL-ebXML Onto Transformation Engine

ebXML registry. The example code snippet 6.0 is converted into following code (code snippet 12.0) in ebXML registry.

```
<RegistryPackage id = "ITServiceInd-id">
    <Name>
      <LocalizedString value = "IT Service Industry Pkg"/>
    </Name>
    <Description>
      <LocalizedString value =
          "It packages all IT service industry categories"/>
    </Description>

    <RegistryObjectList>
        <ClassificationNode id = "Finance-id" >
            <Name>
              <LocalizedString value =
                        "Financial Sector"/>
            </Name>
            <Description>
              <LocalizedString value =
                    "All Financial sector industries"/>
                        </Description>
          </ClassificationNode>
          <ClassificationNode id = "Government-id" >
              <Name>
                <LocalizedString
                value = "GovernmentSector" />
              </Name>
              <Description>
                  <LocalizedString value = "All Public
                          sector industries"/>
              </Description>
        </ClassificationNode>
                  …
    </RegistryObjectList>
</RegistryPackage>
<Association id="ITServiceIndustryAsso-id"
    associationType="Unionof"
    sourceObject="ITServices"
    targetObject=" ITServiceInd-id"
</Association>
```

Code snippet 12.0

5.2.5 Mapping of Services

The most important ontology classification scheme in our reference ontology is the *Service* classification scheme. It represents types of services provided by the IT Outsource domain and these services are advertised in the registry as a web service registry objects. The example code (code snippet 13.0) represents XML instance to register CRMeService as a web service. The *ServiceBinding* registry instance provides information about how to access the web service interface (i.e. URI information). The *SpecificationLink* registry instance provides technical specification details such as Web Service Description Language (WSDL) document.

```
<! --Service objects  - ->

<Service id="CRMeSerivce-id">
    <Name>
        <LocalizedString lang="us-en"
          value="CRM Registry Service" />
    </Name>
    <Description>
          <LocalizedString lang="us-en"
            value="CRMeServices " />
    </Description>
    <ServiceBinding
                id="CRMeServiceSOAPBinding"
                accessURI=http://localhost:8080
                /ebxmlrr/registry/soap/CRMeService">
                …
        <SpecificationLink
                id=" CRMeServiceSOAPBinding SpecLink"
                specificationObject=
                "CRMeSErvicesWSDL">
                  …
          </SpecificationLink>
   </ServiceBinding>
</Service>
```

Code snippet 13.0

Once the service is advertised, it is important to define the service specification link information as an *ExtrinsicObject* or *ExternalLink*. The following code (code snippet 14.0) sample shows how to define an *Extrinsic* registry object transform to the CRMeService [18].

```
<!—WSDL document – ExtrinsicObject -->
<ExtrinsicObject id=" CRMeServicesWSDL "
      mimeType="text/xml">
          …
 </ExtrinsicObject>
```

Code snippet 14.0

6. OWL-ebXML Onto

The implementation of ebXML registry with domain specific information which is represented in OWL, was stored using a transformation processes. The mapping of OWL syntax to ebXML RIM was identified during the detail analysis of the OWL and RIM properties, structures and elements. This information was converted according to OWL vs. ebXML RIM mappings given in the Table 3. The results gathered during the detail analysis of OWL syntax to ebXML RIM specification is listed in section 5.

The process of above transformation is automated using "*OWL-ebXML Onto*" transformation engine (Figure 4). The architecture of the "OWL-ebXML Onto" is shown in Figure 5. It has two main components *Ontology Analyzer* and *Ontology Transformer*. The process of transformation of any ontology is started with

conversion of ontology to an OWL syntax using a Protégé converter. The converted file is stored as an .owl and it is loaded to the transformation tool. Once the file is uploaded, it is parsed through an OWL ontology parser (Jena [19]) and an ontology model is created.

- **Enumerated Class Analyzer** - analyze collection related classes and store that information in the database.

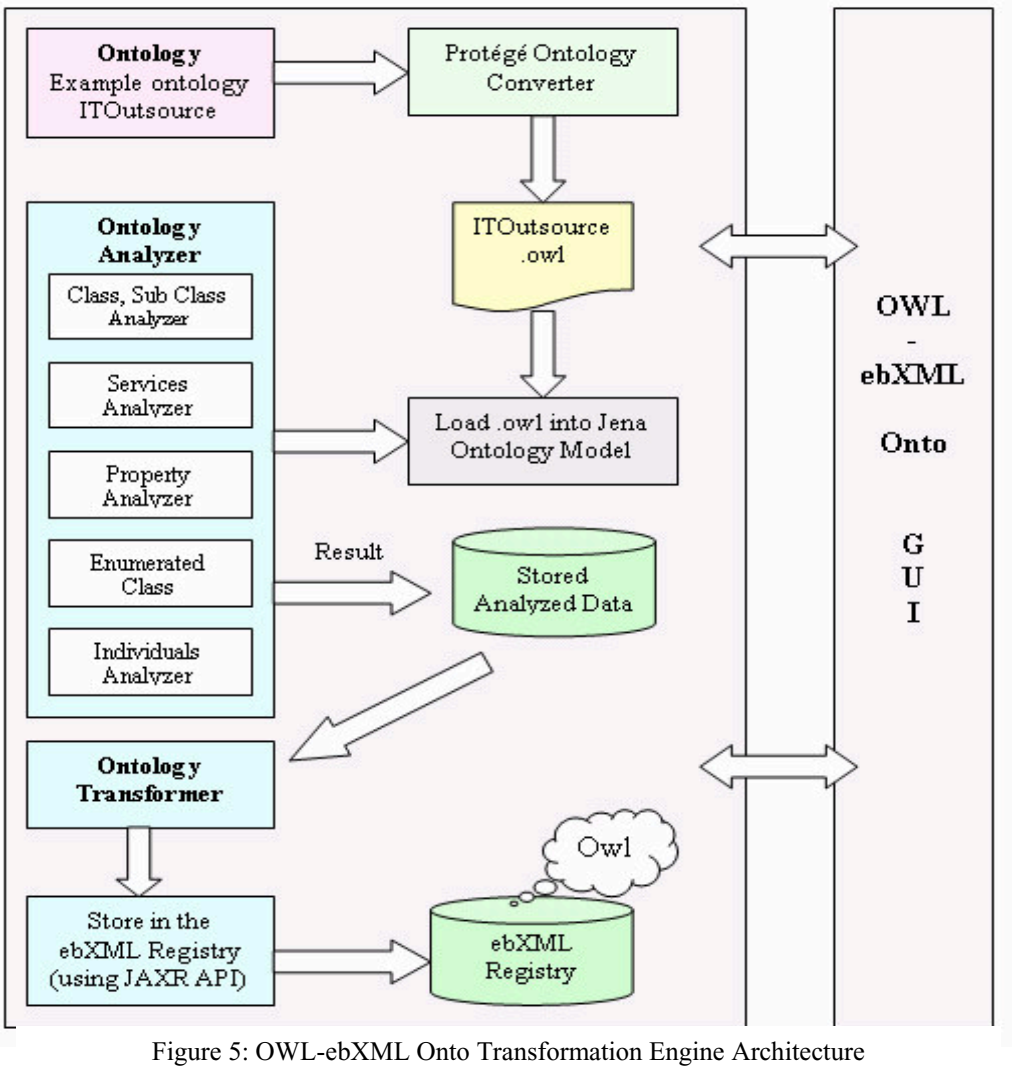

Figure 5: OWL-ebXML Onto Transformation Engine Architecture

The next step is to process the ontology model using ontology analyzer. It has five sub analyzers:

- **Class and Sub Class Analyzer** - analyze classes and sub class relationships and store that information in the database.
- **Service Analyzer** – analyze service ontology and converts into web service specific information and store that information in the database.
- **Property Analyzer -** analyze ObjectProperty, DatatypeProperty, complex and advanced properties. Once analysis is completed, store that information in the database.
- **Individual Analyzer** – analyze instance related data of a class and store that information in the database.

After completion of ontology analysis process, the next step is to transform the analyzed ontology into RIM specification. This is processed using the *Ontology Transformer*. It has two components, namely *"OWL Basic Element Transformer" (OBET)* and *"Web Service Transformer" (WST).* The OBET uses Table 3 data and already analyzed data in the database to perform transformation. It will convert OWL model into ebXML specific registry objects during the transformation process. It creates all classes and sub class relationships, properties as associations and individual elements as data values in classification nodes. Once basic elements are registered to the ebXML business registry, registering of web services can be performed using

OWL	ebXML
owl:class	ClassificationNode
rdfs:subClassOf	ClassificationNode
rdfs:domain	Source Registry Object
rdfs:range	Target Registry Object
owl:ObjectProperty	Association - predefine or new Association type called “objecttype”
owl:Restriction	Created a new Association type called “restriction”
owl:onProperty	Created a new Association type called “onproperty”
owl:minCardinality	Add a Slot attribute called “mincardinality” to an Association
owl:cardinality	Add a Slot attribute called “mincardinality” to an Association
owl:maxCardinality	Add a Slot attribute called “maxcardinality” to an Association
owl:DatatypeProperty	Association between Class and RDF/simple XML datatype. We define new Association type called “datatype”
rdfs:subPropertyOf	Add a Slot attribute called “subproperty” to an Association
owl:oneOf	A Registry Package with collection of a new Association type called “oneof”
owl:TransitiveProperty	Created a new Association type called “transitive”
owl:SymmetricProperty	Created a new Association type called “symmetric”
owl:sameAs	Created a new Association type called “sameas”
owl:differentFrom	Created a new Association type called “differentfrom”
owl:distinctMembers and owl:AllDifferent	Create a Registry Package Object with all distinct members and then assign a Slot which has an attribute “alldifferent”
owl:unionOf	Create a Registry Package Object with all union classes and then assign a Slot which has an attribute “unionof”.
owl:hasValue	Add as a Slot to a Registry Object
owl:FunctionalProperty	Created a new Association type called “functional”. If the domain is defined as a rdfs:subProperty then take it as a domain, otherwise take it as a Slot attribute.
owl:InverseFunctionalProperty	Created a new Association type called “inversefunc”.
owl:allValuesFrom	Add a Slot attribute called “allvaluesfrom” to an Association
owl:someValuesFrom	Add a Slot attribute called “somevaluesfrom” to an Association
owl:intersectionOf	Created a new Association type called “intersectionof” which has source registry object as Class object and target registry object as Registry Package.
owl:unionOf	Created a new Association type called “unionof” which has source registry object as Class object and target registry object as Registry Package
owl:complementOf	Created a new Association type called “complementof”
owl:equivalentClass	Created a new Association type called “equivalentclass” having source and target registry objects are either classes or registry packages.
owl:disjointWith	Created a new Association type called “disjointwith”

Table 3: ebXML and OWL Transformation Table

WST. It creates necessary RIM specific service instance objects, its relationships between service organizations, service binding and points to specification details such as WSDL documents etc..

The transformed ebXML registry objects which was generated from OWL ontology is registered to the ebXML registry using Java API for XML Registry (JAXR) [20]. All meta data in the ebXML registry and analyzed data was stored in the Oracle database. The tool was developed using Java and user friendly GUI features were provided to view, load and analyze transformed ontology to the ebXML registry.

7. Conclusion

Effective business collaboration between two parties in B2B environment in e-commerce is an issue due to inadequacy and inability to represent business domain knowledge. Though there are few commercially available business registries, storing business domain knowledge and formalizing business jargon is a problem. Therefore incorporation of business domain knowledge using ontology to the existing business registries is a practical approach we have adopted.

During our research effort we addressed this concern by exploiting similarities in ontology web language OWL and example registry ebXML to propose a transformation method. Based on this method, we discussed and proposed an implementation of a transformation engine called *OWL-ebXML Onto* which is an ontology based semantic web content plug-in.

This research effort can be further extended by introducing ontology based query engine to business registries, which will utilize stored ontology to deduct and infer effective search results which is not provided with the existing query managers in the business registries.

The future work in ontology and business registry would be direct incorporation of ontology based repositories capable of Semantic Content Management (SCM),

which will remove third party plug-in and make business repositories more organized and effective. This will enhance Small, Medium and Enterprise (SME) business partners to setup there registry centric business operation more effective and efficient by improving existing bottlenecks in the B2B transaction process in e-commerce.

8. References

[1] ebXML,http://www.ebxml.org,[visited on 12/05/2004]
[2] UDDI, http://www.uddi.org, [visited on 12/05/2004]
[3] OASIS - http://www.oasis-open.org/home/index.php, [12/05/2004]
[4] freebXML - http://ebxmlrr.sourceforge.net/, [12/05/2004]
[5] Adobe eForm, http://xml.gov/presentations/adobe/PDFA-XMP-Registry.pdf, [12/05/2004]
[6] SDMX , http://www.sdmx.org,[12/05/2004]
[7] Apelon – http://www.apelon.com, [2/05/2004]
[8] IBM UDDI registry, https://uddi.ibm.com/testregistry/find, [12/05/2004]
[9] Manjith Gunatilaka, Gihan Wikramanayake, Damitha Karunaratne, "Improving B2B Transactions by Exploiting Business Registries", In Proceeding for the 23rd National IT Conference, Colombo, Sri Lanka, 2004, pp 1
[10] OWL language overview, http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.3, [12/05/2004]
[11] RIM - ebXML Registry Information Model Version 2.0, April 2002, http://www.ebxml.org/specs/ebRIM2.pdf, [12/05/2004]
[12] North American Classification System, http://www.census.gov/epcd/www/naics.html, [12/05/2004]
[13] ISO 3166 - http://www.iso.ch/iso/en/prods-services/iso3166ma/, [12/05/2004]
[14] Thomas R. Gruber, "Towards Principles for the Design of Ontologies Used for Knowledge Sharing", 1993, pp 1-2
[15] DAML/OIL, http://www.daml.org, [28/07/2004]
[16] OWL Ontology Language Guide, http://www.w3.org/TR/2004/REC-owl-guide-20040210/, 2004 [28/07/2004]
[17]] Protégé, http://protege.stanford.edu/, [28/07/2004]
[18] Joseph M. Chiusano, Booz Allen Hamilton, Farrukh Najmi, "Registering Web Services in ebXML Registry", Version 1.0, March 2003
[19] Jena , http://jena.sourceforge.net/documentation.html, [12/05/2004]
[20] JAXR, http://java.sun.com/xml/jaxr/index.jsp, [2/08/2004]