Using Semantic Web Technologies to Integrate the Enterprise

Dr. Marwan Sabbouh
Dr. Joseph K. DeRosa

The MITRE Corporation
Bedford, MA 01730

Abstract. *This paper illustrates the benefits of inserting a semantic layer into information systems that use N-Tier architectures. By introducing the Ontology and Semantic services above the data tier, we succeed in not only specifying semantics but also in integrating software components into the enterprise. This design allows us to integrate legacy systems, or RDBMS and Web services, without software development.*

1 Introduction

People in an enterprise typically address business needs through established processes and complex information systems. In order for business processes to create synergy and value, the information systems that support operations must be integrated across the enterprise. Traditionally, enterprise integration has been accomplished with data aggregation and integration, client server architectures, and middleware. With the advent of web technologies, N-tier architectures and web services have become common facilitators of integration. However, enterprises do not remain static. They are called upon to address new business needs or incorporate additional information sources. Even if there are existing software components available, the IT departments typically build new code to integrate them into the enterprise. This is an expensive and time-consuming process.

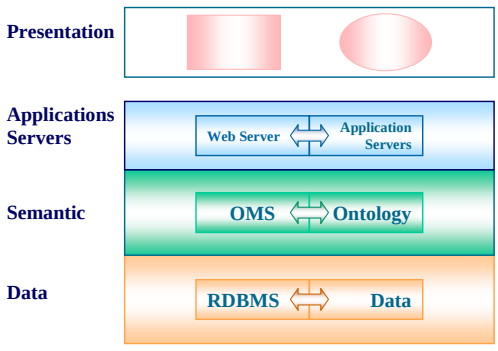

Figure 1. N-Tier Architecture

This paper illustrates the proper use of Semantic Web technologies [1-3] to integrate existing software components – without the need for new software development. We capture the concepts, relationships, and business rules of the enterprise in an ontology [4], and then isolate that ontology in a single layer of the N-tier architecture, as shown in Figure 1. Users can now access software resources, e.g., a relational database management system (RDBMS) or a web service (WS), through an ontology management system (OMS) that exposes those resources in a language familiar to the business users and subject matter experts (SMEs). A new integration paradigm emerges:

(1) software components with structured formalisms are expressible as meta-ontologies,
(2) and by linking their meta-ontologies to the enterprise domain ontology, we succeed in integrating them without writing new code.

For this analysis we consider the integration of a legacy RDBMS and a new WS. For each, we postulate a simple upper ontology and meta-ontology. The meta-ontologies are derived from the RDBMS schema and the Web Services Descriptive Language (WSDL) description. First, SME's link concepts from the meta-ontologies to the enterprise domain ontology, then the OMS augments the domain ontology with instance data according to the defined mappings. The OMS manages the ontology analogously to the way the RDBMS manages data for legacy applications. We then implement a Semantic Viewer that, for a given request, is able to access data in the RDBMS and invoke the WS necessary to satisfy the request. We present the details of this process and illustrate with a simple example.

2.0 Integration Using Semantic Web Technologies

While Semantic Web technologies provide all the integration benefits of using XML syntax, they also capture both the meaning (semantics) of the enterprise concepts and their relationships in the vocabulary of the business experts, i.e., the SME's. Once this is done, it allows the integration of new business processes at the level of the semantic layer.

We illustrate how this works with a simple case in which an enterprise with an N-tier architecture wishes to integrate

This paper was written under the support of Air Force ESC Contract FA 8721-04-C-001

a new web service with the existing enterprise database. The steps we take are as follows:

1. **Create enterprise domain ontology/ontologies.** If the enterprise does not already have an enterprise domain ontology, we create one. In practice an enterprise may contain several domain ontologies that may or may not be linked. In time, these domain ontologies mature and become linked through an iterative process. For our purposes we assume the enterprise contains a single domain ontology.
2. **Create and link database meta-ontology.** If the legacy database does not have a meta-ontology, we create one based on the database schema. For our purposes we assume the legacy applications access the database through the traditional interfaces in the RDBMS, and we have to create a database meta-ontology. We link the concepts in the database meta-ontology to those in the domain ontology, and use the OMS to augment the domain ontology with the instance data from the database.
3. **Create and link web service meta-ontology.** Similarly, we create a web service meta-ontology based on the WSDL description, link concepts from the web service meta-ontology to those in the domain ontology, and use the OMS Edit service to augment the domain ontology with concepts then describes the web service
4. **Broker a User Request.** We create a semantic viewer in the application server layer that accesses the ontology according to user input. The Semantic Viewer application is created once and used many times (for all ontology-based integration of software components).

In this section, we illustrate how to build the semantic web solution. Our example considers an enterprise that wishes to integrate into the business processes a customer database and a mapping web service to place customer locations on a map.

Assume that the enterprise uses the terminology "Street," "City or Town," "State," "Zip-Code" and has the concept of "customer" and "location of customer" and a mapping web service, in this case Mapquest [5], that has the terminology "address or intersection," "city," "Zip Code" and the concept of "Map" and "Aerial Photo."

.

2.1 The Domain Ontology

The enterprise domain ontology E captures all the concepts and relationships in a domain of the enterprise and expresses them equivalently in an ontology language or a directed graph. The ontology languages OWL [6], RDF [7] provide a uniform structure of triples, {subject, relationship, object}, for the description of "any" data. For our example, the part of the enterprise domain ontology relating customer location can be expressed by the triples:

E::{Business, Has_A, Customer}
E::{Customer, Has_A, Location}
E::{Location, Has_A, Street}
E::{Location, Has_A, State}
E::{Location, Has_A, Zip-Code}
etc...

Equivalently we could express these relationships in the graph of Figure 2.

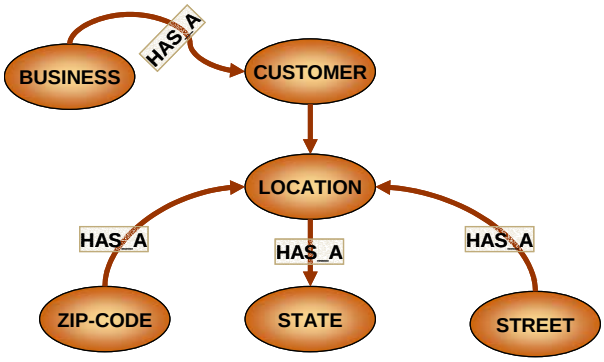

Figure 2 – Location Ontology

Note that the domain ontology is created by experts in the business domain in which the enterprise is engaged (perhaps with the assistance of ontology experts). Contemporary Ontology Management Systems [8,9] contain EDIT and STORE tools to create and store the domain ontology.

2.2 Integration of an Enterprise Database into the domain Ontology

Next we need to populate concepts in the Domain Ontology with instance data resident in the corporate databases. To do this we create a meta-ontology for the database. This meta-ontology consists of two parts: the database upper ontology and the database entities and relationships as instances of the database upper ontology.

A. The Database Upper Ontology D^U

This ontology contains triples that characterize databases in general. Generally speaking, this ontology is the modeling of the database formalism in triples. For example:,

D^U:: {Database, Has_Tables, Tables}
D^U::{Table, HAS_Columns, Columns}
D^U::{ Column, Is_A Primary-Key}
D^U::{ Column, Is_A Foreign-Key}

Where D^U:: represents the database upper ontology namespace.

B. The Database Meta-Ontology D_M

When integrating a database into the ontology, the database schema is read by the DB MAPPER tool in the OMS and a database meta-ontology is created. Consider the simple Address table shown in **Figure** 3 It has an Address_ID as the primary key, and Street, City, State, and Zip as columns (attributes). For brevity, only two of the data records are shown.

Address_ID	Street	Zip_Code	STATE
001	255 North Rd.	01824	MA
002	21 High St.	01851	MA

Figure 3 Address Table

The triples for the meta-ontology become

D_M::{ Address, Is_Instance_Of, Table}
D_M::{ Street, Is_Instance_Of , Column}
D_M::{ State, Is_Instance_Of , Column}
D_M::{Zip, Is_Instance_Of, Column}
D_M::{Address, Has-Columns, Street}
etc...

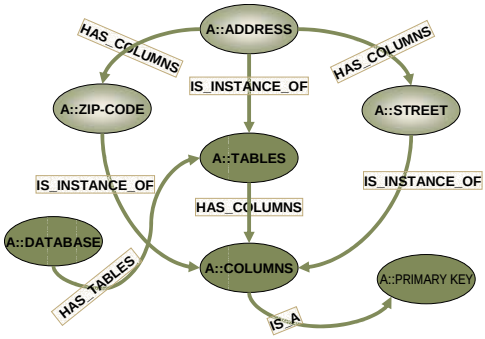

Figure 4 - A Portion of the Database Upper- And Meta-Ontologies

Where D_M:: represents the database meta-ontology namespace. The database upper- and meta-ontology graphs are shown in Figure 4

C. The Augmented Enterprise Domain Ontology E^+

To create an augmented domain ontology $\mathbf{E}^+$ with the instance data from the database, we simply link the concepts in the database ontologies D^U and D_M to the concepts in the domain ontology E. This is done using the relationship HAS_SOURCE from the DB MAPPER tool within the OMS. This relationship specifies each column in the database as instances of a particular enterprise domain concept. In our example

{ E::Location, HAS_SOURCE, D_M::Address}
{ E::Street, HAS_SOURCE, D_M::Street}
{ E::City or Town, HAS_SOURCE, D_M::City}
{ E::State, HAS_SOURCE, D_M::State}
{ E:: Postal Zip, HAS_SOURCE, D_M::Zip}

When a relationship in the ontology connects two different classes to specify its triple, the mapping of instance data from the database to this triple must explicitly specify how instance data are matched. In this case, the relationship IS_JOINED_BY links triples in the domain ontology to entities in the meta-ontology. These entities are typically Foreign keys columns in the joining table.

To create the augmented Domain Ontology containing all the database instance data, the DB MAPPER simply creates a new graph containing the compostion of the Domain ontology linked with the meta-ontology

{Customer, Has_A, Location}
{Location, Has,_A Street }
{255 North Rd., Is_A, Street }
etc...

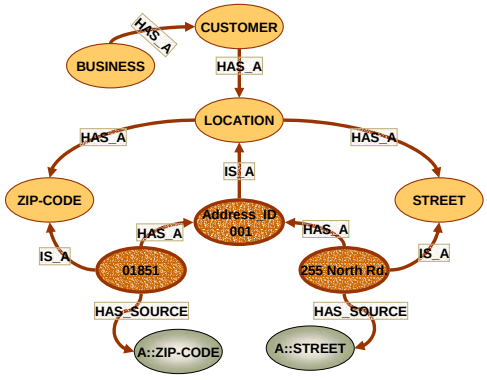

Figure 5 – Augmented Enterprise Domain Ontology (E^+)

The augmented enterprise domain ontology graph is shown in Figure 5. In what follows, we will drop the triple notation and revert to the equivalent graph notation.

2.3. Integration of a Web Service into the Domain Ontology

To integrate a Web service into the enterprise, we augment the domain ontology with concepts from the Web services description language (WSDL) [10] file, and we create a meta-ontology that contains the invocation information of the Web service. Then, we link the meta-ontology to the augmented domain ontology.

A. The Web Service Upper Ontology W^U

We postulate an upper ontology that models Web services as concepts of Inputs (in-parameters), Output, Classification-Conditions and Effects. Figure 6 shows a portion of the graph of W^U. Classification-Conditions are conditions that relate inputs to concepts in the domain ontology. These conditions must hold true in order that a Web service be able to operate on an instance of the domain ontology. Effects are post processing conditions that are true as a result of running the Web service on instance data in the domain ontology. The Semantic Viewer (described below) makes use of both Classification-Conditions and Effects. W^U is similar to Service Profile of DAML-S. [11]

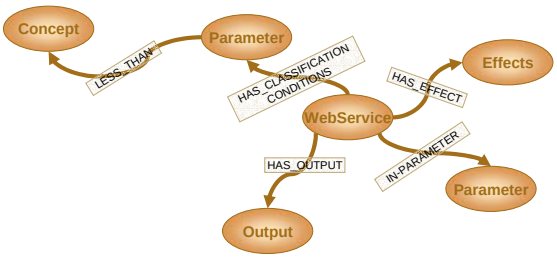

Figure 6 Portion of the Web Service Upper Ontology

B. Augmenting the domain ontology with Instances of W^U to form Ontology E^{++}

When integrating a Web service into the enterprise, an instance of ontology $\mathbf{W^U}$ is created**.** The instance ontology $\mathbf{W^I}$, is then linked to the augmented domain ontology using the IS_INPUT_TO relationship to form Ontology $\mathbf{E^{++}}$. Currently, this is done manually by the SME using the Edit service of the OMS. This is shown in Figure 7.

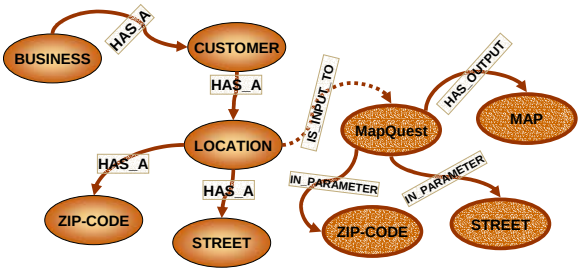

Figure 7: Instance Ontology W^I

C. Web Service Invocation Upper Ontology V^U

We postulate an upper ontology that models how to access and invoke a Web service. The model has concepts like those found in WSDL. Figure 8 shows a subgraph of this ontology. V^U is similar to Service Grounding of DAML-S.

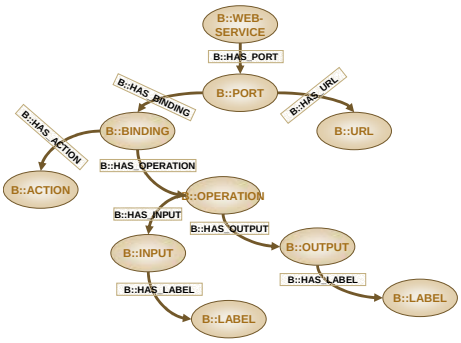

Figure 8 – A Portion of the Web Service Upper Invocation Ontology

D. Instances of Ontology V^U : Web Service Meta-Ontology W_M

When integrating a web service into the ontology, the WSDL file is read by the WSDL MAPPER tool in the OMS, and a Web service meta-ontology $\mathbf{W_M}$ is created.

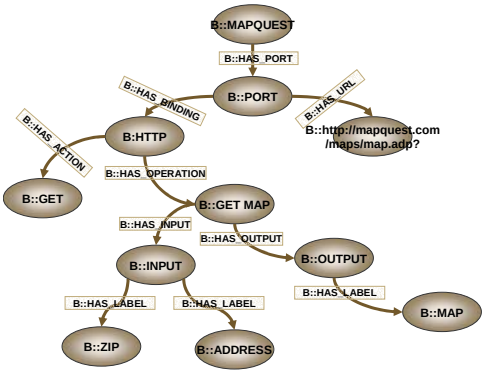

Figure 9 - Mapquest Meta-Ontology

$\mathbf{W_M}$ ontology is an instance of $\mathbf{V^{U.}}$ For our MapQuest example, the meta-ontology is shown in Figure 9.

E. Linking the Augmented Enterprise Domain Ontology to Ontology $\mathbf{W_M}$

In this step, the SME uses the IS_VALUE_OF, IS_SERVED_AT, and HAS_RESULT relationships to link the meta-ontology $\mathbf{W_M}$ to the augmented domain ontology $\mathbf{E^{++}}$. The IS_VALUE_OF links the range or object value of the HAS_LABEL relationship in the meta-ontology $\mathbf{W_M}$ to concepts in the augmented domain ontology $\mathbf{E^{++}}$. The IS_SERVED_AT relationship links the subject or domain of the HAS_OUTPUT relationships in the $\mathbf{E^{++}}$ontology to the object of the HAS_URL relationship in the $\mathbf{W_M}$. The HAS_RESULT relationship links the range of HAS_LABEL relationship in the $\mathbf{W_M}$ to the range of HAS_OUTPUT relationship in the ontology $\mathbf{E^{++}}$. This relationship is useful when the output of the Web service contains the inputs of another. This is shown in Figure 10.

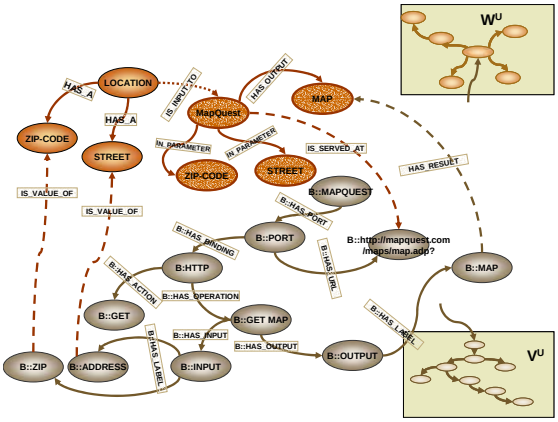

Figure 10: . Example of Ontology and Instance Data in Graph Format

2.4 Broker a User Request: The Semantic Viewer

The Semantic Viewer is a Web-Enabled application whose purpose is to broker a request and to return the desired result to the end user. When a user enters any input data, it is linked to concepts in the augmented domain ontology using the USER_INPUT USER_OUTPUT, and USER_CRITERIA relationships. The USER_OUTPUT links a goal the user is interested in a concept in the augmented domain ontology. The USER_INPUT links the user's input to concepts in the augmented domain ontology. The USER_CRITERIA is used to link a preference of the user to concepts in the augmented domain ontology. To illustrate the inner workings of the Semantic Viewer, we examine multiple scenarios:

CASE 1: User Inputs Are Found In Ontology $\mathbf{E^{\pm}}$

When a user enters input1 = "01851", input2 = "255 North Rd." and output = "Map," i.e, the user requests a map, the SME links:

1. input1 to ZIPCODE using the USER_INPUT
2. input2 to STREET using the USER_INPUT
3. output to Map using the USER_OUTPUT

Having linked the request to concepts in the augmented domain ontology, and having found "01851", "255 North Rd." and "Map" in the augmented domain ontology $\mathbf{E^{+}}$, the Semantic Viewer does the following:

1. The Semantic Viewer issues the request of the Compose service
2. The Compose service selects a sub-graph from the ontology that connects "01851", "255 North Rd." with "Map," as shown in Fig 11.
3. The Semantic Viewer displays the graph to the user
4. Upon a user's mouse click, for example clicking map, the Viewer automatically invokes MapQuest.com

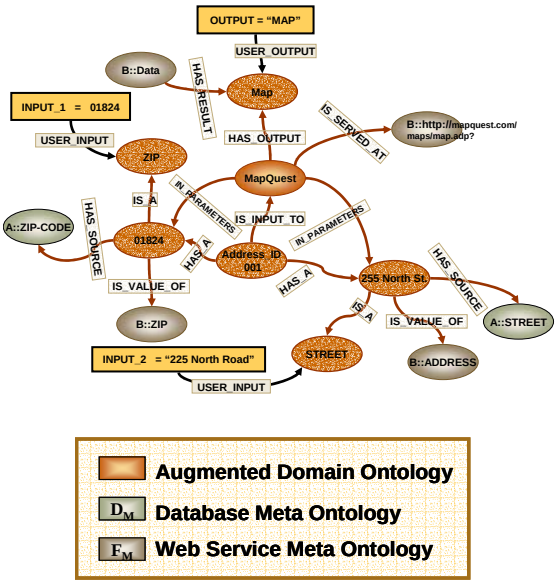

Figure 11 – Output of Compose Service

To illustrate the "context sensitiveness," suppose we integrate a web service "Aerial Photo" that reads in a .gif

image of the map and returns the Aerial View (in fact, this is "Aerial Photo" of MapQuest.com). By integrating web service "Aerial photo" [not shown in Fig 11], we mean connecting "Aerial Photo" to "MAP" and running the process illustrated in section 3.3. Now, suppose the user enters input1= "01851", input2= "255 North Rd." and requests "Aerial Photo". Having linked the request to concepts in the augmented domain ontology, and having found "01851", "255 North Rd." and "Aerial Photo" in the augmented domain ontology $\mathbf{E}^+$, the semantic viewer does the following:

1. The Semantic Viewer issues the request of the Compose service.
2. The Compose service selects a sub-graph from the ontology that connects "01851", "255 North Rd." with "Aerial Photo"
3. The Semantic Viewer displays the graph to the user
4. Upon a user's mouse click, for example clicking Aerial Photo, The Viewer figures out which services to call and in what order from the graph
5. Invokes the map service of MapQuest.com first
6. Obtains the result and extracts the desired portion of the output, and passes it to the aerial photo Web service as input
7. Invokes the "Aerial Photo".

CASE 2: One Or More Of The User Inputs Are Linked To The IN_PARAMETERS Of A Web Service

Unlike the previous case, the user's input cannot be found as instances of a concept in the domain ontology $\mathbf{E}^+$. In this case, we don't have a unique instance that the Web service can operate on as in the previous case. This is where classification conditions become useful. A classification condition exists when the object of IN_PARAMETERS of a Web service is linked to a concept in the domain ontology conditionally. (i.e. LessThan, GreaterThan,…) For instance, suppose a temperature weather forecast ontology exists for a region. I.e. temperature forecast for North East United States. Now suppose the existence of Web service that for a given latitude and longitude, it returns the temperature. In order for the Web service to operate on the temperature forecast, the latitude and longitude must fall within the North East United States region. In this case, the above mentioned service has in its $\mathbf{W}^{\mathbf{I}}$ ontology the following relationships:

- {WS HAS_CLASSIFICATION_CONDITIONS LATITUDE}
- {WS HAS_CLASSIFICATION_CONDITIONS LONGITUDE}
- {LONGITUDE LESS_THAN LONGITUDE-EAST}
- {LONGITUDE GREATER_THAN LONGITUDE-WEST}
- {LATITUDE LESS_THAN LATITUDE - NORTH}
- {LATITUDE GREATER_THAN LATITUDE - SOUTH}
- {WS IN_PARAMETERS LONGITUDE }
- {WS IN_PARAMETERS LATITUDE }
- {FORECAST IS_INPUT_TO WS}
- {TEMPERATURE IS_OUTPUT_TO WS}

Thus, having not found latitude and longitude in the ontology $\mathbf{E}^+$, the semantic viewer does the following:

1. queries for a Web service with IN_PARAMETERS=longitude and IN_PARAMETERS=longitude, which returns WS.
2. queries for classification conditions on latitude and longitude, which returns LONGITUDE-EAST, LONGITUDE-WEST, LATITUDE–SOUTH, and LATITUDE–NORTH.
3. queries for the values of LONGITUDE-EAST, LONGITUDE-WEST, LATITUDE–SOUTH, and LATITUDE–NORTH that satisfies the conditions: LONGITUDE LESS_THAN LONGITUDE-EAST, LONGITUDE GREATER_THAN LONGITUDE-WEST,LATITUDE LESS_THAN LATITUDE – NORTH, and LATITUDE GREATER_THAN LATITUDE -SOUTH
4. passes the values in 3 along with the output temperature to the compose service.
5. proceeds as in the previous case.

Taking Effects and USER CRITERIA into consideration

Effects are useful for Web services that operate on instance data in order to produce another instance data. Suppose a temperature weather forecast is produced according to a Forecasting Model. i.e. NOGAP. Assume a user desires a temperature weather forecast that is produced according to MM5 forecasting model. Now, suppose the existence of a Web service that operates on NOGAP and produces MM5. Then, the input is a temperature weather forecast and the

output is also a temperature weather forecast. However, the output is produced according to the MM5 model. In this case, the above mentioned service has in its $\mathbf{W}^{U}$ ontology the following relationships:

- {WS HAS_EFFECT MM5}
- {WS IN_PARAMETERS NOGAP}
- {WS IN_PARAMETERS MM5}
- {FORECAST IS_INPUT_TO WS}
- {FORECAST IS_OUTPUT_TO WS}
-

Thus, when the semantic viewer cannot match the USER_CRITERIA to an instance in the augmented domain ontology, it queries the augmented domain ontology for a Web service whose Effect matches the user criteria. If found, it invokes that web service to produce the desired instance data, then proceeds with its processing as in 2.4.

3.0 General Concept

A semantic layer consisting of an ontology and the Semantic Services above the data layer, explicitly captures the enterprise concepts, relationships, and business rules using RDF and OWL, as shown in Figure 12

In Figure 12, we view the Enterprise as a collection of Legacy systems and a set of evolving software applications. Often, the challenge is to integrate an existing Data Source (RDBMS) with an application program or a Web service.

This architecture accommodates the development of new applications, as well as integrating existing ones. The latter can be done without any new software development.

The Semantics Services are used to manage the ontology and to integrate existing and new software components. They provides a set of facilities to Edit, Store, Compose, and Query the ontology, as well as a Database Mapper to relate the ontology to the database schema and to load the data in the database to the ontology, and a WSDL Mapper to integrate Web services into the ontology. The Edit service is used to build and modify the ontology. Subject matter experts use the service to collaboratively negotiate the semantics. The Store service captures agreements negotiated by subject matter experts and persists them in a repository. The DB Mapper service maps concepts and relationships to various entities in the database schema. The Compose is a run time component that implements an algorithm that for given inputs and outputs returns a graph. By traversing the graph, this Semantic Viewer is able to provide context-sensitive information to the user.

The Query service provides both SMEs and applications the capability to retrieve concepts and instance data, (i.e., find concepts, find instances of concepts, find relationships between concepts, find concepts with certain relationships, etc.

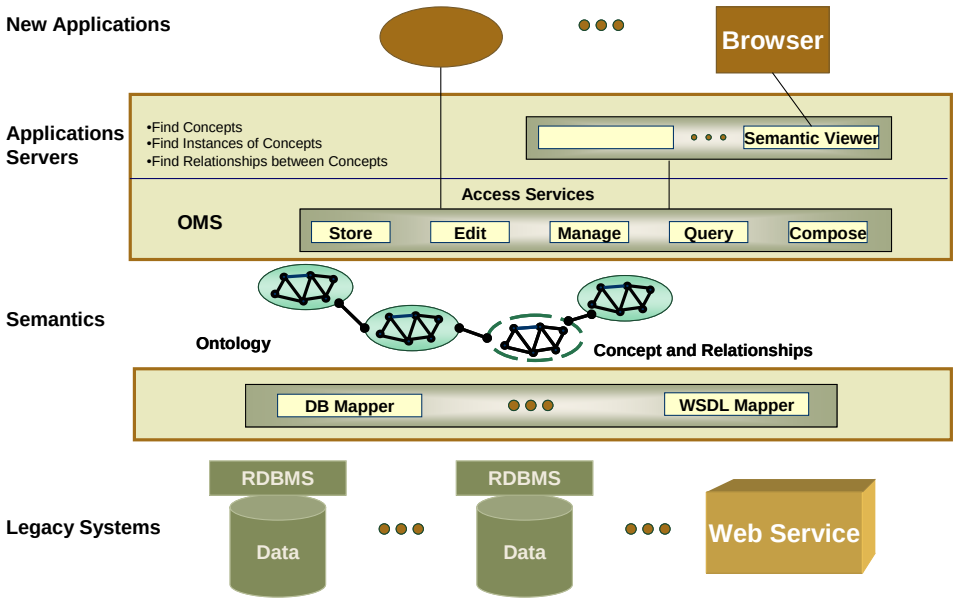

Figure 12. Ontology-Based Architecture

Summary

We have illustrated the integration of a legacy database and a new web service using semantic web technologies without the writing of new code. Specifically, we translated the database schema and the WSDL description of the web service into meta-ontologies. Then using the facilities of the OMS, we augmented the enterprise domain ontology with instance data from the database and with new concepts from web service. Then we showed how the new web service operates on the legacy database through a simple Semantic Viewer application. We showed how the concept can be generalized to the integration of any software components with semantic structure, e.g., disparate databases, Web Services and XML Documents. We presented a roadmap for the current use of semantic web technologies.

References:

1. T. Berners-Lee, J. Hendler, and O. Lassila, The Semantic Web, Scientific American, May 2001. T. Berners-Lee, Hendler J., and O. Lassila.
2. Michael C. Daconta, Leo J. Obrst, Kevin T. Smith, The Semantic Web : A Guide to the Future of XML, Web Services, and Knowledge Management
3. World Wide Web Consortium. http://www.w3.org
4. T.R. Gruber, A translation approach to portable ontologies. *Knowledge Acquisition*,5(2):199-220,1993.http://ksl-web.stanford.edu/KSL_Abstracts/KSL-92-71.html
5. Mapquest, http://www.mapquest.com
6. Web Ontology Working Group. http://www.w3.org/2001/sw/WebOnt/
7. Resource Description Framework (RDF). http://www.w3.org/RDF/
8. Language and Computing. http://www.landc.be
9. Das, Aseem; Wei Wu; and Deborah L. McGuinness. 2001. Industrial Strength Ontology Management. Stanford Knowledge Systems Laboratory Technical Report KSL-01-09 2001. In the Proceedings of the International Semantic Web Working Symposium. Stanford, CA, July 2001. http://www.ksl.stanford.edu/people/dlm/papers/ontologyBuilderVerticalNet-abstract.html. Also published in: Isabel Cruz, Stefan Decker, Jerome Euzenat, and Deborah L. McGuinness, eds. The Emerging Semantic Web. IOS Press, 2002.C. Batini, M. Lenzerini and S.B. Navathe, “A Comparative Analysis of Methodologies for Database Schema Integration,” *ACM Computing Surveys*, 18, 4, December 1986, pp. 323-364
10. http://www.w3.org/2002/ws/desc/
11. DAML-S Technical Overview. http://www.daml.org/services/daml-s/0.7/

This paper was written under the support of Air Force ESC Contract FA 8721-04-C-001