Semantic Queries and Mediation in a RESTful Architecture

John Gilman - Dir. Architecture and Engineering
Blue Shield of California

Matt Fisher - Sr. Systems Engineer
Progeny Systems

Mike Dean - Principle Engineer
BBN Technologies
Solves many-to-many, point-to-point problem for data with an enterprise vocabulary
Loose coupling myth of Web services

- If the information consumer has to know the API and also mediate the metadata disparities, **they are still tightly coupled to the back end.**
Consumers of Info now need to know way too much about the back-end applications and data stores

- Consumer: “I don’t want to have to know any application’s API, or where data is stored, or to know the rules to derive info. I just want to either ask you for info, or tell you info.”
- When a consumer has to know any proprietary info about the “back end”, then you do not have true loose coupling.
- Having consumer interpret or derive data is a data quality risk.
- Consumer knowledge about back-end is also a security risk.
Problem - Scattered Integration Mapping, Rules = Brittle = $$$$$

- Consumer App
- ESB / Message Broker
- Backend App
- ETL Tools
- Reporting App

Rules for business, semantics, transformation, mapping
Semantic Layer

- Provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries.
- Develop and enforce common understanding of enterprise concepts and their relationships.
- Common understanding is constructed into an Enterprise Domain Model.
- Make domain knowledge explicit, instead of being buried in code.
- Enables reuse of domain knowledge (including rules) – supports derived concepts (such as ActivePharmacyMembers).
- OWL is W3C standard Semantic Web technology.
- Everything is expressed in XML – Domain Model and Rules.
Semantic Layer

Presentation Layer
- SPARQL Queries
- REST Queries

Semantic Layer
- Derived Concepts
- ECA Rules
- Classification
- Semantic Mediation

Operational Data
- Raw Concepts
- Member, Provider, etc.
Problem with Alternative - Point-to-Point Lead Pipes
Asio™

Semantic Query Decomposition (SQD)

1. Query: SPARQL
2. Query Decomposition
3. Generation of Sub Queries
4. Data Access
5. Backwards Rule Chaining
6. Query Result Set

Automapper

Semantic Bridge Database

Data Source Ontology

Data Source Ontology

Semantic Bridge Web Service

WSDL Ontology

WSDL Ontology

Semantic Bridge SPARQL Endpoint

RDBMS

SOAP WS

KB

Snoggle

SWRL Rules

Domain Source Ontology

OWL Platform

OWL Query

Decomposition

SPARQL Endpoint

Semantic Bridge

Mapping Ontology

Semantic Query Decomposition (SQD)

2008 Semantic Technology Conference
May 18-22 — San Jose, California
## ARGUS Member Report Flow

<table>
<thead>
<tr>
<th>Daily Member</th>
</tr>
</thead>
</table>
| **SPARQL Query**     | argus-member-query.xml  
| **SQL Queries**      | RTMS/SQL + member-urls  
| **SPARQL Results**   | RTMS + member-urls  
| **Argus Transform**  | argus-member.xsl  
| **Argus Extract**    | ARGUS + member-urls  

![Flow Diagram](chart.png)
NetKernel uses pipelining

**Yahoo! Pipes**
NetKernel Architecture

Indirect Addressing

NetKernel Module

Fulcrum Module
Controlled vocabulary for rules

- Rules and ontologies (container of canonical model) work together.
- Ontologies were originally built to address rules getting out of control (i.e. you can ask what rules contain a certain concept).
- Must know what rules are based on. Before you test a variable in a rule, you must first have confidence in what it is you are testing.
  - “If Employee, then doSomething”
  - What is meant by Employee?
- Rules are composed utilizing concepts from the ontology.
Taxonomy of model allows generalized rules – hierarchy is fed to rules engine

Rule 21:
If Person
then do This
Combining OWA and CWA in with Canonical Model
The Enterprise Immune System

Msg. from Bus

DL Reasoner

Rules Engine (SWRL)

React

S/OX + Finance Msg.

Classify (identify)

ACTION

Log Msg. + Route to Finance
Concepts are identified by URI’s - Uniform Resource Identifier

- A Uniform Resource Identifier (URI) is a compact sequence of characters that identifies an abstract or physical resource.
- Web Pages (physical resource) are addressed using URI's:
  https://www.blueshieldca.com/bsc/home/home.html

- URI’s are also used to uniquely identify non-physical (abstract) resources:
  http://www.blueshieldca.com/canonical#ActivePharmacyMember

- This is how standards bodies reserve words for XML (namespaces).
- URI’s uniquely label concepts and properties. Important for information sharing where concepts have the same labels.
- Used for data Federation and data lineage
  http://www.blueshieldca.com.com/rtms#Member
  http://www. www.blueshieldca.com/facets#Member
Graph Structure in RDF

Kevin Bacon has_name "Kevin Bacon"

kevin_bacon

starred_in a_few_good_men

jack_nicholson

starred_in a_few_good_men

starred_in the_departed

starred_in the_color_of_money

tom_cruise

starred_in the_color_of_money

the_color_of_money

directed_by martin_scorsese

starred_in the_departed

starred_in the_color_of_money

directed_by martin_scorsese
Theory: All knowledge can be represented in the form of statements

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>kevin_bacon</td>
<td>starred_in</td>
<td>a_few_good_men</td>
</tr>
<tr>
<td>jack_nicholson</td>
<td>starred_in</td>
<td>the_departed</td>
</tr>
<tr>
<td>martin_scorsese</td>
<td>directed_by</td>
<td>the_color_of_money</td>
</tr>
</tbody>
</table>
### SPARQL Query with URI’s, based on triples

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>kevin_bacon</td>
<td>starred_in</td>
<td>a_few_good_men</td>
</tr>
<tr>
<td>jack_nicholson</td>
<td>starred_in</td>
<td>the_departed</td>
</tr>
<tr>
<td>martin_scorsese</td>
<td>directed_by</td>
<td>the_color_of_money</td>
</tr>
</tbody>
</table>

PREFIX hwd: <http://www.hollywood.com/3.0#>

SELECT ?subject ?predicate ?object
WHERE { ?subject hwd:starred_in hwd:a_few_good_men .}
The relationship finder allows to explore the DBpedia dataset to find relationships between two things. Enter two things in the following form to find out how they are related:

**First Object:**
- Kevin_Bacon

**Second Object:**
- Martin_Scorsese

**Results:**

<table>
<thead>
<tr>
<th>Distance</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td><img src="" alt="Result 6" /></td>
</tr>
<tr>
<td>4</td>
<td><img src="" alt="Result 10" /></td>
</tr>
</tbody>
</table>

Would you like to save your query?
Goals of Semantic Integration

- To share a common understanding of concepts in the enterprise*
  - Machines and humans use same model
- To enable reuse of domain/business knowledge*
  - Very difficult if point-to-point data resolution is practiced
- To make domain assumptions explicit*
  - Takes a great deal of risk out of integration
- To separate domain knowledge from operational knowledge*
  - Best practice for SOA – business rules should be separate from app components wherever possible
  - KNOWLEDGE reuse, not just software reuse – hard to reuse software when it is mixed with proprietary domain knowledge
- To establish a single canonical information model for the enterprise
  - Logically bridge proprietary domain models
  - Easy data lineage and impact analysis

*from Ontology 101: A guide to creating your first ontology
Questions?