Web and Semantic Web Technologies in Argos

Jose Luis Ambite
USC/Information Sciences Institute

http://www.isi.edu/~argos/
Argos Team

- School of Policy, Planning and Development
  - Prof. Genevieve Giuliano
  - Prof. Peter Gordon
  - Lanlan Wang
- Texas Southern University
  - Prof. Qisheng Pan
- Information Sciences Institute
  - Dr. Jose Luis Ambite
  - Naqeeb Abbasi
- Alumni
  - Prof. Stefan Decker (USC/ISI & DERI)
  - Andreas Harth (DERI)
  - Karan Jassar
  - Matthew Weathers
- Government partners
  - California Department of Transportation
  - Southern California Association of Governments
  - Los Angeles County Metropolitan Transportation Authority
  - Other local agencies
Class overview

Discuss web and semantic web technologies in context of ongoing research project on automatic workflow generation (Argos)

- Ontology modeling with Protégé
- RDF/RDFS
- Triple: Logic for RDF/S
- Web services
  - WSDL
  - BPEL4WS
Argos: Research Objectives

- **Computer Science research:**
  - Model scientific problems as computational workflows
  - Techniques for dynamically composing web services

- **Digital government application:**
  - Intra-metropolitan freight flow model using web services
  - Test application in cooperation with government partners

- **Social sciences research:**
  - Goods movement-based accessibility measures
  - Advances in urban theory and modeling
  - Accessibility impacts on employment concentrations and land values

- **Interdisciplinary research:**
  - Flexible data framework for exploring other problems, e.g. regional accounts
Argos: Automatic Generation of Computational Workflows

- Scientific problems modeled as computational workflows
- Operations:
  - information gathering
  - data processing
- Uniform access: web services
- Goal: Automatic workflow generation in response to user requests
Modeling and Automatic Composition

- Model the domain => Ontology
- Model source contents
- Model processing operations
- Automatically Compose Workflows
- Execute compositions

\[ \{ \text{RDF/RDFS Protégé} \} \]
\[ \{ \text{Triple} \} \]
\[ \{ \text{BPEL4WS WSDL} \} \]
Modeling the application domain: Argos Ontology

- Application domain:
  - Transportation, Urban Planning
  - Typical of many economic modeling problems
  - Time series data
  - Hierarchical classifications: industries, commodities, regions, ...

=> Virtual datacube

- Hierarchical dimensions
- Part-of semantics
Argos Ontology (1)

- Central concept: Measurement
- Dimensions:
  - Geo: geospatial entity
    - Ex: LACMSA, TAZ, Census Tract, Highway, ...
  - Time Interval:
    - Ex: 1997, June2000, 2003Q1, 2005-02-15, ...
  - Product: Commodity, industry classifications
    - Ex: SCTG, SIC, NAICS, ...
  - Flow: product movement
  - Unit: M$, metric tons, short tons, ...

“Agricultural exports from CA in June 1999 in metric tons”
“Gasoline imports of LA CSMA in 2003 in M$”
Argos Ontology (2)

- Domain can be represented with:
  - Resource Description Framework (RDF)
  - RDF Schema (RDFS)
  - So far, no need for more complex logics (OWL)

- Protégé ontology editor
  - Facilitates knowledge acquisition
  - Can output RDF/S, OWL, …
  - Extensible plugin architecture
Protégé

- Demo:
  - Argos Ontology
  - RDF, RDFS outputs
Product Dimension

- 01-05 Agricultural products and fish
  - 01 Live animals and live fish
  - 02 Cereal grains
  - 03 Agricultural products, except live animals, cereal grains and forage products
  - 04 Animal feed and feed ingredients, cereal, straw, and eggs and other products of animal origin, n.e.c.
  - 05 Meat, fish, seafood, and preparations
- 06-09 Grains, alcohol, and tobacco products ...
Simplified Argos Ontology

(a) Flow dimension

(b) Time dimension

(c) Product dimension

Figure 3: Sample dimension hierarchies
Modeling Sources

- Define data descriptors
  - objects in Argos Ontology
    (so far, propositional view)

<table>
<thead>
<tr>
<th>Source</th>
<th>Flow</th>
<th>Product</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>imports</td>
<td>iron</td>
<td>2002-2004</td>
</tr>
<tr>
<td>S2</td>
<td>exports</td>
<td>iron</td>
<td>2002-2004</td>
</tr>
<tr>
<td>S3</td>
<td>imports, exports</td>
<td>iron</td>
<td>2000</td>
</tr>
<tr>
<td>S4</td>
<td>imports, exports</td>
<td>iron, uranium</td>
<td>2000, 2001</td>
</tr>
<tr>
<td>S5</td>
<td>imports, exports</td>
<td>cereals</td>
<td>allYears</td>
</tr>
<tr>
<td>S6</td>
<td>imports, exports</td>
<td>iron, uranium, metals</td>
<td>allYears</td>
</tr>
</tbody>
</table>

source(s1,1)[flow->imports, product->iron, time->2000]. ...
source(s2,1)[flow->exports, product->iron, time->2000]. ...
source(s3,1)[flow->imports, product->iron, time->2000].
source(s3,2)[flow->exports, product->iron, time->2000].
source(s4,1)[flow->imports, product->uranium, time->allYears]. ...
source(s5,1)[flow->imports, product->cereals, time->allYears].
source(s5,2)[flow->exports, product->cereals, time->allYears].
source(s6,1)[flow->imports, product->uranium, time->allYears].
source(s6,2)[flow->exports, product->uranium, time->allYears]. ...
Rule language for the Semantic Web
- query, inference, and transformation language for RDF
- expressive bodies (full FOL syntax)
- based on F-Logic [Kifer (object-oriented]

Native support for:
- Namespaces & resources abbreviations
- Models (sets of RDF statements)
- Reification

Triple tutorial: http://triple.semanticweb.org/
Modeling operations

- Describe operations by input/output signature
  - I/O just data descriptors
  - Otherwise the operation is a blackbox
- Consistent with web service implementation for operations
- Described as Triple rules
Hierarchical Aggregation operation in Triple

- Aggregation along hierarchy dimensions common in our domain
  - If there is no source for a given element in a dimension hierarchy, then compute by finding out available children (recursively)
  - Typical of operations that compute inputs dynamically

- Triple rule:
  FORALL O, SF, F, P, Y
  data(SF,P,Y)[flow->SF,product->P,time->Y] <-
  operation(opSumFlow,SF,P,Y) AND
  (FORALL X ( (NOT ( F[argos:parent->SF] )) OR
    (O[flow->F,product->P, time->Y]))).

\[ a \rightarrow b \equiv \neg a \lor b \]
Automatically Composing Workflows

- Load into Triple logic engine:
  - Domain ontology
  - Source descriptions
  - Operation descriptions
- Ask user query
- Logic program computes workflow graph
Sample workflow

<table>
<thead>
<tr>
<th>Source</th>
<th>Flow</th>
<th>Product</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>imports</td>
<td>iron</td>
<td>2002-2004</td>
</tr>
<tr>
<td>S2</td>
<td>exports</td>
<td>iron</td>
<td>2002-2004</td>
</tr>
<tr>
<td>S3</td>
<td>imports, exports</td>
<td>iron</td>
<td>2000</td>
</tr>
<tr>
<td>S4</td>
<td>imports, exports</td>
<td>iron, uranium</td>
<td>2000, 2001</td>
</tr>
<tr>
<td>S5</td>
<td>imports, exports</td>
<td>cereals</td>
<td>allYears</td>
</tr>
<tr>
<td>S6</td>
<td>imports, exports</td>
<td>iron, uranium, metals</td>
<td>allYears</td>
</tr>
</tbody>
</table>

Time | Product          | Flow     | Source
--- | ----------------- |----------|----------
2002-2004 | iron             | imports  | S1
2002-2004 | iron             | exports  | S2
2000 | iron             | imports, exports | S3
2000, 2001 | iron, uranium   | imports, exports | S4
allYears | cereals          | imports, exports | S5
allYears | iron, uranium, metals | imports, exports | S6

[Diagram showing flow of data with sources and destinations labeled]
Execution Architecture

- Workflow graph translated to the Business Process Execution Language for Web Services (BPEL4WS)
- Sources, operations, and compositions(!) deployed as web services (WSDL)
- Web services exchange RDF data
- Web service implementation
  - Any program (Java)
  - RDF processor (Jena)
  - Triple engine
Execution Architecture:
Sample BPEL4WS

<process name="argos"
  targetNamespace="urn:argos:process:osp1"
  xmlns:tns="urn:argos:process:osp1"
  xmlns:s5="http://localhost:8080/axis/services/S5"
  xmlns:s6="http://localhost:8080/axis/services/S6"
  xmlns:osp1="http://localhost:8080/axis/services/OSP1"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process"/>
<variables>
  <variable name="request" messageType="tns:request"/>
  <variable name="response" messageType="tns:response"/>
  <variable name="s5in" messageType="s5:s5Request"/>
  <variable name="s5out" messageType="s5:s5Response"/>
  <variable name="s6in" messageType="s6:s6Request"/>
  <variable name="s6out" messageType="s6:s6Response"/>
  <variable name="osp1in" messageType="osp1:osp1Request"/>
  <variable name="osp1out" messageType="osp1:osp1Response"/>
</variables>
<partnerLinks>
  <partnerLink name="caller" partnerLinkType="tns:OSP1_PLT"/>
  <partnerLink name="source5" partnerLinkType="s5:S5"/>
  <partnerLink name="source6" partnerLinkType="s6:S6"/>
  <partnerLink name="osp1" partnerLinkType="osp1:OSP1"/>
</partnerLinks>
Execution Architecture:
Sample BPEL4WS process

```xml
<sequence>
  <receive name="receive" partnerLink="caller"
    operation="osp1" variable="request"
    portType="tns:OSP1_PT" createInstance="yes"/>
  <flow>
    <sequence>
      <assign> <copy>
        <from variable="request" part="rdf_data_in"/>
        <to variable="s5in" part="rdf_data_s5_in"/>
      </copy> </assign>
      <invoke name="invoke" partnerLink="source5"
        operation="S5" portType="s5:S5"
        inputVariable="s5in" outputVariable="s5out"/>
    </sequence>
    <sequence>
      <assign> <copy>
        <from variable="request" part="rdf_data_in"/>
        <to variable="s6in" part="rdf_data_s6_in"/>
      </copy> </assign>
      <invoke name="invoke" partnerLink="source6"
        operation="S6" portType="s6:S6"
        inputVariable="s6in" outputVariable="s6out"/>
    </sequence>
  </flow>
  <assign>
    <copy>
      <from variable="s5out" part="S5Return"/>
      <to variable="osp1in" part="rdf_data_s5"/>
    </copy>
    <assign>
      <copy>
        <from variable="s6out" part="S6Return"/>
        <to variable="osp1in" part="rdf_data_s6"/>
      </copy> </assign>
      <invoke name="invoke" partnerLink="osp1"
        operation="OSP1" portType="osp1:OSP1"
        inputVariable="osp1in"
        outputVariable="osp1out"/>
    </assign>
    <copy>
      <from variable="osp1out" part="rdf_data_out_osp1"/>
      <to variable="response" part="rdf_data_out"/>
    </copy>
    <reply name="reply" partnerLink="caller"
      operation="osp1" portType="tns:OSP1_PT"
      variable="response"/>
  </sequence>
</process>
```
Execution Architecture: Sample WSDL process

<definitions targetNamespace="urn:argos:process:osp1"
    xmlns:tns="urn:argos:process:osp1"
    xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/03/partner-link/
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
    <message name="request">
        <part name="rdf_data_in" type="xsd:string"/>
    </message>
    <message name="response">
        <part name="rdf_data_out" type="xsd:string"/>
    </message>
    <portType name="OSP1_PT">
        <operation name="osp1">
            <input message="tns:request"/>
            <output message="tns:response"/>
        </operation>
    </portType>
    <plnk:partnerLinkType name="OSP1_PLT">
        <plnk:role name="caller">
            <plnk:portType name="tns:OSP1_PT"/>
        </plnk:role>
    </plnk:partnerLinkType>
    <service name="OSP1">
        </service>
</definitions>
Experiments

Workflow with:
- 12 operations (18 descriptors)
- 10000 additional sources (30000 descriptors)
- < 45 seconds

Workflow with:
- 256 sources and
- 511 operations
- ~ 100 seconds
Conclusion

- Scientific problems modeled as computational workflows with information gathering and data processing operations
- Model data using domain ontology
  - Multi-dimensional datacube
  - Hierarchical values: subclass, part-of
- Describe sources and operations
  - Uniform data representation: RDF/S
  - Uniform access: Web Services
  - Formalize semantics in Triple
- Automatically compose workflow
- Execute compositions

\[\text{RDF/RDFS} \quad \text{Protégé} \quad \text{Triple} \quad \text{BPEL4WS} \quad \text{WSDL}\]
Future/On-going work

- Generalize source descriptions
- Implement cost optimizer:
  - Select best workflow from workflow graph space
- Model more sources and operations