Ontology Modelling for Semantic Web-driven Application

Iliya Georgiev

Abstract: *The paper describes a bottom-up approach of ontology modeling to support easy prototyping of n-tier system, which combines legacy software components and web services. The created ontologies and web service prototypes were practically implemented in an n-tier intercontinental tracking system that enables data and application interoperability.*

Key words: *web ontology, lightweight semantic web services*

INTRODUCTION

Semantic Web and Ontologies. Several open and industry standards strengthen the web services as a mature technology [1, 10]. The information is presented in XML self-describing text. The messages are packed according to the SOAP specification that defines a message structure. The web services describe themselves using WSDL defining messages that they accept and produce. Web services support their own discovery by UDDI that is implemented as a separate web service to provide the design-time and run-time identification of web services.

The composition of web services requires the existing services to be encoded in a semantic form, such as ontology [11]. Ontology modeling could be divided into two steps: interface parameters modeling and Web services modeling. In the first step, the Web services' input/output parameters are represented in ontology with classes, and transformed to properties, defining relations between classes and properties. Then one can judge Web services' relations based on relations between properties transformed from these Web services and estimate whether these Web services can be composed.

Motivation. With the increasing maturity of semantic web technologies, the designers of an-tier web-driven system will need background to develop ontologies, to incorporate ontology standards and languages, and to use tools that may be useful to develop ontologies. The motivation for this research is to provide an application perspective on enterprise integration based on the meaning of the data.

Contribution of the paper. The paper focuses on ontology modeling to support easy prototyping of web services in a distributed system. We have concentrated the prototyping to web service ontology creating lightweight semantic services that activate the software agents. In this paper, we describe: a semantic model of web services based on web ontology; practical results of the development of an n-tier tracking system.

MODELING OF SEMANTIC WEB SERVICE

Semantic Web service model. The manipulating of web services by software agents is the semantic web objective [11], where the complex tasks are formulated in form of processes. Semantically properties of the web processes are expressed in form of ontologies. Ontology is a document or a file that formally defines relations among terms. In the web service model, ontologies consist of hierarchical definitions of important concepts and description of the properties of each concept. The ontologies can be defined in DAML-OIL or OWL [2, 9]. The latter is an ontology language describing the primitives in XML and RDF/RDF Schema. The services are represented as classes (concepts). Knowledge about a service has two important classes: service profile and service model. A service profile is a class that describes what a service requires and what it provides. A service model is a class that describes properties that concern the service implementation.

During the design of the tracking system we put two *objectives* for prototyping the web services: the structuring of the services to be described as ontology concepts; the implementation of the services to follow the development specifics aiming simplicity and performance. The objectives are contraversial: fully using ontology languages is a tedious task and could increase the size of the application software and development time.

Combining conventional software agents (processes) and web services. The designed tracking system combines several software components that have been traditionally used for years separately: container scanning and geographically recognition, distributed database, GUI, repair service, rent contracts, accounting, etc. Software agents will need to talk to web services that were built by different organizations for use in unanticipated contexts. Without a way to discover the meaning of terms and relationships, the functionality of an n-tier dynamic adaptable systems cannot be reached.

Semantic web technology allows a developer to mark up a document, sensor, database schema, or legacy software interface by linking concepts (i.e., classes, relationships, instances) to other concepts defined in ontology. Each concept is referenced by means of a unique Uniform Resource Identifier (URI). The working idea of the tracking system is to mark up various conventional documents and database tables (using OWL), so that software agents can interpret and reason with the information as shown in Fig.1.

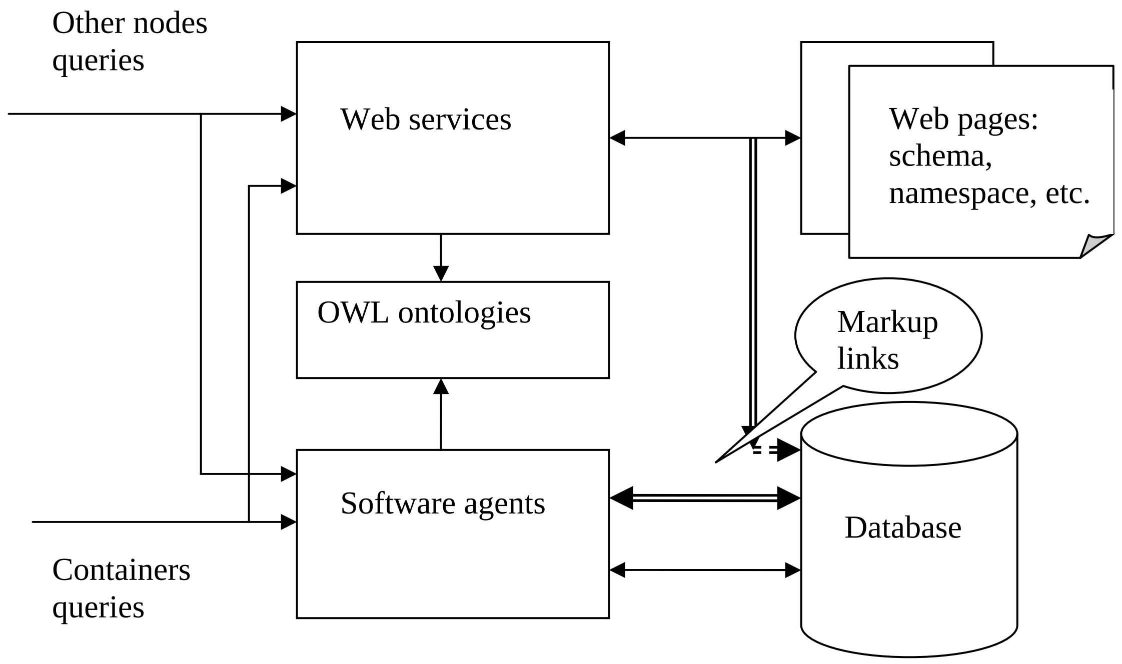

Figure 1. Interoperability between conventional processes + information via OWL ontologies

Creating ontologies. Ontologies are the mechanism that we have selected to provide semantic markup of the data sources and semantic linking between the semantic markups. Our first idea was to develop the ontologies in a top-down fashion. We started to build ontology of the domain of a virtual transportation model, initially at strong idealization of the data sources. The reason to try this approach over the bottom-up approach was we were eager to design a global model of the distributed web-driven system with new specification of the architecture, the node and data sources models. It was a good research activity, but isolated from the real time schedule of the project implementation. During the real project, we quickly realized discrepancies in the representation of essential concepts and fuzziness in the domain ontology, which would have direct implementation on the success of the integration of the existing components. Then we accepted the bottom-up approach, creating markup of the web services, data sources and interfaces to the software agents. The layers of the ontology development are illustrated in Figure 2.

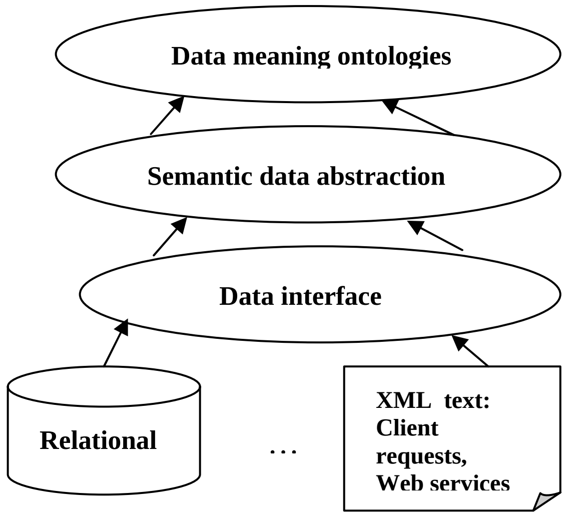
Data meaning ontologies
Semantic data abstraction
Data interface
Relational
. . .
XML text: Client requests, Web services

Figure 2 Layers of ontology design

Distributing the ontology development on separate ontologies, we mapped the ontology structure closed to the data and interfaces representation. This mapping was pretty natural and useful, as the ontology provided a cleaner interface of the data source. At that point, the semantic of the data were not yet exposed. The ontologies close to the information sources are the first layer. They serve as a common interface to extract instance data from the respective data sources. Creating such ontologies is extreme intellectual activity, but such approach has two strategic benefits. First, it eliminates the heterogeneity of the data access mechanism to the underlying data sources. Second, such ontologies are the platform to create lightweight semantically related web services processes. The second layer of the ontology hierarchy abstracts the information into more wide-ranging concepts. Lower level concepts can be associated into these higher level classes. Further, the third level of the ontology hierarchy provides high level of semantics and allows defining the main characteristics of the domain. This layer could represent some meaning elements of the domain.

Our project has been based on commercial end-user software components that are not semantically-enabled. The practical approach that we have explored was to create the lower level ontologies and define semantically rich queries to the data. Step by step we plan to combine these ontologies into more abstract concepts.

Ontology example. In the following example, we will use the current Web Ontology Language (OWL) to illustrate the concepts of the lightweight service description. OWL contains modeling concepts similar to class diagrams in the Unified Modeling Language (UML) plus other modeling formalisms that originated in artificial intelligence knowledge representation languages [9].

The first 4 lines declare the XML namespaces followed by an ontology definition that states that this ontology imports concepts from another ontology. This is followed by a series of class definitions. In OWL, there are object properties and datatype properties. Object properties represent class-to-class relationships. Datatype properties relate classes to things like string and integer attributes.

Next statements declare the namespaces of a container terminal ontology.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd ="http://www.w3.org/2000/10/XMLSchema#">
```

The header component follows.

```
   <owl:Ontology rdf:about="http://www.ctd-example.com/terminal">
```

```
      <rdfs:comment>Ontology of the Terminal class in tracking system</rdfs:comment>
      <owl:imports   rdf:resource="http://www.ctd-example.com/terminal.owl">
      <rdfs:label>TerminalOntology</rdfs:label>
     </owl:Ontology>
```

The statements below define classes, each of which is given with a name. The *BarCodeScanner* is declared as a subclass of the class *Terminal.* The classes *Terminal* and *Company* have axiom (simple) definitions. The object restriction on the C*ontainer* class states that only a company can be in a c*ontainerToCompany* relationship with a container. The class *Status* is defined via direct enumeration of its members by using *owl:oneof* construct.

```
<owl:Class rdf:ID="Terminal"/>
<owl:Class rdf:ID="BarCodeScanner">
   <owl:subClassOf rdf:resource="#Terminal"/>
</owl:Class>
<owl:Class rdf:ID="Company"/>
<owl:Class rdf:ID="Container">
  <rdfs:subClassOf> rdf:resource="#Company"/>
 <rdfs:subClassOf>
 <owl:Restriction>
    <owl:onProperty rdf:resource="# containerToCompany "/>
    <owl:toClass rdf:resource="#Company"/>
  </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Status">
   <owl:subClassOf rdf:resource="#Container"/>
   <owl:oneof rdf:parseType="Collection">
      <owl:Thing  rdf:about="Intact"/>
      <owl:Thing  rdf:about="Damaged"/>
      <owl:Thing  rdf:about="Repair"/>
      <owl:Thing  rdf:about="POS"/>
   </owl:oneof
</owl:Class>
```

The next statements define some object properties and a datatype property.

```
<owl:ObjectProperty rdf:ID="terminalToLocation">
    <rdfs:range  rdf:resource="#Location">
</owl:ObjectProperty
<owl:ObjectProperty rdf:ID=" containerToOrganization "/>
<owl:DatatypeProperty rdf:ID="geographical Place">
  <rdf:datatype ="&xsd;string"/>
  <rdfs:range rdf
</owl:DatatypeProperty>
```

The next example shows markup that corresponds to the ontology above. Markup contains instances found on a web page that are linked to classes in the ontology declared in the *owl:imports* statement. For example, *ABC99999* is a *container* who has a relationship with a *company* named *TransporterGmbh* and *ABC99999* is the same person as *ABC* mentioned in the document.

```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:owl="http://www.w3.org/2002/07/owl#"
 xmlns:xsd =http://www.w3.org/2000/10/XMLSchema#
 xmlns:aac="http://www.ctdeveloper/ship/extract.owl #">
<owl:Ontology rdf:about=" http://www.ctdeveloper/ship/ ">
   <rdfs:comment>Ontology of the Ship class in tracking system</rdfs:comment>
   <owl:imports   rdf:resource=" http://www.ctdeveloper/ship/ship.owl ">
```

```
      <rdfs:label>ShipOntology</rdfs:label>
   </owl:Ontology>
  <aac:Ship rdf:about="QueenAnn">
      <owl:label>QueenAnn</owl:label>
  </aac:Ship>
  <aac:Company rdf:about="TransporterGmbh">
      <owl:label> TransporterGmbh
      </owl:label>
  </aac:Company>
  <aac:Container rdf:about="ABC99999">
      <aac: containerToOrganization  rdf:resource="QueenAnn"/>
      <owl:sameIndividualAs rdf:resource="ABC"/>
      <owl:label> ABC99999</owl:label>
   </aac:Container>
```

EXPERIMENTAL RESULTS

Tracking system development. We have implemented ontologies, which were used to create several versions of web services incorporated in tracking n-tier architecture with different subsystems: for manipulating product documentation, container scheduling and real-time tracking of auto spare-parts distribution. It tracks where the auto part containers are in the network, in what condition they are, and how they are being used. It reports statistics about the part movement and utilization. The reports are related mostly with inventories and times spent by parts in particular statuses and in transit. The reports are organized around a 2D graphical agent that shows the node network and displays relevant information on a geographical map. The 2D interface contains important totals and links to the reports for auto parts. Fig. 3 shows an example of the map and subsystem's GUI.

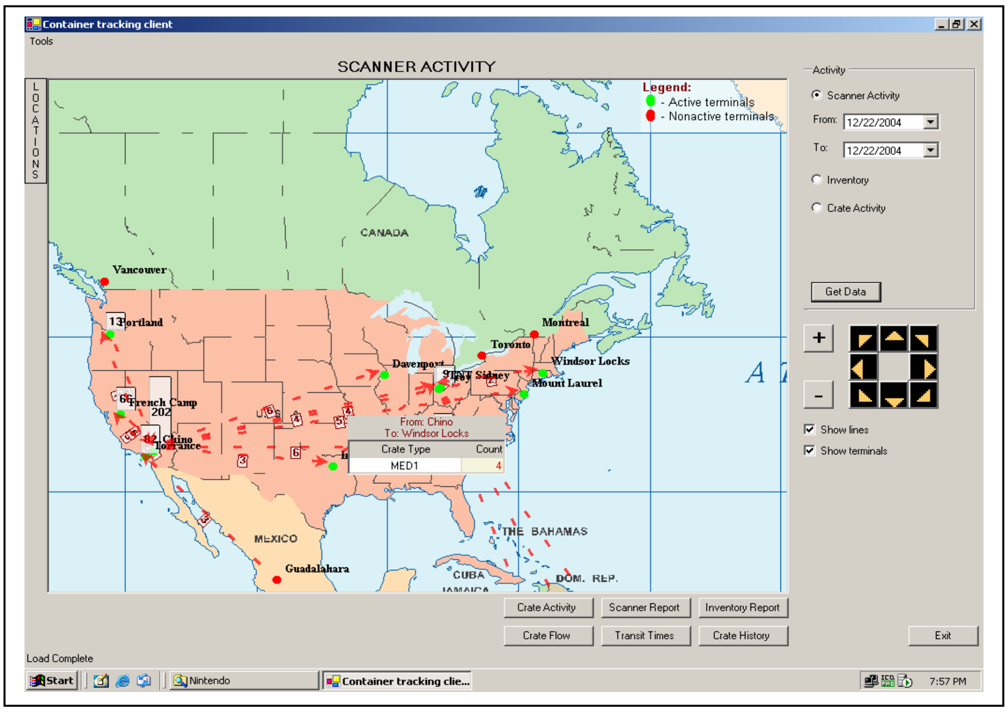

Figure 3 GUI of the developed *n*-tier system

Analysis. Leveraging web semantic model, we created a robust framework, which help designer and programmers alike more quickly from idea to implementation. In general, the opinion of the developers about our approach has been very positive.

We created a layered ontology hierarchy and found it best to build our ontologies from the bottom up. Even at such an early stage, we can address serious challenges:

- Creating ontologies is intellectual product of human beings and is a very difficult task. Ontologies that represent even just a concept become complex quickly. Ontologies evolve by default slower than human ideas, so there needs to be a systematic way to manage change. Ontological engineering requires declarative thinking and understanding of formal reasoning, and users will need to be educated in these.
- Ontological engineering is expensive. The creation of markup from unstructured text description of an innovation model is tedious and time-consuming. The good news is that there are several useful tools, which automatically could generate markup.
- Ontology mapping and translation. Simply having a multitude of ontologies does not imply that intelligent and innovative knowledge search will occur. Ontology has to be translated or mapped to others even if the domain is only slightly different.

RELATED WORK

Web services prototyping. Web service analysis is given in [1]. The role of prototyping using scripts is focused in [11].

Semantic Web technology, ontology modeling. Experience in prototyping of semantic web technology and explanation of challenges and corresponding research is given in [5]. Survey of the most relevant ontological modeling approaches is made in [4]. Several related work declare that the bottom-up method is the only practically useful: [8] shares the argumentation, [6] explicitly declares, "Targeting completeness for the domain model appears to be practically unmanageable and computationally intractable". Formal ontology framework is proposed in [7].

SUMMARY AND CONCLUSION

In this paper, we share our research results applying semantic web technology, specifically ontologies, to a web-driven application area. We proposed a mechanism to insert this technology in a legacy software subsystems and databases, creating ontology interfaces from the bottom up. Then we created well-defined prototypes of services. Finally, we experimentally evaluated each of them during the design and development of tracking system for auto parts.

In Memoriam of ***Prof. Dr. Hugo Oscar***, Technical University of Sofia. I express my eternal gratitude for his ever-insightful comments and friendly support.

REFERENCES

[1] Burner, M. 'The deliberate Revolution'. *ACM Queue,* vol. 1, No 1, pp. 29-38, 2003.

[2] DAML Initiative. 'DAML+OIL Ontology Markup'. http:/www.daml.org.

[3] D.Djuriæ, D. Gaševiæ, V. Devedžiæ. Ontology Modeling and MDA. *Journal of Object Technology*. 2005

[4] Kalinichenko, L., Missikoff, M., Schiapelli, F., and Skvorzov, N. Ontological Modeling. *Proceeding of the 5th Russian Conference on Digital Libraries*, St.-Petersburg, Russia, 2003.

[5] Kogut P., J. Heflin. Semantic Web Technologies for Aerospace. *Proceedings of IEEE Aerospace Conference*, 2003.

[6] Maedche, A., S.Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, No.1, 2001.

[7] Pahl, C., and M. Casey. Ontology Support for Web Service Processes. *Proceedings of EEEC/FSE 2003*, Helsinki, September 2003.

[8] Salim S. K.. Hetherington-Young, and S. Frey, Ontology Engineering: An Application Perspective. *The MITRE Corporation Publishing*, at http://www.mitre.org/work/tech_papers/tech_papers_04/04_0847/, 2004

[9] Smith, M., C. Welty, and D. McGuinness . OWL Web Ontology Language Guide. *W3C Recommendation,* http://www.w3.org/TR/2004/owl-guide/

[10] Vincent C. 'Scripting Web Services Prototypes'. *ACM Queue,* vol.1, No 1, pp. 22-28, 2003.

[11] W3C Semantic Web Activity. *Semantic Web Activity Statement*, 2002. http://www.w3.org/sw.

ABOUT THE AUTHOR

Iliya Georgiev, Prof. PhD, Department of Mathematical and Computer Sciences, Metro State College of Denver, Campus box 38, P.O. Box 173362, Denver 80217, USA, Phone: +303 673 9403, E-mail: gueorgil@mscd.edu, URL: http://clem.mscd.edu/~gueorgil/.