AGENTCITIES / OPENNET INPUT DOCUMENT

References to Ontology Services

Agentcities / openNet Input Document
actf-in-00020, *11 February, 2004*

Authors (alphabetically):
Manjula Patel, UKOLN, University of Bath

Status

Draft

This version: http://www.agentcities.org/in/00020/actf-in-00020a.html
Latest version: http://www.agentcities.org/in/00020/

Abstract

This document identifies various activities currently in progress in the area of providing ontology based services as an input to the work of the agentcities working group on communication. The descriptions of the services have been provided by the contacts as indicated.

Contents

1. Agentcities.RTD (BT) Ontology Server

Contacts: Nick Giles, Simon Thompson (British Telecom)

The description of the ontology server is at:
http://193.113.27.14/services/OntologyService/ServiceDescription.htm

Access point to the service itself is at:
http://193.113.27.14/ontology-server-demoV2.6/

A paper describing the server was presented at the International Semantic Web Conference 2003:
"Beyond Ontology Construction; Ontology Services as Online Knowledge Sharing Communities" Yang Li, Simon Thompson, Zhu Tan, Nick Giles and Hamid Gharib
Lecture Notes in Computer Science, Volume 2870 / 2003, September 2003, pp469 – 483
http://www.cefn.com/papers/ISWC03-LTTGG.html

2. An Ontology Server for the Agentcities.NET Project

Contacts: Manjula Patel, Monica Duke (UKOLN, University of Bath)

A web page relating to the deployment project:
http://www.ukoln.ac.uk/metadata/agentcities/

A browse interface to the server is at:
http://agentcities.ukoln.ac.uk/server/

An agentcities technical report describing the implementation of the server:
"An Ontology server for the Agentcities.NET Project"
Monica Duke and Manjula Patel, October 2003
http://www.agentcities.org/note/00008/

3. University of Technology Aachen, Germany

Contact: Karl-Heinz Krempels (University of Technology Aachen)

Implemented Features:

- ontologies are stored in a relational database (postgresql), based on Protege's data model. The reason for this is to allow continuous access via Protege's JDBC interface.
- each ontology has the attributes: ontology_name, ontology_version, ontology_language
- the following mechanism is implemented in a JADE behavior that can be attached to any (JADE) agent:
    - when such an agent receives a message with an unknown ontology, the agent requests the ontology from the ontology agent.
    - the ontology agent receives the ontology’s name, version and its representation language needed by the agent. If an ontology is found in the database it is transferred to the waiting agent via an HTTP stream

- currently the following representation languages are supported: RDF (Protege), XML (Protege), JESS (JessAgent), JavaBeans (BeanGenerator)

Karl has offered access to the javadoc output as well as the source code CVS

4. Comtec Ontology Server

Contact: Hiroki Suguri (Comtec)

A paper about the server was presented at OAS2001:
"Implementation of FIPA Ontology Service"
Hiroki Suguri, Eiichiro Kodama, Masatoshi Miyazaki, Hiroshi Nunokawa, Shoichi Noguchi
Proc. Ontologies in Agent Systems (OAS2001) at 5th International Conference on Autonomous Agents, pp. 61-68.
http://ceur-ws.org/Vol-52/oas01-suguri.pdf

The source code and additional information is available at:
http://ias.comtec.co.jp/ap/

5. Otago Ontology Repository (OOR)

Contact: Stephen Cranefield (University of Otago)

There are three components to this work:

a) the ontology repository server (and the protocol for communicating with it),
b) an RDF schema to define the types of metadata that can be stored about resources in the repository
c) an applet-based GUI client for human users to use the repository to add, retrieve and query information about ontologies in the server.

The server (a) is independent of the choice of metadata schema (b) since a generic RDF Web API for sending metadata to the server is used (in the paper use of rdftp is discussed, but now Joseki from HP Labs in Bristol is being used). In the paper, the GUI applet (c) was specialised for the particular schema shown, but now the client is generic – it reads the schema and customises the user interface to match the schema (b). This point is important to understand, as the paper showed a very simple schema, in which very only basic information about ontologies was stored, but it is straightforward to extend this to a more complex (and useful) schema.

The philosophy behind OOR is based on the following ideas:

- To avoid the need for a central ontology agent as a mediator between agents and the actual ontology repositories (as used, for example, in the FIPA ontology service specification). This should eliminate a possible bottleneck and single point of failure.
- To allow agents to communicate directly with the repository using the standard HTTP protocol (rather than requiring agent programmers to learn a lesser known and less widely implemented protocol such as OKBC or the FIPA ACL-based one that is used with the FIPA Ontology Service).
- Not to make any assumptions about or restrictions on the representation language used to express ontologies, and to allow multiple representations of ontologies to be stored (e.g. a structural representation in OWL, UML, etc. as well as a graphical one in GIF or JPEG). [This point means that there is no inference-based checking or querying of the repository, which is why it is called a "lightweight ontolology repository". It is envisaged that a plug-in architecture could be developed to provide language-specific functionality for this type of thing.]

The use of HTTP follows the REST architectural style (which is outlined in the paper cited below), and this should help to make access to the repository scalable. The repository schema/data model defines a number of classes (in RDFS) - in the paper these are Conceptualisation, Ontology and Person - and clients of the repository can use HTTP POST, PUT and GET to send or retrieve representations of these resources, each representation having an associated media type, e.g. text document (text/plain), RDF model in XML (application/xml+rdf) or a GIF image (image/gif). In addition, the schema defines the RDF properties that can be used to record metadata about resources, or relationships between them (e.g. a particular Person resource is the author of a particular Ontology resource). Resources are identified by URIs and the repository generates URNs for POSTed resources. Alternatively you can PUT a resource representation if, for example, an ontology already has a standard URI to identify it. URNs are used to identify ontologies to allow repositories to be replicated and federated easily. A primitive mechanism is provided for clients to resolve resource URNs to the URLs that are used when accessing resources from the repository using GET - more scalable mechanisms such as DDDS (see http://www.ietf.org/ids.by.wg/urn.html) are expected to become a standard part of the Internet infrastructure in the future.

As the repository schema shown in the paper is so simple, it means that ontologies stored using that schema are effectively black boxes -representations (such as their definitions in OWL or UML) can be stored and retrieved, and basic metadata (such as version and author information) can be stored using the RDF Web API. Additionally, by extending the schema so that (e.g.) classes are also modelled as resources, the RDF metadata can be used to assert relationships between ontologies and classes (ontology contains class) and between pairs of classes (class A is a subclass of Class B). At present the client (agent or human) must explicitly assert these relationships, but as mentioned before, the use of language-specific plug-ins that can analyse an ontology representation, create resources to represent the ontology's components and assert the appropriate relationships between them is envisaged.

Finally, here is a summary of changes since the paper and current work in progress:
Use of Joseki as an RDF Web API rather than rdftp (in particular, RDQL can now be used as a metadata query language); Making the GUI client independent of the repository schema.

Current work and future plans:
Making use of HTTP authentication; implementing the HTTP DEL (delete) operation; improving the user interface, including better query support; developing more complex repository schemas; developing a plug-in architecture for representation-language-specific functionality.

A paper on the work appears in:
Pan, J., Cranefield, S. and Carter, D. "A Lightweight Ontology Repository", Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems, ACM Press, 2003, pp 632-638
http://portal.acm.org/citation.cfm?id=860677&dl=ACM&coll=portal

6. Other Initiatives:

1.1 Schemaweb Directory

http://www.schemaweb.info/schema/BrowseSchema.aspx
SchemaWeb is a repository for RDF schemas expressed in the RDFS, OWL and DAML+OIL schema languages. SchemaWeb is a place for developers and designers working with RDF. It provides a comprehensive directory of RDF schemas to be browsed and searched by human agents and also an extensive set of web services to be used by RDF agents and reasoning software applications that wish to obtain real-time schema information whilst processing RDF data. RDF Schemas are the critical layer of the Semantic Web. They provide the semantic linkage that 'intelligent' software needs to extract value giving information from the raw data defined by RDF triples. SchemaWeb gathers information about schemas published on the web. SchemaWeb merges the RDF statements from all the schemas registered in the directory into an RDF triples store.
As a human user:

- Browse the schemas held in the SchemaWeb directory and inspect the details of individual schemas including classes and properties, the raw RDF/XML and the RDF triples.
- Search the schema meta-data and RDF/XML by keyword.
- Query the SchemaWeb triples store using an online form.
- Submit schemas for inclusion in the SchemaWeb directory.
- Discuss RDF and RDF schemas in the SchemaWeb forums.

As a machine user:
Query the SchemaWeb directory and triples store using the open standard web service specifications, REST and SOAP.

1.2 DAML Ontology Library

http://www.daml.org/ontologies/

1.3 Ontolingua

http://www-ksl-svc.stanford.edu:5915/

1.4 KAON (supersedes Ontoserver)

http://kaon.semanticweb.org/
KAON is an open-source ontology management infrastructure targeted for business applications. It includes a comprehensive tool suite allowing easy ontology creation and management, as well as building ontology-based applications. An important focus of KAON is on integrating traditional technologies for ontology management and application with those used in business applications, such as relational databases

1.5 IBM SNOBASE: Ontology management system

http://xml.coverpages.org/ni2003-11-03-a.html
The Java-based application provides a "framework for loading ontologies from files and via the internet and for locally creating, modifying, querying, and storing ontologies. It provides a mechanism for querying ontologies and an easy-to-use programming interface for interacting with vocabularies of standard ontology specification languages such as RDF, RDF Schema, DAML+OIL, and W3C OWL. Internally, the SNOBASE system uses an inference engine, an ontology persistent store, an ontology directory, and ontology source connectors. Applications can query against the created ontology models and the inference engine deduces the answers and returns results sets similar to JDBC (Java Data Base Connectivity) result sets. An ontology defines the terms and concepts used to describe and represent an area of knowledge. The ontology management system allows an application to manipulate and query ontology without worrying about how the ontology is stored and accessed, how queries are processed, how query results are retrieved, etc., by providing a programming interface.

1.6 ebXML Registry

A recent announcement on the www-rdf-interest mailing list indicating the formation of a Semantic Content Management SC with the intention
“to extend ebXML Registry to add direct support for publish, discovery and usage of OWL ontologies and other RDF content. The goal is to enable collaborative building of distributed knowledge bases and using these knowledge bases as metadata to describe arbitrary content.”

See: http://lists.w3.org/Archives/Public/www-rdf-interest/2004Jan/0091.html

Change Log

Version a: 11/02/2004

Page 1: Initial version

