Business and Enterprise Ontology Management with SymOntoX[1]

Michele Missikoff and Francesco Taglino

LEKS, IASI-CNR,
Viale Manzoni 30 – 00185 Rome, Italy
{missikoff, taglino}@iasi.rm.cnr.it

Abstract. Ontologies are emerging as a key solution for knowledge sharing in co-operative business environment. From the technology point of view, the growth of Internet use has favoured the development of environments devoted to collaborative and distributed work, allowing different communities to increase flexibility and effectiveness in their work. From the representation point of view, an ontology management system represents a powerful tool to create common and shareable knowledge repositories. The goal of this work is to present SymOntoX, a web-based ontology management system. It is an open source environment supporting collaborative and distributed ontology construction and maintenance.

1 Introduction

SymOntoX (Symbolic Ontology manager XML-savvy), is a software prototype for the management of domain ontologies. It has been developed by LEKS (Laboratory for Enterprise Knowledge and Systems), at IASI-CNR.

An ontology [1] [2] entails some sort of world view with respect to a given domain. It gathers a set of concepts (concerning entities, attributes, processes, etc.), together with their definitions and inter-relationships, obtained by an abstraction process (a conceptualisation [3] that starts from the observation of a given domain).

Therefore, an ontology [4] is a representation of a shared conceptualisation of a domain. Such a shared conceptualisation is necessary for establishing effective communication among actors (human or not) that operate in the domain.

An Ontology may have different degrees of formality and may be constructed using different techniques, but, necessarily, it includes a vocabulary of terms with their meaning (definitions) and their relationships. In this perspective, an ontology can be seen as a domain vocabulary containing a set of precise definitions, or axioms, that

- provide the meaning of the terms,
- enable a consistent interpretation of the terms defined in the vocabulary.

[1] This work has been partially supported by the European project IST-2000-29329 (Harmonise).

I. Horrocks and J. Hendler (Eds.): ISWC 2002, LNCS 2342, pp. 442–447, 2002.

SymOntoX has been conceived for business and enterprise ontology management. Therefore, it offers a few native meta-concepts, such as Business Process, Object, and Actor that help the enterprise experts to better categorise the identified concepts, supporting the demanding ontology building task. In this perspective, its knowledge model is more focused than existing systems, such as Protegé 2000 [5], OntoEdit [6], KSL Ontology Server [7].

Below, in Section 2 the main lines of SymOntoX representation method are represented. The functionalities, and the architecture of SymOntoX are briefly illustrated in Section 3 and 4, respectively. The conclusion reports a brief account of current experimental activities.

2 The Ontology Representation: The OPAL Methodology

In SymOntoX, domain concepts and relations are modelled according to OPAL (Object, Process, and Actor modelling Language) [8], a methodology for ontology representation developed by LEKS, at IASI-CNR, within the Harmonise European project for semantic interoperability in the tourism domain [9].

According to OPAL, concepts are organised by means of three primary modelling ideas: Actor, Processes, and Object. More precisely, we have:

Actor – any relevant entity of the domain that is able to activate or perform a process (e.g, Tourist, Travel Agency);
Object – a passive entity on which a process operates (e.g., Hotel, Flight);
Process – an activity aimed at the satisfaction of a goal (e.g., Making_a_reservation).

Besides the above primary modelling ideas, OPAL proposes the following complementary modelling ideas:
Goal – is a desired state of the affairs that an actor seeks to reach. (e.g., Go_vacation);
State – is a characteristic pattern of values that instance variables of an entity can assume. (e.g., Flight_full);
Rule – is an expression that is aimed at restraining the possible values of an instance of a concept (constraint rule) or that allow to derive new information (production rule). (e.g., "Ticket purchase 30 days before departure");
Information Component - a cluster of information pertaining to the information structure of an Actor or an Object (e.g., Flight_info, Hotel_address);
Information Element - atomic information element that is part of an Information Component (e.g., Flight_price, Nr_of_rooms);
Elementary Action - activity that represents a process component that is not further decomposable (e.g., Cancel_reservation).

The above modelling ideas are necessary for defining (unary) concepts. According to OPAL, concepts are linked together by means of a number of ontological relations, that can be seen as vertical or horizontal.
Vertical relations are: *Broader* (B), that, given a concept, relates its more general concepts; *PartOf* (Pa), and *InstaceOf* (with an evident meaning).

Horizontal relations are: *Similarity* (S), that gathers the similar concepts (with an associated similarity degree); *Predication* (Pr), that link *Information Components* and *Elements* to the current concept, and the (generic) *Relatedness* (R), to link the other related concepts.

More synthetically, in OPAL, a core (i.e., without a reference to instances and with a simplified view, without goal/state/event references) definition of a concept *c*, is represented by the following 8-tuple:

$$c = (n, k, d, B, Pa, S, Pr, R),$$

where: *n* is the label of the concept; *k* is the kind, i.e., one of the modelling ideas of OPAL (Actor, Process, …); *d* is the description, explaining the meaning of the concept, generally in natural language. Then we have the set of concept labels *B, Pa, S, Pr, R* related to *c* as reported above.

Ontological relations play a key role since they allow concepts to be inter-linked according to their semantics. The set of concepts, together with their links, forms a semantic network [1].
In table 1 we provide an example of a concept structured according to OPAL.

Table 1. The Hotel concept in OPAL (simplified)

Hotel	
Def: A building where travellers can pay lodging and meals and other services	*XML tag: <Hotel>* *Kind: Object*
Broader: Accommodation *Similar*: Guest_Farm [0.8] Bed&Breakfast [0.8] *Predication*: Hotel_Address, Hotel_Category	*PartOf*: Receptivity system *Related-object*: Restaurant *Related-actors*: H_Reservation_Service *Related-processes*: Hotel_Reserving Hotel_Room_Purchasing,
(all reported terms, except in the row below the concept label, correspond to concepts in the ontology)	

3 SymOntoX functionalities

SymOntoX is able to manage several ontologies (like a DBMS for databases). Since ist linguistic component is largely independent from the conceptual structures, it seamlessly manages multilingual implementations. Its architecture is organized in three tiers: a front-end, with a Graphical User Interface (GUI); a back-end where concepts are stored and managed; a central layer where the management logic resides.

3.1 The SymOntoX Front-End

SymOntoX supports three kinds of users: **User,** with only reading rights; **SuperUser** who has read and write capabilities (the latter subject to validation); **Ontology Master** who has the full responsibility on the ontology contents and has the task to validate the concepts proposed by the SuperUsers.

The GUI is adaptive, that is, the windows and the enabled functionalities depend on the user access rights. The SymOntoX user interface has several windows, for different functions (create, view, edit, …). There is a primary window that presents two main panels. The left panel presents the list of all the concepts in the ontology; the right panel is dedicated to the management of a single concept. The aspect of the right panel changes depending on what the user is doing (essentially, editing or viewing). The operational prototype can be accessed at http://www.symontos.org.

3.2 The SymOntoX Back-End

SymOntoX provides the standard functionalities for the management of a knowledge base: inserting, modifying and deleting concepts. Furthermore it provides a set of additional functions:

Querying capabilities, that allows one to retrieve concepts that satisfy a given search expression. The search criteria can be expressed by using a guided form.

Multimedia examples management, that allows multimedia documents to be attached to a concept, to foster an intuitive representation of concepts. It is possible to attach texts, images, videos, and VRML files.

References annotation module, that allows information sources useful for the definition of a concept to be referenced. This module provides hyper-linking functions to reach referenced electronic documents and web-sites.

The subsystem aimed at the automatic validation of an ontology, that resides in the middle tier, is currently under development.

4 Architecture and Implementation Issues

SymOntoX has been conceived to be a service available on Internet. It is mainly based on XML technology [10], to guarantee maximal flexibility, interoperability and platform-independence. Furthermore it has been developed as a client-server architecture and, in particular, as a three tier architecture (Fig. 1).

The first tier concerns the client-side that represents the GUI (Graphical User Interface). The SymOntoX client is a web browser. Portability, platform-independence and light weight of HTML, allow SymOntoX to be executed from any remote workstation.

The second tier concerns the server-side that encloses both the manager of the communication with the clients and the SymOntoX application logic. The communication between client and server is performed through HTTP the most common Inter-

net communication protocol. Data (in XML format) are retrieved from the database and transformed into a HTML page by using XSLT[11] and Java. Applying different style sheets, the produced HTML page changes depending on the user role. Then the HTML page is sent to the client.

The third tier is the storage subsystem back-end. SymOntoX is built on an XML-based DBMS (in this first implementation we decided to use Excelon [12], a commercial database system, but we are investigating other free-software solutions). At the database level, there are, for each ontology

- a database containing the concept (ontology content),
- a database containing the instances (of the above concepts),
- a log database containing the history of the activity performed by the users.

Furthermore there is a database devoted to the administration, containing the information about the existing ontologies and the registered users.

The interaction between the application server and the storage subsystem is performed through a set of Java APIs that enable the communication with the Excelon databases and XQuery [13] as data retrieval language.

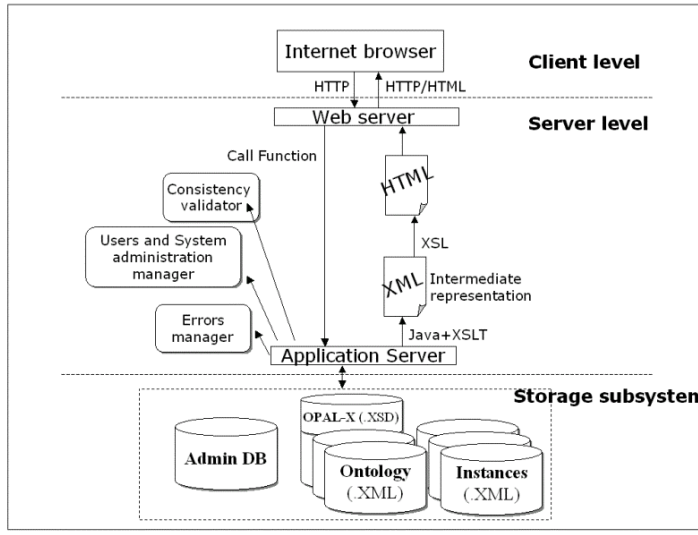

Fig. 1. SymOntoX architecture

5 Experimental Activities and Conclusions

SymOntoX in his preliminary form (called SymOntos) is currently used by a few concrete projects. Among others, worth mentioning *OntoPrivacy*, supporting the research activities of the organisation for the protection of personal information; *Business and Enterprise Ontology* (BEO), that represents the core of an ontology-based platform for Business Games; *Ontotour*, an ontology on the tourism domain.

Ontotour is currently the most advanced activity, taking place within the European project Harmonise, an FP5-IST project aiming at building an interoperability platform for SME operating in the tourism domain [9]. More than 1000 ontologies entries have been already entered in Ontotour, and the number is constantly growing.

Till now the response of the communities that are using SymOntos is very positive. Nevertheless, since SymOntoX represents a significant evolution towards the XML-technology, we expect to further improve its characteristics, expecially in the direction of flexibility and openness. In particular in the near future, a set API (Application Programming Interfaces) will be published, in order to allow external systems to directly interact with SymOntoX. These API will be based on SOAP [14], a simple and lightweight XML protocol for exchanging structured and typed information.

References

1. Genesereth, M. R., Nilsson, N. J.: Logical Foundations of Artificial Intelligence. Morgan Kaufmann Publisher Inc (1987).
2. Uschold, M., Gruninger, M.: Ontologies: Principles, Methods and Applications, The Knowledge Engineering Review, V.11, N.2, (1996).
3. Gruber, T. R.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing, Technical Report KSL 93-04, Knowledge Systems Laboratory, Stanford University (1993). http://ksl-web.stanford.edu/knowledge-sharing/papers/#onto-design.
4. Gruber T. R.: A translation approach to portable ontologies. Knowledge Acquisition 5(2), (1993), pp. 199-220. http://ksl-web.stanford.edu/knowledge-sharing/papers/#ontolingua.
5. http://protege.stanford.edu/.
6. http://ontoserver.aifb.uni-karlsruhe.de/ontoedit/.
7. http://www.ksl.stanford.edu/software/ontolingua/
8. Missikoff, M., Navigli, R., Velardi, P.: The Usable Ontology: An Environment for Building and Assessing a Domain Ontology. 1st International Semantic Web Conference (ISWC2002).
9. Missikoff, M., Callegari, G.: Preliminary Architectural Specification. Technical report within the Harmonise European project IST-2000-29329.
10. http://www.w3c.org
11. http://www.w3.org/Style/XSL/
12. http://www.exceloncorp.com
13. http://www.w3.org/XML/Query
14. Scriber, K., Stiver, M.C.: Understanding SOAP: The Authoritative Solution. Paperback (2000).