Concise Papers

Visualization of Ontologies to Specify Semantic Descriptions of Services

J. Javier Samper, Vicente R. Tomás, Eduardo Carrillo, and Rogério P.C. do Nascimento

Abstract—The present paper describes the main characteristics and components of a tool developed for integrating the definition of profiles for semantic Web Services. This tool is based on the languages DAML-S and OWL-S. It includes the ontology visualization and consistency verification which specifies the concepts that a Web Service interacts with. Starting from a service description interpreted by a computer and by the means used for accessing the service, it is possible to discover which software agents use the service. The tool can generate information in different formats to represent graphic information based on XML, such as the SVG format proposed by the W3C. The tool was developed in Java language. It is being used for the visualization of ontologies and for the semantic description of services in traffic information systems. The LISITT group (Laboratorio Integrado de Sistemas Inteligentes y Tecnologías de la información de Tráfico) developed this tool at the Robotics Institute at the University of Valencia.

Index Terms—Intelligent Web services and semantic Web, ontology design.

1 Introduction

CONTENT semantic marking and the web service capabilities will make it possible to automate a great variety of reasoning tasks. These tasks are currently being developed by human beings through manual codification, which allows a subsequent automation. However there are some questions that need to be solved: What does the service do? How can it be used? Which effects will it have? The automation of tasks such as the discovery of the service, interoperability, selection, composition, and monitoring of the execution, are the basis of Semantic Web Services [1], [2]. The international community has made an effort to look for a more intelligent Web—the Semantic Web—and to apply it for the development of Web Services. For that purpose, ontologies such as DAML-S and OWL-S [3], [4] were proposed, which are based on the semantic-marked languages DAML+OIL and OWL. Such ontologies have classes and properties that allow specifying properties and capabilities of Web Services, no matter how they will be used. OWL-S, and its previous releases DAML-S, are Web Services description languages that make part of Semantic Web technology. With conventional mechanisms, the descriptions of the services are not as expressive as the profiles of the services expressed with DAML-S/OWL-S (preconditions, postconditions, and effects cannot be expressed with present technologies).Web Services Description Language (WSDL) is an example of that. Also, the Universal Description Discovery and Integration (UDDI) does not support semantic descriptions of the services. Since there is not an explicit semantic, two identical descriptions may be different depending on the context of use. This is clearly a great disadvantage for searching and discovering services, as the client will not know which services are available on a given moment. Both providers and clients will have different perspectives and knowledge about the service [5]. Therefore, it is necessary to have a tool that is able to describe semantic services, present the ontologies used for this services description, and integrate the searching architecture of the services. The software tool presented in this paper offers these functionalities including the use of a cache manager for searching. The paper is structured as follows: Section 2 presents the state of the art of tools to describe the Semantic Web services. Section 3 presents the specific characteristics of the new tool. Section 4 details the process of ontologies visualization, including the search for services and optimization using caches. Section 5 describes the environment where the tool was used and, finally, in Section 6, the conclusions are exposed.

- *J.J. Samper is with the Instituto de Robótica, Polígono de la Coma s/n, 46980, Paterna, Valencia, Spain. E-mail: jose.j.samper@uv.es.*
- *V.R. Tomás is with the Universitat Jaume I de Castelló, Campus de Riu Sec, Edificio T1, 12071, Castellón, Spain. E-mail: vtomas@icc.uji.es.*
- *E. Carrillo is with the Autonomous University of Bucaramanga, Calle 48, 39-234, Bucaramanga, Colombia. E-mail: ecarill@unab.edu.co.*
- *R.P.C. do Nascimento is with the University of Algarve, DEEI-FCT-Gabinete Campus de Gambelas, FCT-DEEI, 8005-139, Faro, Portugal. E-mail: rnascimento@ualg.pt.*

Manuscript received 14 Feb. 2007; revised 9 July 2007; accepted 28 Sept. 2007; published online 8 Oct. 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0071-0207.
Digital Object Identifier no. 10.1109/TKDE.2007.190698.

2 State of the Art

During the research, many tools for the edition of ontologies were studied. The functionality of some of them was compared, such as DUET [8], OntoSaurus [11], OntoEdit [10], SWOOP [12], WebODE [13], WebOnto [14], Protégé [7] and OilEd [6], [9]. Important research related to some of these tools was developed by Corcho et al. [15]. The main identified problems was the lack of tools for the edition of ontologies that were allowing to generate code using upper ontologies, like OWL-S, which are necessary to specify Semantic Web Services. On the other hand, most of the OWL-S editor tools start with a WSDL description of the Web service in order to translate WSDL to OWL-S (including WDSLtoOWL-S tool). Based on the information provided by WSDL, OWL-S tool offers a framework of each one of its components. In a "posteriori" phase, the providers can complete such description, which was automatically generated. Regarding the tools for description of semantic Web Services, the main proposals found were: 1) OWL-S Editor and A-Match [16] from Carnegie-Mellon University, 2) OWLS Editor for Semantically Enabled Web-Services from the Department of Computation Sciences and Artificial Intelligence at the University of Malta [17], and 3) OWL-S Editor by SRI International [18] and different tools from the University of Maryland.

Carnegie-Mellon University has developed different tools, such as the OWL-S editor. Its purpose is the creation of different files that describe the semantic capacity of the services. It is a tool oriented to the service provider. It offers graphic visualizations and integrated reasoners. They are capable of verifying properties of the services and allow the simulation and diagnosis of such services. The reasoner that Carnegie-Mellon University uses is JTP and, unlike our reasoner, it is based on First Order Logic (FO). It allows to load the different ontologies in use and to establish the different classes and properties of the services. The possibility of specifying composed services and establishing the service data flows are some of the aspects that make this tool important.

A-Match is a Web interface of the Services matchmaker RETSINA. It was developed by Intelligent Software Agent group, at the Carnegie-Mellon University. In the multiagent system RETSINA, the services matchmaker acts like a link between the agents that request services and those that provide services. With RETSINA, the agents communicate their advertisements and service requests to the match system automatically through KQML

messages. A-Match provides an interface to human users so that they can advertise and find agents that have the capabilities needed. The prototype developed by the University of Maryland combines the description of OWL-S services with current invocations of WSDL descriptions. This allows composed services to be executed. The system can generate semantic OWL-S descriptions from WSDL descriptions. It makes both service composition and execution possible. The University of Maryland studies compatibility between outputs and inputs of the services. They claim that, in order to take a greater advantage, the outputs and inputs should be concepts rather than class types (strategy used in our tool) in established service ontologies. The University of Maryland further tackles the problem related to service composition demanded and the difficulty of establishing a complete automatism without the user's help. Therefore, the automatism can only have a partial automatism. They use service hierarchies to look for the service and, later, the edition of parameters through concepts, just like we do. In order to check inconsistency and incompatibility, they use an OWL reasoner called Pellet [19]. The University of Malta also has a complete editor for describing a Web Service in OWL-S. It includes the generation of Profile, Process, and Grounding files. The mechanism consists of providing a tool with the WSDL description of the service through an interface called OwlsWiz. In this way, the tool is the framework of the different descriptions (Profile, Process, and Grounding) and the client only has to provide nonfunctional parameters, such as the name of the service, the data of the provider, or its quality. The tool consists of three parts depending on its functionality: Creator, valuator, and visualizer. The valuator checks the ontologies through the use of Jena API. The Viewer allows the user to visualize and print the descriptions graphically. The heart of the tool is the service composer, which can manage three different types of compositions: Sequence, If-Then-Else and Split.

It can be observed that, although there is help for profile edition, there is not a direct integration with the visualization of ontologies for concept description and consistency verification with an external reasoner.

3 Characteristics of the New Tool

The first need was to combine the visualization of ontologies with the creation of search profiles. At this time, a tool based on the integration of capabilities was created as an academic experience. The tool is used for specifying semantic Web Services profiles with visualization. It is also used to verify the concepts consistency over which a given service interacts with. The specification of semantic profiles is addressed at the search of services and, therefore, it is oriented to users rather than to service providers. Thus, only the profile file is generated from a template. This implies that the files Process and Grounding are unnecessary. Since the tool can be integrated into an agent platform, a user can either store this profile on the hard disk or send the profile generated to an agent that is in charge of searching the Web Service. Regarding the management of the concept ontologies, the tool implemented includes an ontology Viewer. It may be described in the languages DAML or OWL. The need for presenting information gradually and adapted to the user's needs is an important aspect related to the graphic Viewer of ontologies. Scarce information may be insufficient while excessive information may detract from clarity of the concepts. For that purpose, three layers have been specified. They may help to activate or deactivate the visualization of instances, the dependencies between classes, the properties, and their relations with the classes. Another important aspect that distinguishes the tool presented from the existing tools is the possibility of using a cache manager for searching profiles. The use of this profile cache, which stores its information remotely, allows users to access their personal information from any place they get connected. In this way, the process for profile search is accelerated for similar queries, minimizing the network load and the processing in the agent that makes the search.

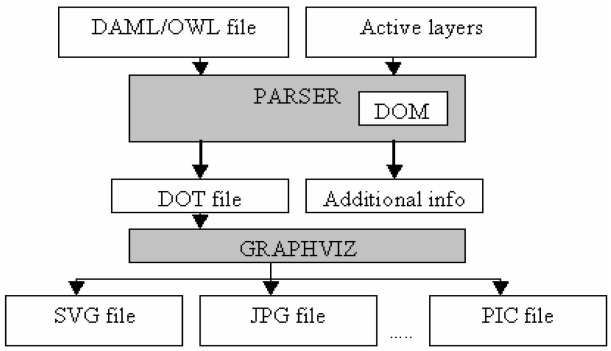

Fig. 1. Phases for graphics generation.

4 Ontologies Visualization

Visualization is carried out through a drawing area that is implemented with BATIK libraries [20]. These libraries allow SVG graphics to be generated, visualized, and edited with a JAVA program. In order to generate code in SVG from DAML or OWL code, a software called Graphviz [21] for managing opensource graphics, was used. Graphviz generates code in different formats, such as SVG, JPG, PNG, among others. It starts from a language for graph description, that allows the tool to export the ontology to multiple formats. Fig. 1 shows the steps that the application follows for generating graphics. Ontologies are loaded through two reading modes: Local files can be loaded by selecting them directly on the hard disk, although it is also possible to read a file remotely from an URL. The origin OWL or DAML file can be translated with an interpreter in DOT code. It is a plain text representation of a graph. In the translation process, unnecessary information that the user does not want to visualize (layers) is eliminated. At the same time, further information on the graphic elements is generated. Graphviz uses the file in DOT format to create code in SVG format. This file is loaded in the SVG drawing area of BATIK and it might be exported to other graphic formats. The objective of the tool is to reach the visualization of the maximum numbers of possible aspects in an ontology. For this, it is necessary to include an information filtering system so that the representation is clearer. For that purpose, three layers have been specified: 1) The first layer, labelled as "Instances," allows the possible instances or individuals of the ontology to be activated or deactivated. In the current version of the tool, it is necessary that the instances and the classes be specified in the same file. 2) The layer "Class Dependencies" visualizes the relations between classes, which include the DAML tags sameClassAs, sameOrEquivalent, equivalentClass, equivalentTo, sameAs, disjointWith, complementOf, intersectionOf, disjointUnionOf, and UnionOf as well as their equivalent tags in OWL. 3) The layer "Object Properties" shows the different properties defined and their relations with classes. In this way, the visualizer allows different elements of the ontology to be activated or deactivated. It is possible to hide elements that are not of interest, which helps to give an overview as clear as possible. When all the layers are deactivated, the class hierarchy is shown, which is expressed with the label subclassOf.

New functionalities were added to the interface. These functionalities are located in four buttons on the bottom of the interface:

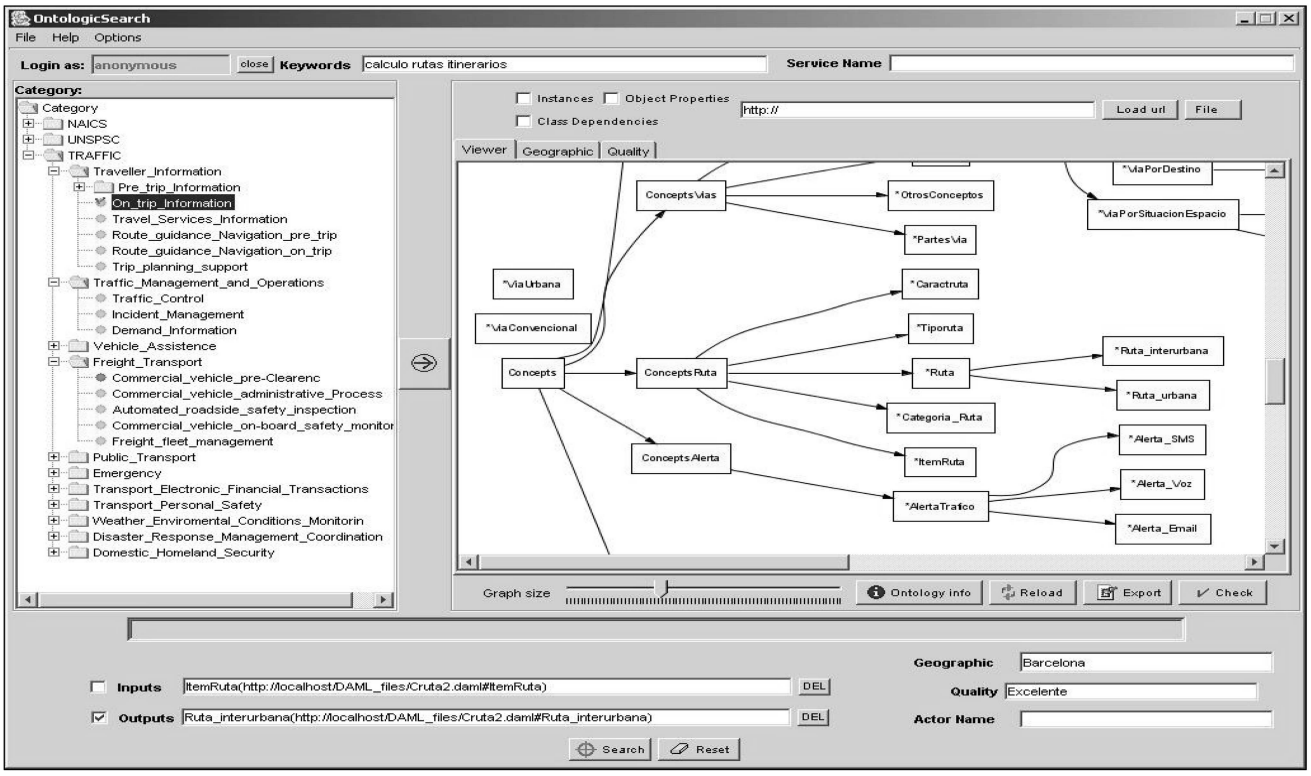

Fig. 2. General aspect of OntoService.

1. The first button is called "Ontology info," which opens a dialogue box that shows general information associated to the ontology. Such information includes the total number of classes loaded, the properties loaded, and the information contained in the label "Ontology." This last one includes the version, comments associated and listing of imports used, listing of namespaces, and classes belonging to each namespace.
2. The second button labelled as "Reload" updates the ontology visualized. It serves the purpose of using the possible changes both at source code and visualization. For example, activating or deactivating layers or changing the size of the graphic.
3. The so-called "Export" button allows us to store the visualized graphic in a given output format. The listing of available formats is GIF, JPG, PNG, SVG, PostScript, PostScript with PDF annotations, CMAP, PIC, and VRML.
4. Finally, the check button is used to search inconsistency and incoherence.

For that last purpose, the application makes use of RACER [22], an inference engine and a description logic reasoner that has been incorporated to the interface. Racer works as an application client-server. The client sends a query through HTTP to the server, which sends back the queries. When clicking the check button, three types of queries are sent to the server: 1) (daml-read-file "file") or (owl-read-file "file"), which is in charge of loading the file to be examined, 2) (classify-tbox), which checks that all the concepts have been met, and 3) (abox-consistent-p), which checks that all the instances are consistent. If everything is correct, the following messages show up: "Ok—Concepts are satisfiable," "Ok—Individuals are consistent."

On the other hand, the drawing area has associated some mouse events that provide additional information about the ontology. Classes that have comments in the file OWL or DAML using the label <comment> have associated events of the OnMouseOver type. When passing over the mouse pointer, comments show up in an information chart below. All classes have an event of the type OnMouseClick which shows additional information about the class. The user can find out about the classes that have additional information, since they have an octagonal form. If not, a description of the error found is given.

4.1 Search for Web Services

The search and call to Web Services, starting from the generation of a search profile, where parameters may be configured, are other functionalities of the tool implemented. The name of the service, keywords for the search, inputs and outputs of the service being searched, geographical localization area of the service, parameters related to the quality, or the name of the service provider are some parameters that may be configured.

The complete cycle could be summarized by the following phases:

1. A user launches a Client Agent on the platform with the semantic description of the service he is searching. The Client Agent contacts the Matchmaker Agent and informs of the service description being searched. The Matchmaker Agent asks the reasoner and contrasts the description of R service (the request), with those of all the available providers (advertisements). Then, it decides that service A is what most closely resembles that described by R and it passes the URL to the client agent.
2. Once the Client Agent dynamically knows the description that closely resembles R semantically, it generates a graphic interface to obtain the input data necessary to make a correct invocation of service A.
3. When the user fills out the form that asks for service input data, the client agent invokes it directly with that data and obtains the result that is returned.

TABLE 1
Searching Parameters of a Web Service

Field	Required	Selected Grafically
Keywords	NO	NO
Service Name	NO	NO
Inputs	NO	YES
Outputs	YES	YES
Geographic	NO	YES
Quality	NO	YES
Actor Name	NO	NO

4. Finally, the Client Agent contacts the user again and shows the result of the service invocation.

SOAP ("Simple Object Access Protocol") provides the definition of the XML-based information. This can be used for exchanging structured and typed information between peers in a decentralized, distributed environment [24]. The feature that deserves further mention is the possibility of introducing new search parameters in a text mode, or filling them in automatically by selecting the concepts in the ontology visualized in the graphic interface. Fig. 2 illustrates the main interface and a tree structure on the left. The structure shows three types of hierarchical classifications of concepts: NAICS ("North American Industry Classification System") [25], UNSPSC ("United Nations Standard Products and Services Code") [26], and TRAFFIC [27]. Going across the tree structure in any of the three classifications, it is possible to reach to a leaf node. Each leaf node has a link to the concept ontology. The interface shows information about the file access to its concept ontology. The interface shows information regarding the file access through a color codification. If a leaf node is light gray, it means that the URL can be accessed and it has information about the concept ontology. However, if it is dark gray, it means that the concept ontology cannot be accessed. Therefore, it should be loaded manually, for example, looking for it in another URL.

Table 1 shows the required or optional parameters that can be selected with the application. The field "Keywords" serves the purpose of indicating which keywords should be included in the search. All the content of the field should be included within the label <profile:textDescription> of the profile. The field "Service Name" indicates the name of the service, in case the user recognizes it. The content of the file is inserted inside a label <profile:serviceName>. The fields "inputs" and "outputs" can be filled out through the graphic interface. For that purpose, there are two charts that indicate that the corresponding field is to be filled. In this way, when clicking on a class in the ontology Viewer, it is possible to make a distinction between information for reading about a class and information for selection, such as input or output. After clicking on a class that represents a given concept, the name of the class is copied in the field selected, including its namespaces. The fields that are filled in the profile are <profile: parameter Name> with the name of the class selected and <profile:restrictedTordf:resource=X>, where X is the name of the namespaces. The field Geographic indicates the geographical scope of the service. For that purpose, it is possible to select a field by clicking on the label "Geographic." The button "Search" has two functions: 1) It searches in the cache memory for a profile similar to the current one. If so, the user informs that a similar search was made and some services were found. The user may use any of the services previously found in such a way that she/he does not have to make a similar search. The norm for comparing the profiles in memory with the one searched is to compare inputs and outputs. If all of them coincide, it is then considered that the service

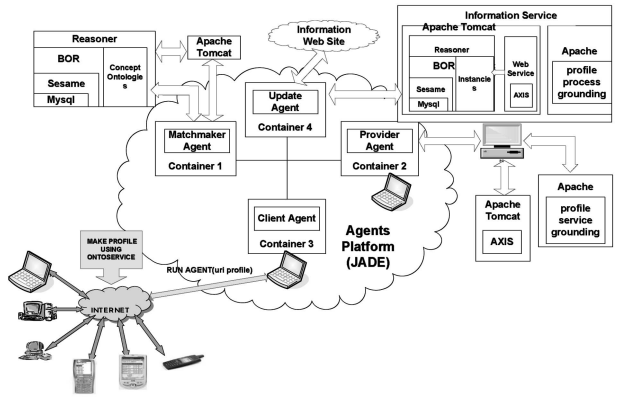

Fig. 3. Integration of the tool into the agent platform.

previously found is of interest for the user and should be shown. 2) It sends the profile generated to the agent in charge of searching the service. After sending a search profile, the agent sends back information in a SOAP message regarding which input parameters should be known to execute the Web service that receives information. When the application receives information, it should show a form which was created dynamically and included the fields that should be filled in. For that purpose, the form was generated dynamically with HTML code shown in a window with canvas for the visualization of the HTML language.

4.2 Optimization Using Caches

In a service search process, one of the hardest parts is the process of "match machine" [23], which consists of locating the services by mapping the profile generated with the profiles given by the service providers. The use of a cache that stores the latest queries sends back the services sought quickly without a search process. This increases the system speed, since the users will probably search for a reduced number of service classes. A storage architecture has been designed to allow a user to get connected with the profile server from any place and to load the latest searches made. When a user loads the tool, he/she can check in the server by opening a new account. The user inserts his/her username and password, and then presses the button "create new account." If the user has an account already, he/she should get in the application as a registered user. The option selected is then "Login." The application accesses the service profile and loads the profile that was sent back by the server. From that moment on, the profile search is made in the local cache. If it is not found, the request is made to the agent in charge of the "match machine." Taking the option of anonymous user, the function of profile cache is unauthorized and, therefore, it cannot use its functions. The agent in charge of the search profile manages any query that has been made. The implementation of the profile server was made with J2EE technology. Three servlets were implemented for three basic needs: creating new user accounts, loading, and storing profiles remotely. The exchange of information between clients and server is made through the SOAP protocol.

5 Use of OntoService

OntoService is being used in the edition process of semantic Web Services for two multiagent system (MAS) prototypes, for incidents traffic information. The first one has been ESWIT (Matchmaker of Traffic Information Web Services) oriented to facilitate, attend, and optimize the processes of advertisment, discovery, and invocation of dynamic Web Services, related to the information area on road traffic. The prototype allows users (represented by client agents) to ask matchmaker agents to search for agreed Web Services. The

final services will have similar semantic capabilities that were required. Finally, they will be invoked by users to get the best pretrip information (Fig. 3). The other MAS has been developed by Tomas and García [28], to improve the Traffic management tasks of road operators when meteorological incidents occur. It uses and validates the traffic information provided by the available information, in order to prepare traffic measures or inform drivers. The knowledge system has been originally based on traffic ontology. Thus, the distributed environment implies the need of information exchange with other external information systems. So, the ontology is being improved, with the tool Ontoservice to upgrade the knowledge model and provide a reasoning model to discover traffic information Web services.

6 Conclusions

This paper presents a summary of the state of the art of those tools that allow a description of semantic Web Services. From the analysis made, a tool was designed and implemented to combine the visualization of concept ontologies with the creation of semantic Web Services profiles. One of the most interesting capabilities of this tool is the use of a cache memory that stores the latest queries (user profiles). This cache increases the systems speed, since the users will probably search for a reduced number of types of services. In this way, the searching will not always be necessary. The tool generated was tested on two multiagent systems and, probably, it will be deployed in the real systems under the supervision of Spanish Traffic Administrations.

Acknowledgment

This research has been supported by the CICYT project with reference TRA2004-06276 MODAL ("Development of a system using a conceptual infrastructure based on ontologies, to exchange good information between trucks and external entities").

References

[1] S. Mcilraith, T.C. Son, and H. Zeng, "Semantic Web Services," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 46-53, 2001.
[2] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Semantic Matching of Web Services Capabilities," *Proc. First Int'l Semantic Web Conf. (ISWC '02)*, 2002.
[3] DAML Services, http://www.daml.org/services/owl-s/, 2007.
[4] OWL-S: Semantic Markup for Web Services, http://www.w3.org/Submission/OWL-S/, 2007.
[5] S. Narayanan and S. McIlraith, "Analysis and Simulation of Web Services," *Computer Networks*, no. 42, pp. 675-693, Elsevier Press, B.V., 2003.
[6] A. Roberts, "An Introduction to OilEd," http://oiled.man.ac.uk/tutorial/, 2007.
[7] Protégé, http://protege.stanford.edu/, 2007.
[8] DUET, http://codip.grci.com/Tools/Tools.html, 2007.
[9] OILEd, http://oiled.man.ac.uk/, 2007.
[10] OntoEdit, http://www.ontoknowledge.org/tools/ontoedit.shtml, 2007.
[11] OntoSaurus, http://www.isi.edu/isd/ontosaurus.html, 2007.
[12] SWOOP, http://www.mindswap.org/2004/SWOOP/, 2007.
[13] WebODE, http://webode.dia.fi.upm.es/WebODEWeb/index.html, 2007.
[14] WebOnto, http://webonto.open.ac.uk/, 2007.
[15] O. Corcho, M. Fernández-López, and A. Gómez-Pérez, "Methodologies, Tools and Languages for Building Ontologies. Where is Their Meeting Point?" *Data Knowledge Eng.*, vol. 46, no. 1, pp. 41-64, 2003.
[16] M. Paolucci, N. Zhendong, K.P. Sycara, C. Domashnev, S. Owens, and M. Van Velsen, "Matchmaking to Support Intelligent Agents for Portfolio Management," *Proc. 17th Nat'l Conf. Artificial Intelligence and 12th Conf. Innovative Applications of Artificial Intelligence*, pp. 1125-1126, July-Aug. 2000.
[17] J. Scicluna, C. Abela, and M. Montebello, "Visual Modelling of OWL-S Services," *Proc. IADIS Int'l Conf. WWW/Internet*, Oct. 2004.
[18] G. Denker, D. Elenius, and D. Martin, "OWL-S Editor," *Proc. Third Int'l Semantic Web Conf. (ISWC '04)*, Nov. 2004.
[19] Pellet, http://pellet.owldl.com/, 2007.
[20] BATIK, http://xml.apache.org/batik/, 2007.
[21] M. Tsoukalos, "An Introduction to Graphviz," *Linux J.*, http://www.linuxjournal.com/article/7275, 2004.
[22] V. Haarslev and R. Möller, "Racer: An OWL Reasoning Agent for the Semantic Web," *Proc. Int'l Workshop Applications, Products and Services of Web-Based Support Systems, in conjunction with the 2003 IEEE/WIC Int'l Conf. Web Intelligence*, pp. 91-95, Oct. 2003.
[23] J.J. Samper, V.R. Tomas, R.P. do Nascimento, and J.J. Martinez, "Proposal of an Architecture to Retrieve Information from Multiple Devices Using Matchmaking Algorithms," *Proc. Fourth European Conf. Universal Multiservice Networks*, pp. 12-21, Feb. 2007.
[24] SOAP, *W3C Recommendation*, http://www.w3.org/TR/soap/, Apr. 2007.
[25] NAICS, http://naicscode.com/, 2007.
[26] UNSPSC, http://www.unspsc.org/, 2007.
[27] J.J. Samper, E. Carrillo, and A. Cervera, "Semantic Modelling of Road Traffic Information Elements," Revista Colombiana de Computación ISSN 1657 - 2831, vol. 7, no 1, pp. 76-91, UNAB Colombia, June 2006.
[28] V.R. Tomás and L.A. Garcia, "A Cooperative Multiagent System for Traffic Management and Control," *Proc. Fourth Int'l Joint Conf. Autonomous Agents and Multiagent Systems (AAMAS '05)*, July 2005.