Semantic Web Services and Agents: A Reality Check

Davide Cavone
Dipartimento di Informatica
Università di Bari
70126, Bari, Italy
davide.cavone@uniba.it

Federico Bergenti
Dipartimento di Matematica
Università di Parma
43100, Parma, Italy
federico.bergenti@unipr.it

Danilo Gotta
Telecom Italia Lab
Via Reiss Romoli, 274
10148, Torino, Italy
danilo.gotta@telecomitalia.com

***Abstract*— This paper aims at analyzing the state of the art of Web services to understand how they can now play a crucial role in the landscape of agent systems, after several years of uncertainty. The study is first outlined by illustrating the main interfaces and protocols that are now emerging and by providing a list of the main repositories of Web services that are really available in the network. The results of the study clearly show that Representational State Transfer (RESTful) services are overtaking Simple Object Access Protocol (SOAP) services. Moreover, the results emphasize the lack of really effective repositories. The study also highlights the almost total absence of semantics in surveyed repositories, thus severely limiting the accurate rating of Web services. The overall judgment on the situation of Web service repositories is that it is surprisingly still immature, especially in the Italian and European landscapes. This is the reason why we decided to propose a novel Web portal meant to become an active and maintained collector of semantic Web services accessible to users (especially in Italy) and able to create a solid base for developing agent-based service systems.**

Keywords-Agent; semantic Web services; RESTful services; SOAP services

I. Introduction

The enormous potential of combining agent technology and Web Services became a certainty some years ago (see, e.g., [4, 18]). However, for several reasons, the great steps forward in the academic field did not keep their expectations in the global market. This paper aims at understanding such reasons, and in particular, at trying to find a possible solution to the impasse in the implementation and use of Web services (semantic Web services in particular). In the purse of this goal, we analyze the strengths and weaknesses of two types of Web services that are currently in competition: the classic SOAP/WSDL Web services and the so called RESTful Web services (also known as Web APIs). Despite the massive growth of latter, in reality the two approaches are not alternatives; rather, they are meant to fit the context of use: more rigorous and standardized the first, while lighter and easier to use the latter. Thereafter, our study focuses, always following a parallel trend, on analyzing the current standards to enable semantic search and publication of both types of services.

Unfortunately the results of our study reveals a still too immature situation that severely limits the wide use of semantic Web services at a business level. For this reasons, our research has led *(i)* to create a novel portal of Web services, which is useful as a collector of Web services (with or without explicit semantics) and *(ii)* to introduce a Semantic Web Services Register (SWSR) available through a FIPA-based Matchmaker Agent.

II. Agents and Semantic Web Services

It is known that the purely syntactic description of a Web service strongly limits its use both for search and for automatically combining atomic services into complex services. In particular, an agent that provides a search tool on purely syntactic services cannot customize the search according to user needs and it must also depend on a specific service and on its actual availability. For these reasons, it is of paramount importance to have explicit semantics to be published at the stage of discovery. Moreover, RESTful Web services, in comparison with traditional semantic annotations, are problematic with this regard since most Web APIs are described in free text in Web sites and they do not have machine-understandable documentation.

Nowadays the literature provides numerous studies that deal with the issue of semantic matchmaking and that accomplish important results. Our research, however, puts emphasis on the entire business semantics that the agent will perform. In particular, it stresses the importance of the phase of Testing and Select, immediately following the matchmaking phase as described in detail below, through which the agent is able to overcome problems of a specific Web service.

Generally speaking, the publication and use of semantic Web services follows a common scheme:

1. Given the services and the syntactic domain of interest, the developer chooses an ontology (e.g., in OWL [14]) to share among requester agent and service providers;
2. If no ontology is available, the developer is in charge of providing a new ontology for describing the services;
3. Given the shared ontology, each service is described semantically, thus producing a set of description documents (e.g., in OWL-S [12]);
4. Agents read and reason on the shared descriptions to decide which service to use and how to use it.

The idea behind semantic Web services is about the use of formal descriptions of the characteristics of services to facilitate the reuse and to automate some of the most common processes, such as discovery, composition and invocation. In order to have such a scenario ready for adoption, there are nowadays two different approaches that are used to semantically describe Web Services:

1. Bottom-up: an incremental approach that adds semantics to existing Web services by linking semantic annotation to available WSDL annotations;
2. Top-down: an approach that makes extensive use of high-level ontology to semantically describe the characteristics of Web services.

The top-down approach uses the OWL-S, which derives from OWL ontologies and, in particular, it uses the ServiceProfile [13]. Briefly, the ServiceProfile describes three basic types of information: the organization providing the service, what the service provides, and other service relevant features. The ServiceProfile is mainly used for the discovery of a service; a service query is built from functional properties (i.e., Inputs, Outputs, Preconditions and Effects–IOPEs) and non-functional properties (that are to be interpreted by human users, e.g., service name and parameters that defines metadata about the service itself). The discovery phase is performed by an agent in the need to invoke a Web service. Given a service need, the agent prepares a service request form (via an OWL-S description) and a so called Matchmaker agent is in charge of discovering the best service on the basis of a semantic matching. To clarify how this complex task works, Figure 1 summarizes the steps involved in using a semantic Web service.

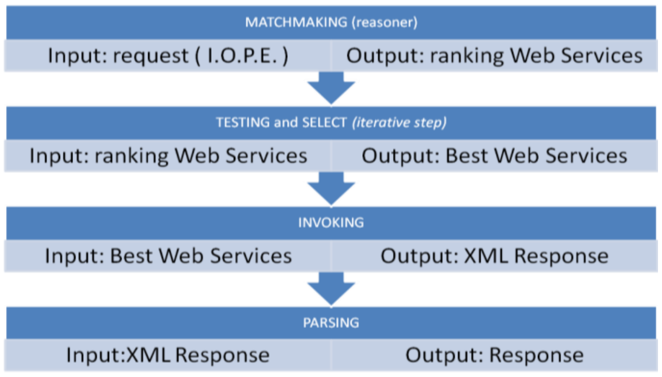

MATCHMAKING (reasoner)
Input: request (I.O.P.E.)
Output: ranking Web Services
TESTING and SELECT *(iterative step)*
Input: ranking Web Services
Output: Best Web Services
INVOKING
Input: Best Web Services
Output: XML Response
PARSING
Input:XML Response
Output: Response

Figure 1. Steps of the use of a semantic Web service.

Please note that the whole process is asynchronous and transparent to the user. In details:

1. MATCHMAKING. It is a way to identify the best match available to fulfill a request (even in part), and to provide a list of tenders ordered on their degree of match. In our case we are talking about the Semantic Matchmaking as requests and offers are expressed in an appropriately structured knowledge domain, i.e., a proper ontology. The input of this step is therefore a query (commonly expressed via OWL-S) while the output is a list of found Web services ordered from the most interesting to the least interesting.
2. TESTING and SELECT. Once we get the ranking of available Web services, it is deemed necessary to carry out a test phase before we hand it for invocation and, in the likely event of positive test results, we can actually move on to the next step.
3. INVOKING. Once the previous phase concretized the details of the service for its successful invocation(e.g., the physical address of the service and the names of operations), we pass to the invocation phase. This is not a simple static call; rather the adopted approach is that of a classic DII (Dynamic Invocation Interface).
4. PARSING. The last phase is meant to extrapolate from the (XML) response obtained the much anticipated output and to transform it into the desired shape for our purposes.

As anticipated earlier, the provision of semantics to RESTful Web Services is still undertaking slightly difficulties (see, e.g., [19, 7, 3, 11]). Many RESTful Web services are simply described in natural language, thus losing any ability to be machine-processable even if there are already standards both for syntactically describing services, e.g., the Web Application Description Language (WADL) [6], and for describing semantic annotations. Without going into too much in details, we can say that, given a good description of a syntactic RESTful Web service we can get to describing it semantically by means of everyday OWL-S. In fact, the OWL-S approach, created to semantically describe WSDL Web services, has recently been extended to accommodate this scenario: in particular, OWL-S provides an abstract layer that allows you to create multiple grounding strategies. In particular, a RESTfulGrounding is already available to serve as a link between OWL-S and RESTful Web Services. Obviously, in order to effectively exploit the RESTfulGrounding, the RESTfulGrounding ontology must be built in OWL format, or even better in OWL 2 (which is a modern reengineering of older OWL).

III. Web Services: WSDL/SOAP vs RESTful

It is nowadays clear that the technology of Web services ensures a uniform method for accessing software components located in different platforms and written in different programming languages. At the technological level, it is not instead clear what is the best protocol stack to use and, in fact, today there are still two main types of Web services: WSDL/SOAP Web services and Web APIs [15]. The first, i.e., the classic Web services, play an important role in the development of distributed applications among enterprises. Such an approach uses the standard Web Service Description Language (WSDL 2.0) to provide a machine-processable description of the structure of a service, its operations and input and output messages, and it uses the standard Simple Object Access Protocol (SOAP) for encoding the messages that the consumer and the provider exchange. On the contrary,Web services APIs, more commonly known as RESTful services, use resources as their key concept and such resources are accessible and editable through a set of well-defined operations, including GET (retrieve the current state of resource), POST (transfer the current state of resource), PUT (create new resource) and DELETE (delete a resource). As far as the Web interface description is concerned, RESTful services use the Web Application Description Language (WADL), which is suitable to describe Web-Based HTTP applications.

Glancing to the market trend until the end of 2010 and in particular considering the Web Services collected by *seekda.com* portal, we can say that, while classic Web services are still more numerous (around 28,000 [1]), the number of RESTful Services is growing quickly (around 1,900[2]) and above all they are quite clearly the choice of the software giants such as Google and Yahoo [17]. Obviously such figures are first approximations of the true (dynamic) situation, yet they are useful to show the current unbalanced situation about the use of SOAP against RESTful Web services.

Moreover, since the last two years, several important companies have been developing tools for mash-up creation that require no programming knowledge and therefore they are pushing the use of simpler and lighter Web APIs against more rigorous, yet complex, classic approach. Such tools, through a simple interface, allow selecting a number of RESTful Web services and chaining them together by piping one service's output into the next service's input while filtering content and making (slight) format changes.

A. WADL vs WSDL 2.0

The two most promising specifications that emerged in recent years have the main objective of providing machine-processable interfaces: Web Service Description Language (WSDL 2.0) for the classic services and Web Application Description Language (WADL) for Web APIs. While WSDL 2.0 (released by the W3C as a Recommendation on June 26th, 2007) is a formal standardization of WSDL 1.1, WADL (submitted to the W3C as a member submission two years ago) is designed to provide a machine processable protocol description format for use with HTTP-based Web applications, especially those using XML.

The two specifications may appear similar but, in reality, the differences that distinguish them are essential. In general WADL is simpler yet somewhat more limited, while WSDL 2.0 is more feature rich but complex. Coming into more technical detail, the following is a list of the most important differences:

- Resources vs interfaces: WADL is a resource-centric description language where documents are composed of a set of resource descriptions. On the other hand, WSDL is an interface-centric description language where documents are composed of a set of interface definitions.
- HTTP-only vs transport protocol independence: WADL is bound to HTTP transport protocol while WSDL is transport protocol independent.

B. WSDL/SOAP Services: Strengths and Weaknesses

As mentioned above, WSDL/SOAP Web services are mainly characterized by the complexity of their descriptions, yet they stand out for their wide dissemination because of their ability to provide a valuable tool for interoperability between heterogeneous systems. Another strong point of such services is the protocol transparency and independence, i.e., their ability to deliver the same message, in the same format, not only via HTTP but also via any other suitable transport protocol. Moreover WSDL description provides fine-grained, machine-processable details of request and response message syntax.

Looking for important weaknesses, SOAP is typically slower than other middleware technologies, e.g., CORBA, because it is based on XML format. Moreover, there is also a characteristic that at first sight might seem an advantage: SOAP was designed to slip through firewalls as HTTP using port 80 and people now sees that this might be a danger as *"SOAP goes through firewalls like a knife through butter"* [10].

TABLE I. CLASSIC WEB SERVICES STRENGTHS AND WEAKNESSES

SOAP/WSDL Web Services	
PROs	C**ONs**
• Protocol trasparence and independence; • Request and response message descriptions are machine processable; • Ideal for data centers and structured communication	• Slower than others; • Messages pass too easily through firewalls; • There is no possibility to use SOAP message easily from JavaScript

C. RESTful Services: Strengths and Weaknesses

Although RESTful is not appropriate for every scenario, it clearly provides interesting features for creating and interacting with Web services in a simple way and with a uniform interface that is highly stable (with no problems of compatibility or potential client break). Its simplicity is evident also because we just need a browser to start using Web services and there is no need of a Web service middleware. RESTful Web services leverage on existing well-known W3C standards, e.g., HTTP, XML, URI, and MIME, and the effort required to implement a client for a RESTful service is limited. Finally, since the possibility to choose among several lightweight message formats, e.g., the JavaScript Object Notation (JSON), RESTful services provide greater flexibility to optimize general performances.

Despite the clear advantages listed above, RESTful Web Services have several important weaknesses. First, we lack a unique method for building this type of Web services. In fact, we can choose between Hi-REST, using all of the 4 available verbs (GET, POST, PUT, and DELETE) and the

[1] End of 2010, http://webservices.seekda.com/

[2] Numbers from April 2010, http://www.programmableweb.com/

use of “nice URIs” and Lo-REST, in which only 2 of the 4 verbs are used (GET, POST). The latter type of RESTful Web services born to *(i)* cope with the fact that firewalls may not allow HTTP connections that use other verbs than GET and POST, and *(ii)* support the method attribute of an XHTML form. Such limitations have led to several workarounds but these may not be understood by all Web services, thus requiring additional development and testing efforts.

TABLE II. RESTFUL WEB SERVICES STRENGTHS AND WEAKNESSES

RESTful Web Services	
PROs	**CONs**
• Simpler than others; • Uniform, very stable interface; • Leverage on existing W3C Standars; • Testable by any browser; • Possibility to choose among several lightweight message formats; • Services can act like resources; • There is the possibility to easily invoke resources from client side code	• Can work only with HTTP; • Restricted set of allowed verbs; • Potential confusion between Lo-REST and Hi-REST services; • Difficult to build strongly typed objects to work within server side code;

D. Discussion

Comparative analysis revealed substantial differences between the two schools of thought. In general we can say that the most appropriate use heavily depends on the context of use. So, if the context of use is, e.g., a data center where you need interoperability between different servers and performance is of crucial importance, then the SOAP/WSDL approach is still the best choice. On the contrary, the use of the RESTful architectural style becomes an important choice if you need a simpler client side.

It is not accidental that the features that characterize the RESTful style services coincide with the first three principles of simplicity in software engineering: reduction, organization and time [10].

Finally, we must consider that such a simpler and high level approach is proving a huge success in the world so that most of new public services from large vendors (see, e.g., Google, Yahoo!, Microsoft, and Amazon) invest on RESTful Web services to share information.

IV. THE WEBSERVICES4AGENTS PORTAL

One of the first activities of our research was to look for a large group of Web services primarily by linking research institutions in the attempt to select a set of efficient and accessible services, especially in Europe and particular in Italy.Despite that the possibility to search for the right Web service is an essential prerequisite for their composition and reuse, search and discovery has become very problematic with the ever increasing number of services. Moreover, we experienced frequent discontinuity from the most important service providers: many repositories, working until a few years ago, are now off-line and those that are still running do not have efficient and effective search engines. In addition, today we have a very low availability of semantically described, working and accessible Web services. Given that such a premise is not encouraging, we decided to focus initially on the selection of purely syntactic Web services and then leave semantic annotations at a later stage.

In recent years the landscape of the Web Service repositories has changed substantially [17, 5]. Some repositories have ceased to provide their service, others continue to be available but they are not updated, while new repositories emerged in the International landscape.

The first result of our research is a prioritized list of service repositories that are now available and that offer good expectations for the near future. The following list discusses the repositories we found in our research, in relative order of importance:

1. IServe[3]: it can be considered as a hybrid provider that supports both service types . Although for what concerns the search it makes available a small number of service of type WSDL/SOAP, it was extremely interesting for finding many Web services described semantically. IService can be considered the first true global provider of Semantic Web Services.
2. Seekda! Services[4]: the largest, it has about 28,500 Web Services. It is active and very helpful.
3. Service-Finder[5]: the second largest and busiest, with about 20,000 Web Services assets. Also very active.
4. WebserviceX.NET[6]: it provides about 70 Web services grouped into seven categories. This provider was established about six years ago but since then it has never been updated. Many Web services tested were faulty and inefficient.
5. Xmethods[7]: it provides hundreds of services but it also stalled frequently and it has no search engine.

On the one hand the large and almost vain effort made to catalog a series of useful Web services, and on the other the need of a folder of active Web services, pushed for an independent solution: to create a portal to host fully usable Italian (syntactic) Web services and hopefully, in the near future, an Italian community centered around the exchange of useful semantic Web services.

Moreover, in addition to providing a collection point for Web services in Italy, offering a Web interface for their search, our goal is primarily to provide an interface for

[3] http://iserve.kmi.open.ac.uk
[4] http://webservices.seekda.com/
[5] http://demo.service-finder.eu/index
[6] http://www.webservicex.net
[7] http://www.xmethods.com

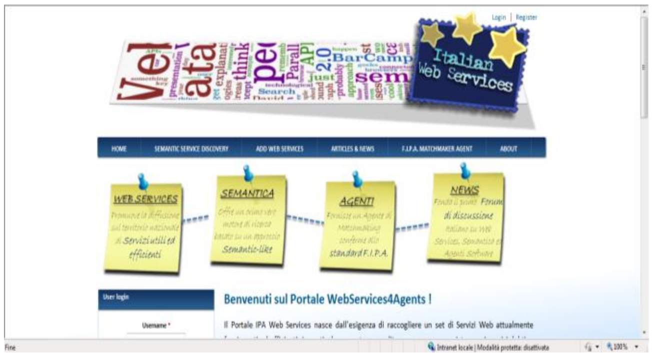
FIPA agents. In particular we would like to offer clients the opportunity to communicate with a matchmaker agent that, by using a shared ontology and the FIPA standard, returns the most suitable Web Services after the appropriate matchmaking task. We have short term and long term goals. The long term goal is to create a collection of thousands of Web services described by semantic standards (e.g., OWL-S) and to make them searchable by full semantics-aware techniques through any agent capable to formalizing a request in the form of IOPEs. In particular, we will make available to any agent, a Semantic Web Services Register (SWSR) containing all the semantically described Web Services. Our idea is to use the well-know OWLS-MX Semantic Matchmaker [9] because it provides a hybrid semantic Web service matching facility and it utilizes both logic-based reasoning and content-based information retrieval techniques for services specified in OWL-S. In addition, for an adequate control of the messages used to communicate with the agent, we will provide shared vocabulary (i.e., an ontology for communication) and interaction protocols. We decided to use JADE for implementing the matchmaker agent and to exploit its functionality to support a standard-based, reliable communication.

However, the successful provision of semantics to Web services has already been a difficult undertaking: all the attempts carried out so far, despite some relevant positive results, have led to final negative results. The difficulty lies trivially in still not having a large group of semantically described services. For this reason, our first short term goal is to create a portal for bringing together a set of syntactically described Web services, that are meant to be useful and immediately effective for the Italian market. For example, the portal is expected to publish interesting Web services targeting concrete problems like the discovery of opened pharmacies of a certain Italian city in a certain day, or secretarial service for a student of an Italian University. Thanks to our novel Web services folder it will be possible to create an interface through which FIPA agents could seek and discover Web services. Up to this point, the beneficial results of the proposed portal would be essentially the availability of services for the Italian market and their ease of access.

Before entering the big issue of semantics, we will then consider hybrid approaches intended to empower the pure syntactic search, as follows:

1. To give the possibility of using so-called social tagging of Web Services, to make them searchable via different, yet related, keywords than those used in descriptions;
2. To use the Italian WordNet Ontology API [16] to exploit the synonyms of keywords.

Starting by these two intermediate steps, toward the heavier and true semantic approach, we can have both a folder Web services ready to use, and the opportunity to carry out interesting research on hybrid syntactic/semantic search approaches.



Benvenuti sul Portale WebServices4Agents !

Il Portale IPA Web Services nasce dall'esigenza di raccogliere un set di Servizi Web attualmente

Figure 2. *www.webservices4agents.com* portal for Web services

V. DISCUSSION

In this paper we intended to make a reality check on the overall status of the Web service technology and we also aimed at better understanding the relationship between agents and Web services today. Thus, the research presented here has led to ask ourselves a number of interesting questions and to observe that the Web service technology is still far from its full maturity.

Glancing at the current trends, we can certainly say that the RESTful Web services standard is quickly taking place over other approaches mainly because of its lightweight nature. Furthermore, it is equally clear that the advent of the Semantic Web would accelerate the rise of semantic Web services and, in particular it would boost the need of agents to effectively use semantic Web services.

For all these reasons, we have recently created the WebServices4Agents Portal (*www.webservices4agents.com*) in the intent to provide both an up-to-date repository for (even RESTful) Web services and to open an access to such services to FIPA agents. We hope that the portal will be an important and useful meeting point for all agentswho want to discover and invoke useful Web services.

Our first goal is to set up a large repository of Web Services and to have an agent capable of providing a matchmaking service on such a repository.After that we immediately want to match with the reality of semantic Web services landscape: first by trying with semi-semantic approaches, e.g., using social tagging, and finally going to the full power of the Semantic Web.

We conclude this paper with a list of questions that arose during our work and that are still open and demanding for discussions.

What are the results of efforts made so far to make semantics to Web services available? Are there "less invasive" ways capable to bring (softer) semantic to Web services?

The issues about the Semantic Web have been subjects of extensive discussions since about ten years. However, even today, especially from the point of view of Web services, there is not a great use of semantics. The lowering of performances and the difficulties for service providers to offer semantically annotated Web services are just two of the problems hindering the rise of semantic Web services. In order to avoid this bottleneck, our idea is to propose softer approaches: *(i)* the use of keywords by exploiting the power of WordNet, and *(ii)* the adoption of ordinary social tagging to make richer and more effective searches.

Taking now formal semantics into account and noting the sudden growth of RESTful Web Services, can we use OWL-S descriptions for this kind of Web services?

As briefly discussed in earlier, we can say that, starting from a good description of a syntactic RESTful Web service, we can get a semantic description by means of OWL-S because it has been recently extended to exploit the semantics of RESTful Web services. In particular, the recent introduction of the RESTfulGrounding should provide the possibility of using OWL 2 for semantically annotating RESTful Web services.

Will service developers be encouraged to advertise their services syntactically and also semantically?

First, please remember that in Italy the development of truly efficient and free Web services is still far from reality and we do not have yet a means to expose them and to make the effectively available. Through the WebServices4Agents portal producers of Web services and their respective consumers would finally meet through an interface capable of selecting the best service for a consumer's needs. The portal would give the possibility to query a JADE matchmaker agent which would then return the best service.

Under the assumption that we have a folder of semantically described services in the portal and a matchmaker agent for searching, what would the implications be from the point of view of agent systems?

This is definitely one of the most interesting points emerged during our research. Our vision is to have a large number of accessible Web services and to have the possibility of composing such services with the help of a semantic infrastructure. In recent years, Telecom Italia S.p.A. and the University of Parma have already implemented WADE (Workflow Agent Development Environment) [2], a software platform based on JADE capable of bringing to agents the possibility of structuring their work by means of workflows. Therefore the idea that we propose in the long term is to harness the power of WADE to bring to the consumer the best Web services at the right time and in the right place in a fully personalized way.

ACKNOWLEDGMENT

This research project has been supported by Working Capital Initiative of Telecom Italia S.p.A.

REFERENCES

[1] Bellifemine, F., Caire, G., and Greenwood, D. (2007) *Developing multi-agent systems with JADE*. Wiley Series in Agent Technology.

[2] Caire, G., Gotta, D., and Banzi, M. (2008) WADE: A software platform to develop mission critical applications exploiting agents and workflows, *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*.

[3] Gomadam, K., Kopecký, J., and Sheth, A. P. HTML Microformat for Describing RESTful Web Services and APIs, In *IEEEWICACM International Conference on Web Intelligence and Intelligent Agent Technology (2008) Volume: 1, Publisher: Ieee, Pages: 619-625.*

[4] Greenwood, D., and Callisti, M. (2004) Engineering Web Service-Agent Integration. In *Proceedings of the IEEE Conference of Systems, Man and Cybernetics*.

[5] Hagemann, S., Letz, C., and Vossen, G. (2007) Web Service Discovery - Reality Check 2.0, in *Proceedings of the Third International Conference on Next Generation Web Services Practices* (NWeSP'07).

[6] Hardley, M. (2006) *Web Application Description Language (WADL)*, Technical report, Sun Microsystems.

[7] Hernandez, A. G., and Garcia, M. N. (2010) A Formal Definition of RESTful Semantic Web Services, *In 1st International Workshop of RESTful Design, Carolina pp. 39-45.*

[8] JADE - Java Agent Development framework, *http://jade.tilab.com.*

[9] Klusch, M., Fries, B., Khalid, M., and Sycara, K. (2005) OWLS-MX: Hybrid OWL-S Service Matchmaking, Agents and the Semantic Web Papers in *Proceedings of the AAAI Fall Symposium*, AAAI Press.

[10] J.(2006) Cambridge (Ma), *MIT Press.*

[11] Maleshkova, M., Pedrinaci, C., and Domingue, J. (2008) Supporting the Creation of Semantic RESTful Service Descriptions, *In Proceedings of ISWC09, Washington D.C., USA, 2009*

[12] Martin Davide, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci (2004),OWL-S, Semantic markupfor web services; *W3C member submission*.

[13] Martin, D., Paolucci, M., and Wagner, M. (2007) Bringing Semantic Annotations to Web Services: OWL-S from the SAWSDL Perspective. In Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) *Proceedings of ISWC 2007.* LNCS, 4825:340-352. Springer, Heidelberg.

[14] McGuinness, D. L., and van Harmelen, F. (2004) OWL Web Ontology Language Overview, *W3C Recommendation*.

[15] Pautasso, C. (2008) RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision, In *Proceedings of the 17th International World Wide Web Conference (WWW2008), Bejing, China, April 2008.*

[16] Pianta, E., Bentivogli, L., and Girardi, C. (2002) MultiWordNet, Developing an aligned multilingual database, In *Proceedings of the First International WordNet Conference*, 293-302.

[17] Sabou, M., Maleshkova, M. and Pan, J. (2010) Semantically Enabling Web Service Repositories, in Pan, J. Z., and Zhao, Y. (eds.) *Semantic Web Enabled Software Engineering.*

[18] Savarimuthu, B. T. R., Purvis, M., Purvis, M., and Cranefield, S. (2005) Integrating Web services with agent based workflow management system (WfMS), In *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI 2005).*

[19] Sheth, A. P., and Lathem, J. (2007) SA-REST *Semantically Interoperable and Easier-to-Use Services and Mashups*, Published in *Journal IEEE Internet Computing, Volume 11 Issue 6.*

[20] Takase, T., Makino, S., Kawanaka, S., Ueno, K., Ferris, C., and Ryman, A. (2008) Definition languages for RESTful Web services: WADL vs. WSDL 2.0, *IBM Reasearch.*