Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2010 Proceedings

Americas Conference on Information Systems (AMCIS)

8-2010

Semantics take the SOA registry to the next level: an empirical study in a telecom company

Catarina Ferreira da Silva
University of Coimbra, cferr@dei.uc.pt

Paulo Rupino da Cunha
University of Coimbra, rupino@dei.uc.pt

Paulo Melo
University of Coimbra, pmelo@fe.uc.pt

Marinos Themistocleos
University of Coimbra, marinos@dei.uc.pt

Semantics take the SOA registry to the next level: an empirical study in a telecom company

Catarina Ferreira da Silva
Centre for Informatics and Systems of the University of Coimbra (CISUC)
cferr@dei.uc.pt

Paulo Rupino da Cunha
CISUC and Department of Informatics Engineering of the University of Coimbra
rupino@dei.uc.pt

Paulo Melo
Instituto de Engenharia de Sistemas e Computadores de Coimbra and University of Coimbra
pmelo@fe.uc.pt

Marinos Themistocleos
Centre for Informatics and Systems of the University of Coimbra
marinos@dei.uc.pt

ABSTRACT

We describe an empirical study of the creation of a Semantic Service Registry in the context of the Operations Support Systems (OSS) department of a telecom company, to address an emerging problem of finding the right services to build new business processes in a pool that steadily increases. We show how to obtain an ontology for the telecom domain to annotate services and thus benefit from semantic technologies to effectively find them based on description logics inference mapping. We designed and implemented a proof of concept for providing a matching degree even when the cardinality of the service elements of the query and the cardinality of the service elements being sought differ. This is relevant for web services reusability and flexibility. Our solutions are overviewed and a set of lessons learned are discussed.

Keywords

Semantic Service Registry, semantic matching of web services, services semantic annotation, ontology, SAWSDL.

INTRODUCTION

Competitive markets constantly pressure companies to become more agile, forcing them to leverage their legacy systems in novel ways: new business processes should be designed and quickly deployed to support new products or services; existing ones need to be reengineered or frequently tuned to keep up with best practices in the industry. The need to cope with these constant reconfigurations of existing and new functionality is one of the main drivers for the evolution towards Service-Oriented Architecture (SOA). A key component in an SOA is the registry or repository of services that are available to be composed into different orchestrations that make up the higher-level business processes. However, as the number of available services rises, the ability of the analyst to find the ones s/he needs decreases. It's not feasible to manually sift through hundreds of services, reading each description every time a new business process needs to be created. A traditional syntactic search is not reliable either, since for any given service name in the registry, there are numerous other semantic equivalents by which the user can query. For instance, a syntactic search for *GetCustomerData* service name would not find the semantic similar service named *ObtainClientRecord*.

Semantic technologies, thus, become increasingly relevant in this search context, enabling the analyst to query the existing pool of services in terms of their business meaning instead of some arbitrary name. In an attempt to study this area and further explore relevant issues and concepts, we have designed and built such a Semantic Service Registry as a proof of concept for a major telecom company. In this paper we describe how we did it and what were the advantages, problems, and general lessons learned from this process.

The remainder of this paper is organized as follows: we start by reviewing semantic standards for Web services descriptions and related semantic matchmaking techniques. We present the methodology we used in this research and the related empirical data about a Semantic Service Registry implemented in the context of the Operations Support Systems (OSS)

department of a telecom company which is gradually migrating to an SOA. Then we discuss some outcomes, problems we found, and lessons that we learned so far, just before concluding with some remarks about future work.

ON THE INTERSECTION OF SOA AND SEMANTICS

Usually, services are described using Web Services Description Language, WSDL 2.0 (Chinnici, Moreau and Ryman, 2007), that specifies a format to define interfaces, *i.e.* the technical aspects of calling web services. It can describe two different aspects of a service that are its signature, particularly service name and service elements, and its binding and deployments details, such as protocol and location. Although WSDL 2.0 provides the ability to extend WSDL files, the underlying XML language (Bray, Paoli and Sperberg-McQueen, 2006) is not capable of conveying precise and unambiguous semantics. Indeed, WSDL does not support semantic description of services. It focuses on the grounding of services and it does not support the definition of logical constraints between their input and output elements, even though it has a concept of input and output types as defined by XML Schema Definitions, XSD (Van der Vlist, 2002).

Representational techniques being developed for the Semantic Web can be used to capture and process semantics. Some of these techniques are grounded on XML language, bringing other complementary language constructors. The Semantic Web Activity group ("W3C Semantics," 2004), from the W3C, recommends specific languages such as:

- Resource Description Framework, RDF (Beckett, 2004),
- Resource Description Framework Schema, RDF(S) (Brickley and Guha, 2004),
- Web Ontology Language, OWL (McGuinness and Van Harmelen, 2004).

Particularly, OWL includes three sublanguages: OWL-lite, OWL-DL, and OWL full. The first two correspond to decidable description logics (Baader, Calvanese and McGuinness, 2003). Decidability implies that fundamental questions about an ontology are guaranteed to be answerable, such as the question of subsumption. OWL 2 (OWL2-Overview 2009), adds new functionality with respect to OWL 1. Some of the new features are simple syntactic improvement (e.g., disjoint union of classes) while others offer new expressivity, such as enhanced annotation capabilities and property chains.

In the domain of Semantic Web Services, the research community proposed several structured service description languages. Examples of these languages are Semantic Markup for Web Services, OWL-S (Martin, Burstein and Hobbs, 2004) and Web Service Modelling Language, WSML (Bruijn, Lausen and Polleres, 2005) which have formal logic semantics groundings. Another outcome in this domain is the Semantic Annotations for WSDL and XML Schema, SAWSDL (Farrell and Lausen, 2007), a W3C recommendation from 2007. We briefly describe these three Semantic Web Services description languages hereafter and compare their major similarities and differences in Table 1 and Table 2, and advantages and limitations in Table 3.

	WSML/WSMO	
OWL-S	**Major similarities**	**Major differences**
	Use of logic languages for formal representation	WSML attributes are defined locally to a class and should in principle not be used outside of the context of that class and its subclasses
	Use of XML Schema Datatypes	WSML provides several sublanguages with different expressive power whereas OWL-S preconditions and effects are represented as logical formulas that can be expressed in any appropriate logic (rule) language such as KIF, PDDL, and SWRL
	Use of URI for object identification	WSMO explicitly decouples the requester and provider viewpoints: users goals are defined independently from Web Services capability and they can be linked through wgMediators
		In OWL-S, non-functional properties can only be associated with the service profile while they can be defined for any of the core WSMO elements

Table 1. Major similarities and differences between OWL-S and WSML/WSMO.

	OWL-S		WSML/WSMO	
	Major similarities	**Major differences**	**Major similarities**	**Major differences**
SAWSDL	Both languages provide attributes to extend WSDL and thus to attach semantics to web service descriptions	OWL-S sets up an ontology of web service and process model while SAWSDL only provides a WSDL semantic annotation mechanism	Both languages intend to provide mechanisms to semantically describe web services	WSML/WSMO provide a vast framework to model semantic web services whereas SAWSDL is easier to learn and to implement

Table 2. Major similarities and differences between SAWSDL and OWL-S and WSML/WSMO.

	Advantages	**Limitations**	**References**
OWL-S	OWL-S offers the choice between different languages for the specification of preconditions and effects	It is unclear how preconditions and effects specification language interact with OWL	Lara, Polleres, Lausen, Roman, Bruijn and Fensel, 2005
	Provides mappings between OWL-S parameters and WSDL input/output message parts	In contrast to WSDL, an OWL-S process model cannot contain any number of completely unrelated operations	Klusch, 2008a
		Has limited expressiveness of service descriptions, which corresponds to that of its underlying OWL-DL	
		The semantics of the OWL-S process model is missing	
WSML	Provides mapping to OWL ontologies	Lacks of formal semantics of WSML service interface	(Bruijn, Fensel, Keller, Kifer, Lausen, Krummenacher, Polleres and Predoiu, 2005); (Klusch, 2008a) and http://www.w3.org/Submission/WSML/
	Surrounded by a complementary framework and tools, such as WSMX, several reasoners, WSMO studio and API	Lacks of of principled guidelines for developing the proposed types of mediators for services and goals in concrete terms	
SAWSDL	Is independent of semantic representation languages	Has no formal semantics	(Klusch, 2008a) and http://www.w3.org/2002/ws/sawsdl/
	Less complex than OWL-S or WSML as it only adds three basic constructs to extend WSDL	Is a mere syntactic extension of WSDL, as semantics in the scope of this specification refers to sets of concepts identified by annotations	

Table 3. Advantages and limitations of OWL-S, WSML and SAWSDL.

Semantic Markup for Web Services (OWL-S)

Based on OWL (Martin et al. 2004) propose OWL for Services (OWL-S) that currently supersedes DAML-S (Burstein, Ankolenkar and Paolucci, 2003) and intends to add precise semantics to service description. Some attributes are added to WSDL extensions so that to connect this language with OWL-S and the generated files. OWL-S defines a top abstract ontology for services with four major elements, namely: *service*, *service profile*, *service model* and *service grounding*. The *service* concept is the root reference point for declaring a web service; the *service profile* sets out what a service does at a high level, describing its functionality (by using functional parameters that are hasInput, hasOutput, Precondition and Effect, known as IOPE) and non-functional properties (such as serviceName and serviceCategory) that are used to advertise and then to locate services based on their semantic description; the *service model* describes how the service achieves its functionality, including the detailed description of its constituent processes; the *service grounding* specifies how to use the service i.e. how a client can invoke the service. In OWL-S 1.1 (Martin, Hodgson, Horrocks and Yendluri, 2006), the IOPE parameters are specified in the process model, captured inside the service model, with unique references to these definitions from the service profile, where Inputs and Outputs service parameters specify the data transformation produced by processes. Here a process means a specification of the ways a client may interact with a service.

OWL-S benefits from a large support from the community, as several software and applications are being developed for this language and ontology of semantic service descriptions, such as the OWL-S editor (Scicluna, Abela and Montebello, 2004), the OWL-S Application Programming Interface (API) and OWL-S service matchmakers, like OWLS-UDDI (Paolucci, Kawamura, Payne and Sycara, 2002), OWLSM (Jaeger, Rojec-Goldmann, Liebetruth, Mühl and Geihs, 2005) and OWLS-MX (Klusch, Fries and Sycara, 2009), to name a few.

Web Service Modeling Language (WSML) and Web Service Modeling Ontology (WSMO)

The Web Service Modeling Language (WSML) (Bruijn, Fensel, Keller, Kifer, Lausen, Krummenacher, Polleres and Predoiu, 2005), is a formal language for the semantic markup of web services and aims at providing means to formally describe all the elements defined in the Web Service Modeling Ontology (WSMO); this has four main elements (Roman, Keller, Lausen, Bruijn, Lara, Stollberg, Polleres, Feier, Bussler and Fensel, 2005): ontologies, which provide the terminology used by other WSMO elements, Web services, which provide access to services that, in turn, provide some value in some domain, goals that represent user desires, and mediators, which deal with interoperability problems between different WSMO elements. WSML provides means to describe those four main aspects and it is used to describe a semantic web service - based on

different logical formalisms, namely, Description Logics, Frame Logic and Logic Programming - in terms of its functionality (service capability), imported ontologies and interface to enable access. Service capability describes desired and provided state-based functionalities in terms of its precondition (conditions over the information space), postcondition (result of service execution delivered to the user), assumption (conditions over the world state to met before service execution), and effect (how does the execution change the world state).

To support WSML some software is being developed, such as WSML-DL and WSML-Rule reasoners, WSML variant validator and WSML service editor associated with the WSMO studio; the SUPER project uses WSMO as the underlying ontology (Dimitrov, Simov, Momtchev and Konstantinov, 2007) and the Internet Reasoning Service (IRS-III) framework allows applications to semantically describe and execute WSMO-based Web services (Domingue, Cabral, Hakimpour, Sell and Motta, 2004). Haller, Gomez and Bussler (2005) propose a specific SOA architecture that applies WSMO framework and uses a specific execution environment, Web Service Execution Environment, WSMX (Zaremba and Oren, 2005); in this environment, there is a need for specific adapters to transform external messages into the WSML compliant format understood by WSMX, and mediators that perform tasks such as translation between ontologies.

Semantic Annotations for WSDL and XML Schema (SAWSDL)

SAWSDL approach (Farrell and Lausen, 2007), grounded on previous WSDL-S (Akkiraju, Farrell, Miller, Nagarajan, Schmidt, Sheth and Verma, 2005), proposes three extension attributes for WSDL and XML Schema definition languages that allow description of additional semantics of WSDL components. The SAWSDL specification defines how semantic annotation is accomplished using references to semantic models, such as ontologies. It provides mechanisms by which concepts from these semantic models, typically defined outside the WSDL document, can be referenced from within it and XML Schema components using annotations. SAWSDL defines the following extensibility attributes to WSDL 2.0 elements for their semantic annotation:

- A `modelReference` extension attribute that is used to specify the association between a WSDL or XML Schema component and a concept in some semantic model. It is useful to annotate XML Schema type definitions, element declarations, and attribute declarations as well as WSDL interfaces, operations, and faults;
- `liftingSchemaMapping` and `loweringSchemaMapping` extension attributes. The former transforms XML data into instances of a semantic model and the latter does the opposite. It transforms semantic model instances into XML data. This SAWSDL schema mapping intends to address post-discovery issues when using Web services, such as how to overcome structural mismatches between the semantic model and the service inputs and outputs.

To support SAWSDL some software is being developed, such as Lumina and Radiant, both part of the METEOR-S project (Patil, Oundhakar, Sheth and Verma, 2004).

Semantic matching of web services

Some approaches try to bring about automation in order to help the complex and tedious mapping task of finding a specific service by comparing the request with the available services. (Klusch 2008) classifies semantic matchmaking techniques, and their associated tools, as logic-based, non-logic-based and hybrid:

- Non-logic-based matching applies techniques such as graph matching, data mining, linguistics, or content-based information retrieval to exploit semantics that are either commonly shared (in XML namespaces) or implicit in patterns or relative frequencies of terms in service descriptions;
- Logic-based semantic matching of services like those written in the service description languages OWL-S and WSML exploit standard logic inferences;
- Hybrid matching refers to the combined use of both types of matching.

We reviewed and tested a set of Semantic Web service matchmakers, such as:

- OWL-S UDDI matchmaker (Srinivasan, Paolucci and Sycara, 2004),
- OWLS-MX (Klusch, Fries and Sycara, 2006),
- SAWSDL-MX (Klusch and Kapahnke, 2008),
- WSMO-MX (Klusch and Kaufer, 2007),

- OWLSM (Jaeger, Rojec-Goldmann et al., 2005) and
- FUSION (Kourtesis and Paraskakis, 2008).

Although the in-depth description of this analysis is beyond the scope of this paper, the main outcome was the selection of the SAWSDL-MX tool for our experiments in the telecom company, that are described in section 5. Generically, the choice of the matchmaker depends on the context, particularly on the ontologies and service descriptions at hand.

TAKING THE SOA REGISTRY TO THE NEXT LEVEL

Independently of specific SOA infrastructure or addressable registries of services, at some moment in SOA lifecycle it is necessary to match service request descriptions with available service descriptions, in order to verify if the latter corresponds to service consumer needs. To automate this task as much as possible, the semantics of service descriptions have to be precisely described, e.g. in ontologies, as these semantics when expressed in formal languages can help disambiguate the description of Web services during their automatic discovery and composition.

We propose a novel approach on semantic matching of web services where the main innovations are new partial mapping types between service elements. These ideas were implemented and tested in the context of a proof of concept on Semantic Service Registry where semantic web services are published and available for further search.

As a consequence of the standards review and the analysis of the matchmaker tools we did in preview steps, we selected the SAWSDL W3C recommendation since it is less complex than OWL-S or WSML in the sense that it only adds three basic constructs to extend XML WSDL representations and thus connect these to external metadata information. So, SAWSDL enables easily attachment of semantics to WSDL descriptions and is convenient for applications and domain reference models that do not need the complexity or expressivity of OWL-S or WSML languages. We decided to incorporate and adapt the SAWSDL-MX tool in our proof of concept, as it has a high maturity level according to the evaluation we conducted.

RESEARCH APPROACH

The IS research is characterized by methodological pluralism and the selection of an appropriate research methodology is a topic that attracts researchers' attention (Galliers, 1994; Miles and Huberman, 1994). Galliers (1992) stated that the study of information systems is a complex topic, and very much a social, rather than a wholly technical subject. Therefore, a researcher in the IS field has to choose among a variety of research methods, approaches and techniques to develop an appropriate research framework. Galliers (1992) reported that it is unlikely that there is a universal IS research approach, which can include all the domains of knowledge needed for the study of Information Systems. Thus, the selection of an appropriate research approach that can support the study of Information Systems is one of the most difficult and critical decisions for a researcher.

The aim of this paper is to investigate a proof of concept on Semantic Service Registry. In order to test our proof of concept we have selected an interpretive, qualitative approach that employs action research strategy. Interpretitivism stance was selected as it allows us to navigate and explain better this phenomenon. Also, we suggest that in the context of this research a qualitative approach is more appropriate as such approach can be used to: (a) investigate little-known phenomena such as Semantic Service Registry; (b) examine in depth complex processes; (c) examine the phenomenon in its natural setting and, (d) learn from practice.

Moreover, we employed an Action Research strategy as it a "systemic inquiry that is collective, collaborative, self-reflective, critical and undertaken by participants in the inquiry" (McCutcheon and Jung 1990:148). Action research allows the researchers to work and collaborate with an organization and it supports the use of spiral cycles consisting of four main phases: (a) Planning, (b) Acting, (c) Observing and (d) Reflecting. This is illustrated in Figure 1.

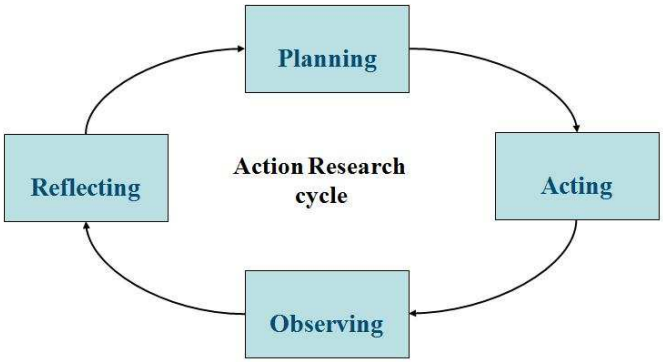

Figure 1. The four main phases of the Action Research cycle.

There are three types of action research: (a) the scientific-technical view of problem solving, (b) practical-deliberative action research and (c) critical-emancipatory action research. In this paper, we used the first type as it allows us to test a particular intervention based on a pre-specified theoretical framework-proof of concepts. In this type the nature of cooperation among the organization and the researcher is technical (like our case). In addition to this, the first type of action research allows us to identify a problem (Service Registry) and a specific intervention (our proof of concept on Semantic Service Registry). According to this type practitioners from the telecom company are involved and they agree to facilitate with the implementation of our intervention.

EMPIRICAL STUDY IN A TELECOM COMPANY

We have been working with the OSS department of a telecom company which is gradually migrating to an SOA (Cunha, Melo and Ferreira da Silva, 2009). This group handles the systems used for service provisioning, including maintaining the network inventory, configuring resources, and monitoring operation. As required pieces of business logic become exposed as web services, on an as-needed basis, the number of accessible services increases. When business analysts need to find specific ones, it is useful to provide them with a tool that enables to match that service request descriptions with service descriptions available in the registry as close to business meaning as possible. For instance, when a user looks for a service that calculates client invoice, the matching system should be able to provide information about an available service named *customer_bill*. Although registries or repositories are frequently pointed out as the solution to list which services are available and how to invoke them, a syntax-oriented mechanism is of decreasing usefulness when composing new processes. In this situation, the analyst needs to be able to query the existing pool of services in terms of semantics, to find perfect or close matches to desired business logic regardless of the actual service names. So, more than just list the services and how to invoke them, we need information about their meaning in the business context. It is not feasible to sift through hundreds of services, reading each description every time a new business process needs to be created.

The solution we developed explores the use of semantic technologies to address this problem, by introducing a semantic registry of services with appropriate annotations that link concepts in service descriptions to ontological concepts (Guarino, 1998). WSDL service descriptions are annotated with SAWSDL annotations before publication in the registry of services (Figure 2); the SAWSDL `modelReference` attribute associates a WSDL input or output to an explicit concept defined in a suitable ontology. Since none exists at the moment for the telecom company, we built a draft from other accepted standards in this domain, such as the OSS through Java (OSS/J) Initiative (Buschmann, Ebbert, Raymer, Dillon, Gauthier, Milham, Pedneault, Perrot, Plutino, Reilly and Wilmes, 2006) from the TeleManagement Forum (TMF, 1988).

The method we used for the creation of a (simplified) ontology is one contribution of this work. The usage of converters to bootstrap the creation of ontologies is somewhat standard, however we describe a process to create such ontologies even when standard converters are hard to use. We propose adapting industry API to extract common data, adapting those API to a format that can be handled by the converters and cleaning up the resulting ontology using domain data obtained from documentation.

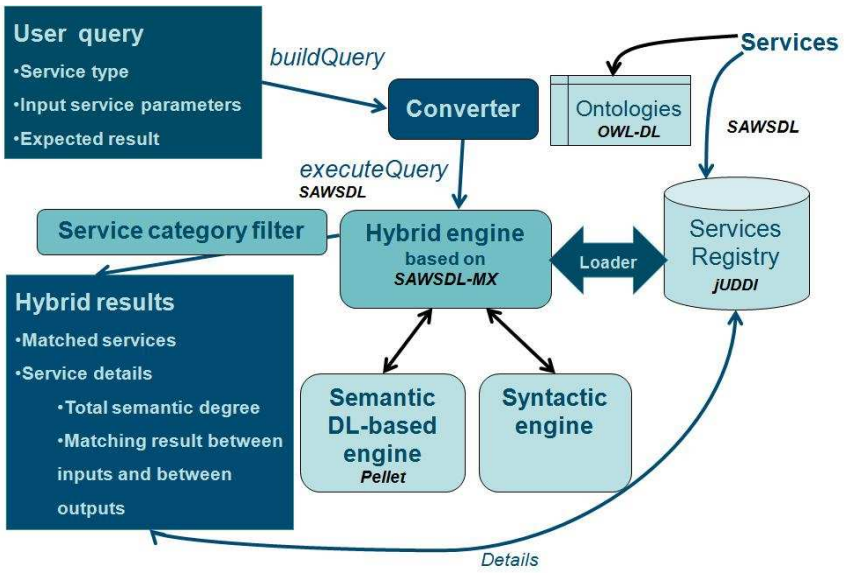

Figure 2. Main components of the proof of concept on Semantic Service Registry.

Obtaining suitable ontologies

To create our ontology, we selected the Order Management SOA Enablement OSS/J API (Dillon, Uzieblo, Vermaas, Millatiner and Wilmes, 2007) as original data, because it represents core OSS tasks. We then converted it from XSD format into OWL using an XSD2OWL converter (García, 2005). This task generated an OWL-Full version of the Order Management OSS/J API (OM API shortly) that was then corrected in order to obtain a consistent OWL-DL version that can be processed by a description logics inference engine. Nevertheless, the structure of this OM API OWL-DL ontology version is not very deep as the original XSD version already has a similar narrow structure, so we decided to semantically enrich the OWL-DL version to further benefit from these semantic technologies. Therefore, we obtained two OM API OWL-DL meaningful improved versions: we decided to organize one of the ontologies by types of functional OM service categories, such as *create*, *start* and *suspend*. The other ontology was also improved with new concepts that deepen the hierarchical structure with information from the OM API Java documentation. Although supported by ontology editors, such as Protégé (Figure 3), these tasks are arduous as they were done manually and therefore prone to errors.

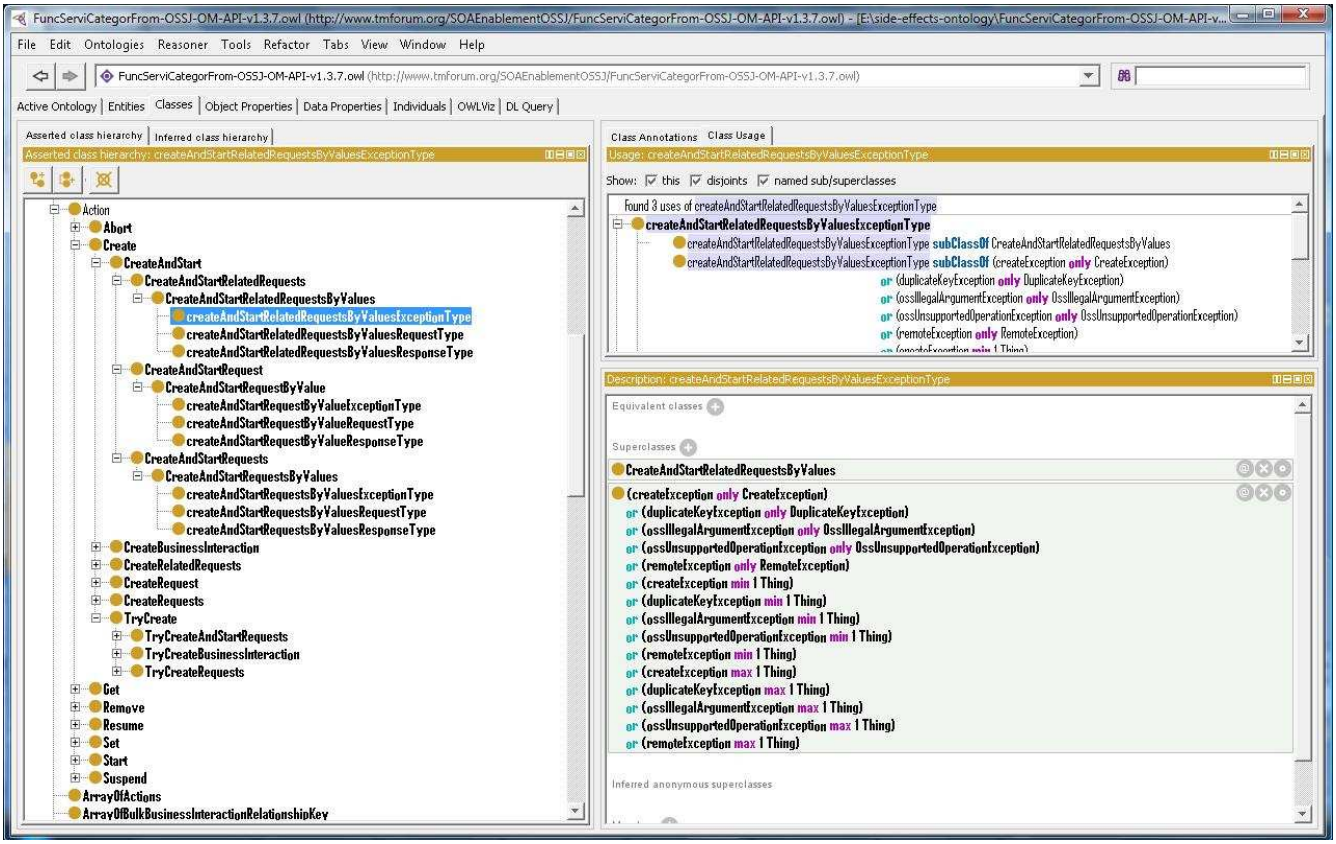

Figure 3. Ontology of Order Management API service categories, using Protégé ontology editor.

Semantic annotation of WSDL OM API operations

We analyzed each WSDL OM API operation to annotate them with adequate concepts of the ontologies achieved in the previous step. Each WSDL operation input and output was annotated with one or more concepts of the ontology enriched with elements of the OM API Java documentation whereas each WSDL operation `portype` was annotated with a corresponding concept of the service categories ontology. For instance, the input of the OM API *createRequestByValue* operation (Figure 4) was annotated with the following concepts: ClientId, Bulk, ExpectedCompletionDate, Priority, RequestedCompletionDate, ValidFor, RequestKey, ValueType; while its output was annotated with RequestKey concept and `<wsdl:portType>` element was annotated with CreateRequestByValue service category. After the semantic annotation of thirty OM API operations, these where published in the OSS services registry. The application combines a UDDI semantic registry server of OSS services and an adapted version of the SAWSDL-MX (Klusch and Kapahnke, 2008) semantic and syntactic matching engine (Figure 2).

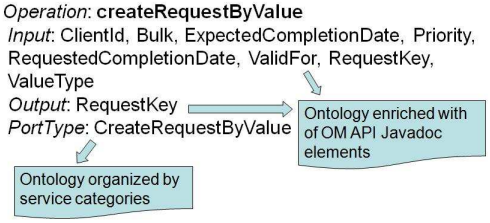

Figure 4. Representation of the annotation elements of the OM API *createRequestByValue* operation.

Semantic mapping of OM API operations

The user query interface of the application (Figure 5) enables the search for services by its category (service type), input parameters and expected results. Thus, a user query is composed of words that are OWL-DL concepts or attributes, defining

the service category and service input/output (I/O) elements s/he seeks for. A service can have zero or more I/O elements (input parameters and expected results). The query is converted to a SAWSDL document in such a way that it can be further processed by the SAWSDL-MX engine; the document represents an operation with as much I/O elements as the query and a service category. Then the semantic search process combines a hybrid (i.e. semantic and syntactic) matching of I/O elements with a semantic matching of service category. Semantic mapping results returned by SAWSDL-MX are *Exact*, *Subsumes*, *Plug-In*, *Subsumed By* or *Fail* and compare the total semantic degree between a query and a published service, using an injective bipartite graph matching (Klusch and Kapahnke, 2008). Notice that the support for service category had to be added to the original tool, which only compared inputs and outputs.

The original process mentioned above works correctly when the cardinality of the services is known, but is not adequate when the number of elements required are not known in advance, or when the person doing the search has no knowledge of the kind of elements required for a particular service (which may happen in the process design stage since the developers may not know enough about the semantics of particular services to understand which elements are indeed required). To support this usage, and as an additional contribution of this work, we designed a process for providing a matching degree even when the cardinality of the service for the query and the cardinality for the service being tested differ. We provide a mechanism to supply partial mapping results, for instance, when a user query asks for a service with two input elements and the closest available service in the registry requires three elements.

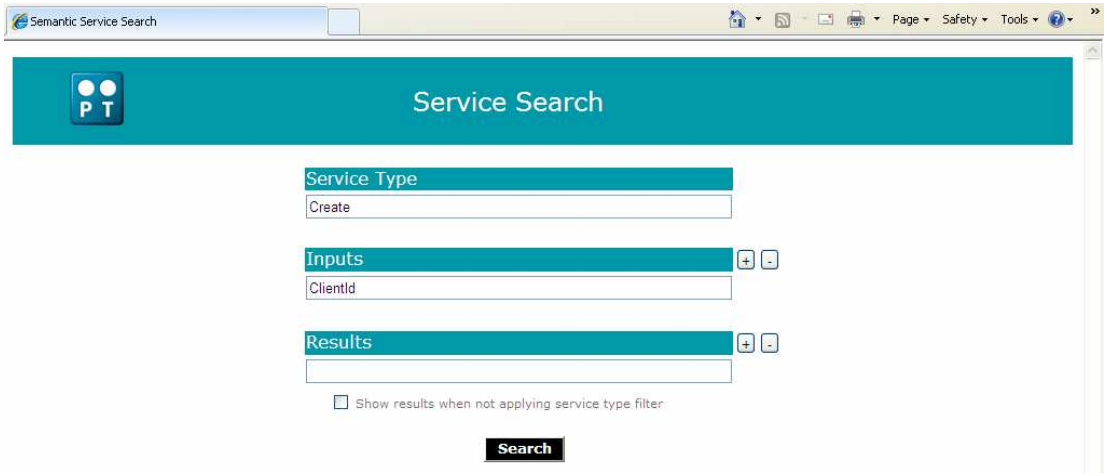

Figure 5. User query interface for service search.

We modified the SAWSDL-MX API to obtain more detailed results about semantic matching of individual I/O elements in order to address some particular cases such as when the published service provides more outputs than those asked for, in which case the user can call the service and discard the unneeded outcome or in turn the service retrieved needs more input parameters than those of the query in which case s/he could find alternative ways to provide them. Thus, modifications implied the definition of new mapping types to encompass situations where only part of the query service elements match those of the published service, in which case the total mapping between inputs or outputs is labeled as "Incomplete" and partial input, corresponding output, mapping is as follows:

- Semantic degree is labeled "Not necessary" when a particular query input parameter matches none of the published service input parameters, or the published service has an output that is not included in the set of query service outputs;
- Semantic degree is labeled "Not provided" when a published service input matches none of the query service input parameters, or a particular query output is not supplied by a published service.

After this semantic mapping process, results are filtered according to query service category if any; the application retains mapped published services which annotated service category is subsumed by the query service category. The presentation of results includes also non-functional information details of mapped published services extracted from the UDDI service registry, such as service textual description, supplier identification and service category.

Example of semantic search of OM API services

Let's consider a search query for OSS/J OM API operations enabling to create a provisioning order with *ClientId* input parameter (Figure 5); one of the query results is the *tryCreateAndStartRequestsByValues* operation (Figure 6) which input is annotated with *RequestValue* concept that is more general than *ClientId*. This operation also has an output annotated with *RequestkeyResult* concept that was not requested in the query. As shown in the ontology portion, this operation is categorized under the *Create* service type (Figure 7).

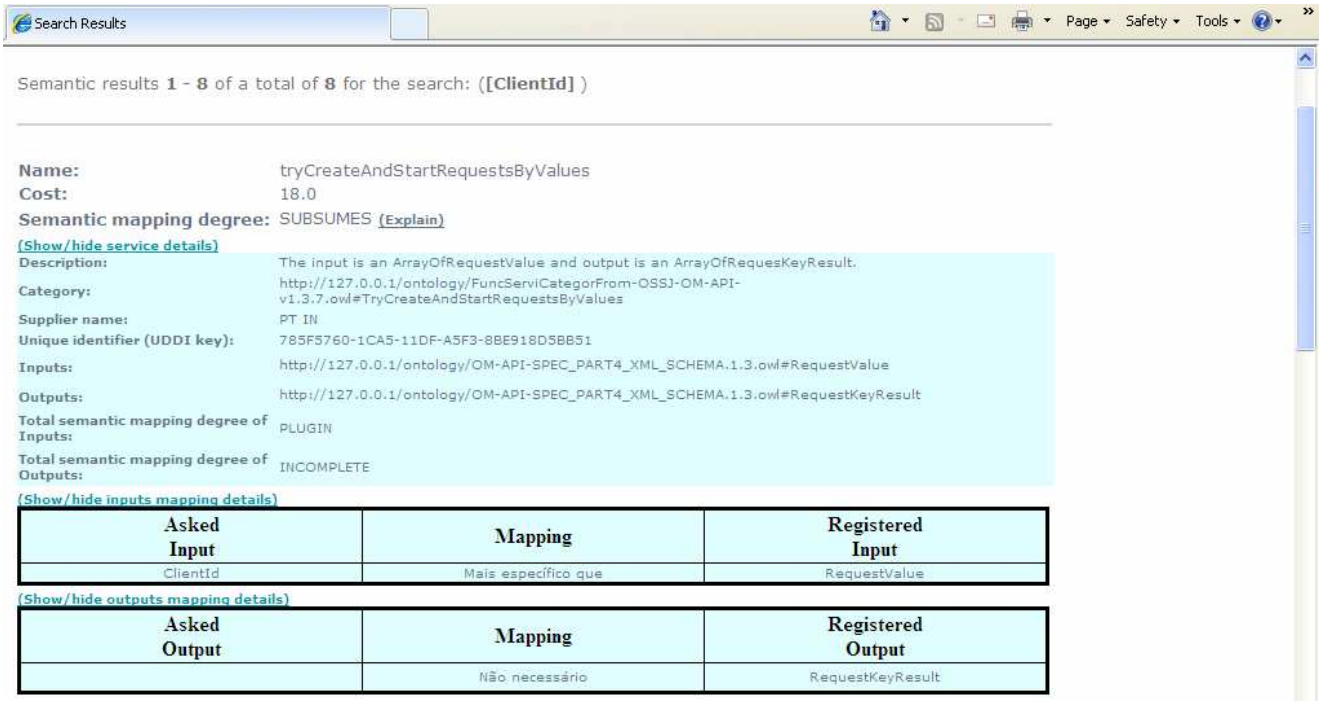

Figure 6. Screenshot of part of the semantic results, showing the *tryCreateAndStartRequestsByValues* operation details.

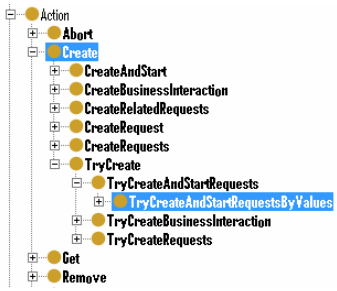

Figure 7. Ontology portion where *tryCreateAndStartRequestsByValues* is categorized under the Create service type.

DISCUSSION AND LESSONS LEARNED

This research and applied work shows how a particular OSS department of a telecom company can benefit from a set of semantic technologies in order to effectively find Web services in a Semantic Service Registry, even if the query does not employ the actual service names but rather a similar business meaning. However, there was a long away to achieve this result, as the domain ontology attainment and the service annotation processes are mainly manual, prone to errors and time consuming tasks.

In our work with the telecom company we have learned three lessons regarding the Semantic Service Registry that are important to emphasize.

Lesson learned #1: Obtaining domain ontologies is a hard task, since we need to represent shared domain knowledge that should be consensual so that users are willing to then use it to annotate services. In the present study, as the original structure of the OSS/J OM API based ontology was shallow we needed to semantically enrich it to further use it to expressively annotate services. Furthermore, to extend this work to the other OSS/J API can be a huge task.

Lesson learned #2: The service annotation process is very domain knowledge dependent. The persons involved in this task need to understand telecommunication in OSS context so as to ensure I/O operation elements are correctly connected with semantic related ontology elements. Particularly, several OSS/J OM API operations are difficult to semantically annotate as a lot of them are very regular, generic and abstract thus hard to differentiate. If operations were similarly annotated this could hardly contribute to an efficient semantic retrieval process.

Lesson learned #3: Search results in the registry can be enhanced by accounting for hits with partial mappings between service inputs or between service outputs. For instance, cases such as when the published service provides more outputs than those specified in the search (in which case the user can call the service and discard the unneeded ones), or cases of services requiring more input parameters than those specified in the search (in which case the user can consider also providing them).

CONCLUSIONS AND FUTURE WORK

We present a proof of concept of Semantic Service Registry and point out the importance of discovering partial mappings between service elements. Our novel approach on semantic matching of web services is implemented in the context of an OSS department of a telecom company which is gradually migrating to an SOA.

We explain how we generated telecom domain ontologies to semantically annotate Order Management API services. These annotations allow us to later benefit from semantic technologies to effectively retrieve those services. For this we adapted the SAWSDL-MX matchmaking tool that relies on description logics inference mapping. The main improvement modifications are related to enabling to provide a matching degree even when the cardinality of the service elements of the query and the cardinality of the service elements being tested differ and also to support searching services by their functional category.

The use of a semantic service registry poses new socio-technical challenges. There are normal governance considerations, such as the allocation of decision rights to add, change, update and retire services, but also specific issues regarding how the services are semantically described and by whom. The reference ontology has to be agreed on and maintained. Decisions have to be made on whether developers become responsible for annotating their services or if a dedicated team should be constituted. Resistance to change may arise, so reward mechanisms should be discussed in tandem with the technological solutions to ensure effectiveness.

On a different level, although our present experiments focus on the use of the semantic registry at process design-time, its use for run-time dynamic service discovery and binding is being considered in the longer term. In that case, selecting from multiple matches returned by a service search is not trivial. A rules mechanism must be able to account for both technical and business concerns, such as simultaneous interoperation of interdependent services, quality of service, and cost. Processing performance of this kind of computation will also be an issue.

REFERENCES

1. Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.-T., Sheth, A. and Verma, K. (2005) Web Service Semantics - WSDL-S W3C Member Submission 7 November 2005 Version 1.0, from http://www.w3.org/Submission/2005/SUBM-WSDL-S-20051107/1.
2. Baader, F., Calvanese, D. and McGuinness, D. (Eds.) (2003) *The description logic handbook; theory, implementation, and applications*. Cambridge University Press.
3. Beckett, D. (2004) RDF/XML Syntax Specification (Revised), *W3C Recommendation 10 February 2004*, from http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.
4. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E. and Yergeau, F. (2006) Extensible Markup Language (XML) 1.0 (Fourth Edition), *W3C Recommendation 16 August 2006*, from http://www.w3.org/TR/2006/REC-xml-20060816/.
5. Brickley, D. and Guha, R.V. (2004) RDF Vocabulary Description Language 1.0: RDF Schema, *W3C Recommendation 10 February 2004*, from http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.
6. Bruijn, J. d., Fensel, D., Keller, U., Kifer, M., Lausen, H., Krummenacher, R., Polleres, A. and Predoiu, L. (2005) Web Service Modeling Language (WSML) W3C Member Submission 3 June 2005, in Jos de Bruijn and Holger Lausen (Eds.) *W3C Member Submission,* from http://www.w3.org/Submission/2005/SUBM-WSML-20050603/.

7. Burstein, M., Ankolenkar, A. and Paolucci, M. (2003) DAML-S: Semantic markup for Web services, The DAML Services Coalition, from www.daml.org/services/daml-s/0.9/daml-s.html.
8. Buschmann, A., Ebbert, A., Raymer, D., Dillon, E., Gauthier, P., Milham, D., Pedneault, M., Perrot, V., Plutino, A., Reilly, J. and Wilmes, J. (2006) The OSS through Java™ API Roadmap Version 3.1, TeleManagement Forum, OSS through Java (OOS/J) Initiative, from http://www.ossj.org/downloads/docs/wp_ossj_api_roadmap.pdf.
9. Chinnici, R., Moreau, J.J. and Ryman, A. (2007) Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Recommendation 26 June 2007, from http://www.w3.org/TR/2007/REC-wsdl20-20070626.
10. Cunha, P. R., Melo, P. and Ferreira da Silva, C. (2009) A Funny Thing Happened on the Way to SOA: Insights from a Three-Year Experience with a Telecom Company, in Grace A. Lewis, Dennis B. Smith, Ned Chapin and Kostas Kontogiannis (Eds.) P*roceedings of the t International Third Workshop on a Research Agenda for Maintenance and Evolution of Service-Oriented Systems (MESOA 2009)*, September 20-26, Edmonton, Canada, CMU/SEI-2010-SR-004, 41-51.
11. Dillon, E., Uzieblo, A., Vermaas, G., Millatiner, A. and Wilmes, J. (2007) JSR264 - Order Management API Final Release OSS through Java™ Initiative, from http://www.tmforum.org/SOAEnablementOSSJ/4492/home.html.
12. Dimitrov, M., Simov, A., Momtchev, V. and Konstantinov, M. (2007) WSMO Studio – a Semantic Web Services Modelling Environment for WSMO (System Description), in Enrico Franconi, Michael Kifer and Wolfgang May (Eds.) *Fourth European Semantic Web Conference*, June 3-7, Innsbruck, Austria, Springer-Verlag Berlin, Heidelberg, 749-758.
13. Domingue, J., Cabral, L., Hakimpour, F., Sell, D. and Motta, E. (2004) IRS III: A Platform and Infrastructure for Creating WSMO based Semantic Web Services, in Christoph Bussler, Dieter Fensel, Holger Lausen and Eyal Oren (Eds.) *WSMO Implementation Workshop*, Frankfurt, Germany.
14. Farrell, J. and Lausen, H. (2007) Semantic Annotations for WSDL and XML Schema W3C Recommendation 28 August 2007, from http://www.w3.org/TR/2007/REC-sawsdl-20070828/.
15. Galliers, R.D. (1992) Chosing Information Systems Research Approaches, Blackwell Scientific, Oxford, UK.
16. Galliers, R.D. (1994) Points in Understanding Information Systems Research, *Systemist*, 16:1, 32-40.
17. García, R. (2005) A Semantic Web approach to Digital Rights Management, PhD thesis, Universitat Pompeu Fabra, Barcelona.
18. Guarino, N. (1998) Formal Ontology in Information Systems, IOS Press, Amsterdam, Netherlands.
19. Haller, A., Gomez, J.M. and Bussler, C. (2005) Exposing Semantic Web Service principles in SOA to solve EAI scenarios, *WWW 2005 conference*, Chiba, Japan.
20. Jaeger, M., C. , Rojec-Goldmann, G., Liebetruth, C., Mühl, G. and Geihs, K. (2005) Ranked Matching for Service Descriptions Using OWL-S, *Kommunikation in Verteilten Systemen 2005 (KiVS 2005)*, 91-102.
21. Kaufer, F. and Klusch, M. (2006) WSMO-MX: A Logic Programming Based Hybrid Service Matchmaker, *Proceedings of the European Conference on Web Services (ECOWS 2006)*, December 4-6, Zurich, Switzerland, IEEE Computer Society Washington, DC, USA, 161-170.
22. Klusch, M. (2008a) Semantic Web Service Description, in Michael Schumacher, Helko Schuldt and Helkki Helin (Eds.) *CASCOM: Intelligent Service Coordination in the Semantic Web*, Birkhäuser Basel, 3, 31-57.
23. Klusch, M. (2008b) Semantic Web Service Coordination, in Michael Schumacher, Helko Schuldt and Helkki Helin (Eds.) *CASCOM: Intelligent Service Coordination in the Semantic Web,* Birkhäuser Basel, 4, 59-104.
24. Klusch, M., Fries, B. and Sycara, K. (2006) Automated Semantic Web Service Discovery with OWLS-MX, *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, May, 8-12, Hakodate Japan, ACM New York, NY, USA, 915-922.
25. Klusch, M., Fries, B. and Sycara, K. (2009) OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services, *Web Semantics: Science, Services and Agents on the World Wide Web*, 7:2, April 2009, 121-133.
26. Klusch, M. and Kapahnke, P. (2008) Semantic Web Service Selection with SAWSDL-MX, in Rubén Lara Hernandez, Tommaso Di Noia and Ioan Toma (Eds.) *Proceedings of the Second International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web*, October 27, Karlsruhe, Germany, CEUR Workshop Proceedings 416, from http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-416/paper1.pdf.

27. Kourtesis, D. and Paraskakis, I. (2008) Web Service Discovery in the FUSION Semantic Registry, in Witold Abramowicz and Dieter Fensel (Eds.) *Business Information Systems*, Springer Berlin Heidelberg, 7, 285-296.
28. Lara, R., Polleres, A., Lausen, H., Roman, D., Bruijn, J. d. and Fensel, D. (2005) A Conceptual Comparison between WSMO and OWL-S, DERI, from http://www.wsmo.org/2004/d4/d4.1/v0.1/20050106/.
29. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N. and Sycara, K. (2004) OWL-S: Semantic Markup for Web Services W3C Member Submission 22 November 2004, in David Martin (Ed.) *W3C Member Submission,* from http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/.
30. Martin, D., Hodgson, R., Horrocks, I. and Yendluri, P. (2006) Submission Request to W3C: OWL 1.1 Web Ontology Language, from http://www.w3.org/Submission/2006/10/.
31. McCutcheon, G., and Jurg, B. (1990) Alternative Perspectives on Action Research, *Theory into Practice*, 24:3 Summer.
32. McGuinness, D.L. and Van Harmelen, F. (2004) OWL Web Ontology Language Overview W3C Recommendation 10 February 2004, in Deborah L. McGuinness and Frank van Harmelen (Eds.), from http://www.w3.org/TR/2004/REC-owl-features-20040210/.
33. Miles, M.B. and Huberman, A.M. (1994) Qualitative Data Analysis: An Expanded Sourcebook, Sage publications, Newbury Park, California, USA.
34. OWL2-Overview (2009) OWL 2 Web Ontology Language Document Overview W3C Recommendation 27 October 2009, in W3C OWL Working Group (Eds.), from http://www.w3.org/TR/2009/REC-owl2-overview-20091027/.
35. Paolucci, M., Kawamura, T., Payne, T. and Sycara, K. (2002) Semantic Matching of Web Services Capabilities, in Ian Horrocks and James Hendler (Eds.) *The Semantic Web — ISWC 2002,* , Springer Berlin / Heidelberg, Sardinia, Italy, 333-347.
36. Patil, A., Oundhakar, S., Sheth, A. and Verma, K. (2004) METEOR-S Web Service Annotation Framework, in *The Thirteenth International World Wide Web Conference (WWW 2004)*, Association for Computing Machinery, Inc. (ACM), New York, USA.
37. Roman, D., Keller, U., Lausen, H., Bruijn, J. d., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C. and Fensel, D. (2005) Web Service Modeling Ontology, *Applied Ontology Journal*, 1, 1/2005, 77-106.
38. Scicluna, J., Abela, C. and Montebello, M. (2004) Visual Modelling of OWL-S Services, in *IADIS International Conference WWW/Internet*, Madrid, Spain.
39. Srinivasan, N., Paolucci, M. and Sycara, K. (2004) Adding OWL-S to UDDI, implementation and throughput, in Proceedings of first International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004).
40. TMF. (1988) TeleManagement Forum, from http://www.tmforum.org/.
41. Van der Vlist (2002) E. *XML Schema* O'Reilly Media, Inc.
42. W3C (2004) W3C Semantic Web Activity, from http://www.w3.org/2001/sw/.
43. Zaremba, M. and Oren, E. (2005) WSMX Execution Semantics. WSMX Working Draft D13.2 v0.2.