Accessing and Manipulating Ontologies Using Web Services

Olivier Dameron, Natalya F. Noy, Holger Knublauch, Mark A. Musen

Stanford Medical Informatics, Stanford University,
251 Campus Drive, x-215, Stanford, CA 94305, USA
{dameron, noy, holger, musen}@smi.stanford.edu

Abstract. Ontologies and Semantic Web Services are the two core technologies of the Semantic Web. The Semantic Web hinges on the ability of computer programs to perform some task involving the autonomous resolution of semantic issues. This ability requires providing standard access for software to ontologies. Moreover, for the Semantic Web to gain widespread acceptance, it needs to reach a critical mass of applications that can interact. This last point requires providing standard access to functionalities for manipulating ontologies. Therefore, it is relevant to bring ontologies and Web Services together by providing access to ontologies through Semantic Web Services. We analyse different kinds of ontology-manipulation functionalities that could be implemented as ontology Web Services (OWS). We then propose an architecture allowing programs to insert calls to ontology Web Services into the more general framework of Web Services. We show that this architecture is a necessary complement to OWL-S for Semantic Web applications to perform dynamic discovery and invocation of Web Services, thus addressing a key requirement of the Semantic Web. We then demonstrate the scalability of our architecture as it allows the composition of (ontology) Web Services for performing complex tasks.

1 Web Service Access to Ontologies

Ontologies and Semantic Web Services are arguably the two core technologies of the Semantic Web. Ontologies provide the backbone of the Semantic Web, defining the semantics of the data and Web resources. Web Services enable programs to call functions provided by a remote server. The structure of the parameters and of the result of the function are represented in an explicit way so that the service can be invoked by any client. Web Services, albeit not yet Semantic Web Services, have become important components of business applications.

Currently however, most examples of Web Services (and in fact Semantic Web Services) operate on *data* that conforms to some schema or ontology. At the same time, ontologies themselves are first-class objects on the Semantic Web. Therefore, we believe that any infrastructure for Semantic Web Services will be incomplete without components to access and manipulate ontologies themselves through Web Services. Such access and manipulation include query of ontology

information, generation of ontology views, translation of ontologies between formalisms, mapping and alignment of ontologies, ontology versioning, reasoning services, and so on. Applications in general, and business applications on the Semantic Web in particular, not only need to access these services in isolation, but also, and more important, need to have these services composed automatically. Such a feature requires a machine-processable description of the semantics of the services (*i.e.*, what they do, how they do it, what kind of parameters and results are necessary to communicate with them beyond simple syntactic considerations). In turn, this description requires some tools to be processed.

To consider a simple example, suppose we are running an e-Commerce site powered by Semantic Web technologies performing price-comparison for books, CDs, and DVDs, compare.com. The site has its own ontology of products, their properties and relations, represented in OWL. In order to perform the price comparison, we use Web Services to request price and availability information from various sites selling similar products. To use this mechanism, though, we must perform several ontology-level operations. First, we need to *map* relevant parts of our own ontology of products in which visitors to our site are interested to ontologies of different sites from which we request information. If some of those ontologies are represented in languages other than OWL, we must *translate* these ontologies into OWL. After we have obtained a mapping between our site's ontology and one of our usual providers, books.com, the ontology of the provider may have evolved. Thus, we need to find out how it has *changed* and whether or not this set of changes affects the mapping that we have stored, and if it does, then to what extent.

An infrastructure that would make this scenario feasible is an infrastructure of Semantic Web Services that can perform these ontology-level operations, taking one or several ontologies as inputs and producing mappings, diffs, translations of ontologies as outputs. We call this infrastructure for ontology access and manipulation through Semantic Web Services *Ontology Web Services (OWS)*.

This "marriage" of ontologies and Semantic Web Services has benefits for both sides. On the one hand, by providing Web Services access to various ontology-manipulation operations and supporting composition of these Web Services, we enable business applications to use ontologies just as they use data, through the same communication protocols and technologies that they use to access other services. On the other hand, by specifying protocols and infrastructure for ontology manipulation and access, we can streamline the variety of ontology-management tasks for which many different tools and algorithms have sprung up in recent years by providing a unified infrastructure, specifying inputs, outputs and effects of each operation and facilitating automated composition of ontology Web Services based on their semantic descriptions.

In this paper, we identify some of the major classes of ontology Web Services, their inputs, outputs, capabilities, and effects. We propose an architecture that allows us to treat OWS as a special class of Semantic Web Services. We then show the scalability of this architecture by discussing how OWS can be composed to implement complex ontology services. We began implementing prototypes of

some of these services, but the goal of this paper is to outline the framework that other providers of OWS can contribute to.

2 Ontology Web Services capabilities

We envision several broad categories of ontology access and manipulation through Web Services:

- Query access to ontologies;
- Generation of ontology views;
- Translation of ontologies from one language to another;
- Management of multiple ontologies;
- Reasoning on ontologies.

The list is not exhaustive and it will grow larger as use of SWS with ontologies becomes more popular. Moreover, realistic usage of OWS will involve a combination of the categories above. For each category of services, we specified which ontologies such web service takes as input; what type of access it provides; what types of outputs it has. Table 1 summarizes the results and we discuss the details in the rest of this section.

Unless stated explicitly, *ontology* refers to a single file as well as to the result of composing or merging several files [14]. Moreover, we do not make any assumption about the representation language for ontologies, or about the specific protocols used for Web Services. We do not specify whether requesters of OWS pass ontologies as files (by value) or as URLs to their location on the Web (by reference). Therefore, we assume that ontologies can be passed as parameters either by value or by reference.

2.1 Query access

Query access to an ontology is probably the most traditional access to structured information. In general, for this type of services we enable the requester to specify the ontology to be queried, the query itself, the query language used by the query, and, possibly, a specific query engine to use. A query engine typically returns a set of variable bindings associating the variables of the query with the possible classes or relationships.

For example, if a user of our compare.com site introduced in Section 1 wanted to find all books and their titles that had movies on DVDs with the same title, a query in a typical Semantic Web query language would request "all books $?x$ with title $?t$ such that there is a DVD $?y$ with title $?t$." The result of the query would be a set of possible bindings for the variable $?x$ and $?t$. Note that the result specifies only the matching concepts and values, but does not contain any information on the semantics of these concepts. In a Description Logics context, such queries can also return the individuals that instantiate the class refined by the query [7].

Service type	**Input**	**Output**
Queries	– query language (possibly specified by the service) – query engine – query string – ontology	– variable bindings
Views	– ontology or ontology reference – view definition (possibly the service name)	– subontology representing the view
Translations	– ontology (possibly fixed by the service) – target language	– ontology in the target language
Mapping	– source ontology – target ontology – initial mapping	– mapping definition
Versioning	– ontology version 1 – ontology version 2	– diff between versions
Merging	– ontology 1 – ontology 2 – mapping	– the merged ontology
Reasoning	– ontology or ontology reference to be classified – type of classification	– ontology or ontology reference (classification result) possibly bearing a classification flag

Table 1. Type of the input and output parameters for the various types of Web Services

There are a number of query languages for Semantic Web languages, such as XQuery, RDQL, RQL [8,9], TRIPLE [20], and DQL, among others. These languages are based on database query languages such as SQL and provide bindings for values in the query as their result. Note that the semantic processing of the query may vary depending on the language: RDQL does not take into account RDFS semantics, whereas RQL does [21]; and TRIPLE can handle multiple semantics, whereas RQL is limited to RDFS [20].

The goal of OWS is not to replace these languages, but to encapsulate them. Thus, OWS provide a standard communication layer that allows any application (1) to send its query and the ontology to be queried to a remote application dedicated to computing the result, and (2) to retrieve this result in a potentially structured way, even though the two applications do not share the same address space.

2.2 Ontology Views

One of the original motivations behind ontology research was the ontologies' ability to help with reuse in knowledge representation [5]. By virtue of being formal and explicit representations of a domain, ontologies could be shared domain descriptions that different applications and agents use. However, while many large standard ontologies provide a shared, tested, and well-accepted vocabulary, many of them pose a formidable challenge to Semantic Web users: these resources are huge, often containing tens of thousands of concepts. However, many Semantic Web users need only a small fraction of the resource for their application. Currently such a user still needs to make sense of the complete resource, importing it as a whole, and dragging along this extra "baggage" of concepts, most of which he is never going to use. In addition to the cognitive challenge of understanding the large resource (or at least figuring out which swaths of it are not relevant and can be safely ignored), there can be a computational penalty as well if the user needs to perform reasoning with the whole ontology, and the cost of downloading and storing a large resource.

An *ontology view* provides a self-contained subset of an ontology based on the user's specification [17]. View specification depends on the view-definition language but it generally defines a subontology of the source ontology that contains the concepts that the user is interested in.

This notion of a view is different from views in databases where views are defined as SQL queries. The binding for variables essentially provides a definition of a new virtual table or set of virtual tables in a database [6]. However, such an approach can't be transposed to ontologies: the binding for variables is a set of the identifiers of the concepts that match the query, and not a set of concepts. This point has been made particularly clear by Volz and colleagues: "*From the perspective of relational and object-oriented databases, it is natural to consider views as arbitrary stored queries. This is not apt for RDF*" [21]. Similarly, retrieving the individuals who instantiate a query on an ontology [7] is not computing a view on this ontology.

If the variable binding resulting from a query can be seen as new tables for a database, they are by no mean a portion of an ontology or a restriction of its domain. Variable bindings are not class or property definitions; they bear no semantics.

Researchers have recently proposed several approaches for specifying ontology views. For example, Volz and colleagues [21] define a view language based on the RQL query language [8]. In this framework, a view represents a new class or property in the ontology. The authors restrict the RQL queries that can be used in the view to ensure that the view returns only unary or binary relations (classes or properties respectively).

Magkanaraki and colleagues [11] take the approach of defining query-based views further. In their RVL language, which also uses RQL for querying ontologies, the authors propose mechanisms for restructuring the original class and property hierarchies, allowing the creation of new resources, property values, classes, or properties. Therefore, a view definition includes not only the query itself, but also a set of statements that define these new structures, linking them to the query results.

Noy and Musen propose a concept of a Traversal View [17], where a subontology is specified by defining concepts of interest, the relationships to traverse to find other concepts to include in the view, and the depth of the traversal. For example, suppose a web site (another one that compare.com may want to use) provides an extensive ontology of book topics, with relationships between topics such as `broader`, `narrower`, `related-to`, `subtopic-of` and so on. And suppose we are only interested in the topic of Semantic Web, its subtopics, and its closely related topics. A Traversal View corresponding to such specification will specify `Semantic Web` as the topic of interest, request all the concepts related to this topic by `subtopic-of` link and all the topics that are reachable from the Semantic Web topic by traversing the `related-to` link no more than 3 times. The resulting subontology will include definitions of all classes encountered in such traversal.

In the context of OWS, a request for a web service providing an ontology view contains the source ontology and the view definition. The result is also an ontology—in this case a subontology of the source ontology corresponding to the view (or a reference to it).

2.3 Translation

Ontologies can be represented in different ontology languages such as RDF(S), OWL, or formats specific to ontology-development tools, such as Protégé.[1] Some ontologies and terminologies are expressed in XML Schema or UML. While recent initiatives in W3C have proposed standards for ontology languages, ontologies represented in different languages will probably continue to exist on the web for a while. Furthermore, different languages espouse different modeling

[1] `http://protege.stanford.edu`

paradigms, such as Description Logics, or Frames, and choosing the more appropriate one for the task at hand is often a compromise between architectural choices, computational constraints and expressiveness requirements. Therefore, different application may need to access different representations of the same ontology or to translate an ontology available in one language into another language.

Translation services in OWS convert an ontology from one representation language to another. Note that in general such services provide a syntactic transformation [4], and not a mapping between two different ontologies.

Such services can be completely generic, performing the conversion based on general semantic mapping between primitives of the two languages. There are many initiatives currently underway to create such mappings for popular languages on the web. For instance, a simple search on google returns a large number of efforts to create a mapping between OWL and UML. However, because any two languages usually have different expressiveness, some information may be lost in the translation process. For example, when converting an ontology represented in language with meta-modeling capabilities, such as many frame-based languages (Protégé is an example of an ontology tool using such a language), to a Description Logics language, such as OWL DL, we may not be able to faithfully represent the metaclasses. In the previous example, note that this limitation is due to Description Logics expressivity, ant not to the tool, as there is an OWL plugin for Protégé.

While translation services that perform generic syntactic transformation based on the source and target languages are necessary, we also envision translation services can be dedicated to a particular ontology. This approach would enable more accurate translation that takes into account semantics of the source ontology. Consider for example the Foundational Model of Anatomy (FMA)—a declarative representation of anatomy developed at the University of Washington [19]. The ontology represents the result of manual and disciplined modeling of the structural organization of the human body. While the FMA ontology is a relatively recent development, many users in medical informatics already consider it to be a tremendous resource that will facilitate sharing of information among applications that use anatomy knowledge.

The FMA was developed in a frame-based formalism and uses metaclasses extensively [18]. Translating parts of it into OWL DL, for example, is not trivial and would require translations that are specific to FMA. Given the popularity of the FMA and the desire of many groups to use it in the DL context, it would make sense to have a special translation service that knows the semantics of the structures in the FMA and performs the translation that takes advantage of the modeling principles structuring the FMA but that couldn't be represented in the original frame-based format.

Note that as simple as they seem to be, translation services blur the line between "simple" services and composed ones. Particularly, a server providing a dedicated translation service could call a remote generic translation service, and focus on the domain-specific tuning. Moreover, translation services can be

associated to view services, for example, for providing an OWL representation of a specific view on the FMA, such as heart and its surrounding organs.

2.4 Management of multiple ontologies

The Semantic Web relies on the explicit annotation of resources' semantics based on ontologies, in order to enable the manipulation of these resources by software. At the same time, the Semantic Web approach is based on federating and combining these ontologies, rather than creating a centralized standard set of ontologies. Therefore, many, often overlapping ontologies or different versions of the same ontology exist on the Semantic Web and applications need to establish correspondences between multiple ontologies, recognize where they are different, compose them, or handle versioning issues.

Providing ontology mapping, merging, and versioning of ontologies through Web Services enables applications to handle semantic heterogeneity dynamically. A number of tools for ontology mapping, merging, and versioning exist (the Prompt plugin for Protégé is one such example [16]). These tools partially automates ontology mapping, merging, and versioning. Wrapping these components into Semantic Web Services will make them a part of the general OWS infrastructure, not only enabling web-service access to them, but also allowing them to be composed with one another and with other services within OWS.

Mapping Mapping two ontologies is establishing explicitly the semantic correspondence between them. It includes finding the semantic equivalence, or at least subsumption between the concepts of the source and those of the target ontologies [4], and representing these correspondences declaratively. Researchers have developed a number of ways to represent mappings declaratively. Some examples include representing mappings as instances in an ontology of mappings; defining bridging axioms in first-order logic to represent transformations; and using views to describe mappings from a global ontology to local ontologies. When restricting the scope from the Semantic Web in general to reasoning about OWL-S descriptions of Web Services, Burstein emphasizes the need for mapping capabilities in several stages of the process [2,3].

Generally, this mapping process can be only partially automated. Conceptual ambiguity and lack of explicit axiomatisation have to be addressed manually by domain experts and knowledge engineers [13]. However, there are many cases where the application requires only the part of the mapping that can be generated automatically or where an application can tolerate a certain level of imprecision in the mappings. In general, users are already accustomed to some levels of uncertainty on the web: if we are searching for prices for airline tickets we know that we may not get the lowest possible price, but will be satisfied once we find a price that falls within our budget.

Merging Because developing a single global ontology is neither scalable nor maintainable, the Semantic Web relies on developing independent semantically heterogeneous ontologies, and on composing them dynamically.

Therefore, Semantic Web applications may need to access services for combining ontologies. The most simple operation is to perform a simple union of the two ontologies, using namespaces for disambiguation, which is similar to the "import" operator in OWL.

However, Mitra and colleagues advocate the need for a sound and more elaborate set of operations. They propose an ontology algebra performing union, intersection and differences of ontologies [14].

Furthermore, if we have a declarative mapping between two ontologies, we can use this mapping to perform a more informed merge rather than a simple union [1]. We can merge concepts that are declared as equivalent in the mapping, and so on.

Therefore, in OWS, a merging service takes not only two ontologies to be merged as input, but also a mapping between ontologies if it exists.

Versioning Ontology versioning consists of providing mechanisms for storage and comparison of different versions of the same ontology [15]. Whereas mapping applies to two different ontologies and focus on the similarities, versioning deals with only one ontology and emphasizes differences between two versions.

In OWS, a service that supports ontology versioning would take two versions of an ontology and return a structural diff of the versions, identifying concepts that were added, deleted, or changed and specifying how they were changed. Klein and Noy, for example, proposed an ontology of changes based on the OWL language [10]. A diff returned by an OWS service can be a set of instances in this ontology.

2.5 Reasoning

Reasoning encompasses all kinds of semantically-based manipulations of one or more ontologies. Reasoning can be used for computing the subsumption hierarchy of an ontology, determining the classes of which an individual is an instance of [7], or checking the ontology's consistency, making sure that all the classes are satisfiable. Eventually, reasoning can be used more dynamically in order to answer queries such as query access (Section 2.1) or views (Section 2.2).

Features of reasoner depends more heavily on the language in which the ontology is represented than on the reasoning task itself [20]. Classifiers such as Racer[2] or FaCT++[3] can take advantage of the set-based semantics of Description Logics for inferring subsumption between classes or for classifying individuals. Processing rules such as RuleML[4] [12] or SWRL[5] requires first-order-logics provers such as Hoolet[6]. For frame-based ontologies, reasoning capabilities relying on first order logics are also available[7].

[2] `http://www.cs.concordia.ca/~haarslev/racer/`
[3] `http://owl.man.ac.uk/factplusplus/`
[4] `http://www.ruleml.org`
[5] `http://www.daml.org/2003/11/swrl`
[6] `http://owl.man.ac.uk/hoolet/`
[7] `http://protege.stanford.edu/plugins/domain_reasoning.html`

In the OWS framework, reasoning services are also "wrapped" as Semantic Web Services and become part of the general architecture. The requester of a reasoning service specifies the ontology to perform reasoning on and also a specific type of reasoning required. The result of the service depends on the reasoner but could be for example a classified version of the ontology with additional metadata indicating that the ontology does not require any additional processing (i.e., it is post-coordinated).

2.6 Composition of Services in the OWS Framework

In the previous section, we have described an array of possible ontology services and their capabilities. This list is only a first pass at a comprehensive specification of ontology Web Services. However, even this initial specification highlights the main advantage of bringing together ontologies and Semantic Web Services: we can now use the web-services composition capabilities to compose automatically various services to perform complex tasks.

Consider for example the group of services in OWS that support management of multiple ontologies (Section 2.4). Most mapping and merging services require that source ontologies are represented in the same ontology language. Therefore, if the service receives two ontologies represented in different languages, it can call another service within the OWS framework to translate one of the ontologies into the language that the other one uses.

We will need to compose versioning services (Section 2.4) with most other services to determine for example if an ontology view or a query result that we have generated based on an earlier version of an ontology is still valid for the new version of the ontology or if it needs to be regenerated.

We may want to perform multiple-ontology services such as mapping and, in particular, merging not on a whole source ontology but on some subontology. Therefore we will need to compose a service that provides an ontology view with an ontology-merging service.

As any Web Services, OWS can be used directly by Web Services middleware. In the more general Semantic Web context, it is also important to enable the automatic discovery, invocation and composition of Web Services, and particularly of OWS. In the next section, we propose an architecture based on semantic descriptions of OWS. Such descriptions are necessary for leveraging the use of OWS by Semantic Web Services, and for providing some of the tools necessary for processing the semantic descriptions of Semantic Web Services in Semantic Web context.

3 Architecture

OWL-S aims at providing a computer-interpretable description of a web service, and ways to access it. Typically, this description should be used for: (1) automatic Web Service discovery, (2) automatic Web Service invocation, (3) automatic Web

Service composition and interoperation, and (4) automatic monitoring of Web Wervice execution.

The OWL-S description encompasses three types of knowledge about a service. A service presents a *profile* that describes what the service does, based on a domain ontology. It is useful for determining if the service sells books, comics or plane tickets. The *model* describes how the service works by decomposing it into processes. It represents the organization of the processes and their tasks. Eventually, a service presents one or several *groundings* that describe how the service should be used for automatic invocation.

Based on the description of ontology Web Services in Section 2, we notice that:

- in order to be usable autonomously by applications, OWS need to be associated with a description, and OWL-S is a good candidate;
- conversely, the exploitation by a web service client of the semantic information represented in OWL-S descriptions requires some ontology manipulations, and ontology Web Services can provide these manipulations.

OWL-S provides a semantic description of the functions offered by Web Services. OWS provides some functions for processing these descriptions. By proposing to implement these functions as Web Services, we have created an inter-connection between OWL-S and OWS. Far from being a disadvantage, this situation allows us to lay out an architecture that itself *enables* many features in the use of OWL-S descriptions, making OWL-S itself more scalable.

First, we show how the various types of services identified at Section 2 can be integrated into the original OWL-S architecture. Second, we show how this architecture can be enabled by using OWS to process OWL-S descriptions.

3.1 Architecture principles for OWS services

Figure 1 represents the chronological decomposition of the interaction between a web service client and an OWL-S enabled server. The client first requests a service profile (transactions 1 and 2) in order to determine if the service meets its expectations. Note that this determination itself, which often includes ontology comparison or other ontology-manipulation steps, is out of scope of OWL-S, but could use OWS. Section 3.2 presents a simple scenario. If the service is relevant, the model provides the technical information—such as communication protocol or endpoint—for actually calling the service (transactions 3 and 4). Note that these feature can also be part of the client's constraints for deciding if the service is relevant. Eventually, the client actually accesses the service (transactions 5 and 6). The client's query is processed locally on the server and may involve local resources. At this point, unless the model explicitly specifies it, the server can call functions provided by other remote Web Services without the client being aware of it.

The mechanism described above is also valid for specifying ontology Web Services in the OWS framework. It allows a client to look for an OWS server

providing a specific ontology manipulation function, such as the computing of views of a specific ontology, or translation of an RDF ontology into OWL. For computing the answer, the server operates on a local representation of one or several ontologies that he may retrieve dynamically from other sources.

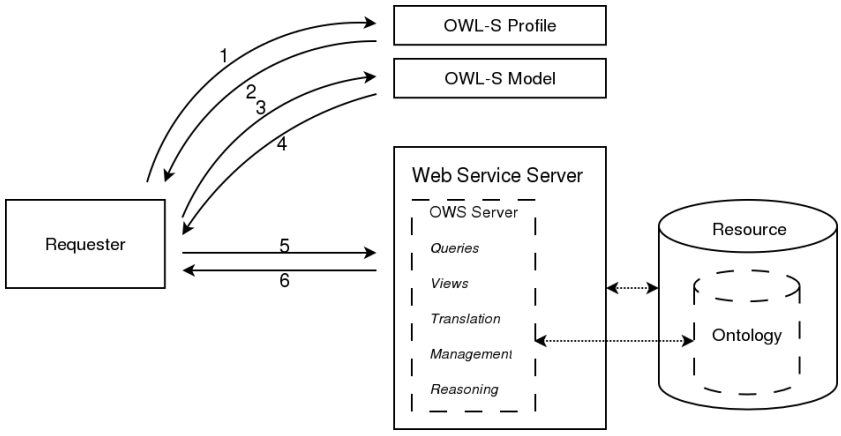

Fig. 1. Interactions between a web service client and and an OWL-S compliant server. Being Web Services themselves, OWS comply with these principles for providing the ontology manipulation functionalities described in Section 2.

3.2 Processing the semantic description of Web Services using OWS

In the previous section, we pointed out that the way the client was supposed to determine whether a service is relevant or not with regards to its expectations was left unspecified. Moreover, the client may need to establish a correspondance between its own semantic workspace (i.e., its ontology and internal data format) and the one of the server for putting the parameters in the correct format and for interpreting the service result. This section shows how OWS can be used to address these issues. The following sequence of interactions between a web service client and several servers is illustrated in Figure 2. Note that this scenario is a generic example: the 14 steps could apply to a lot of real situations, but they are by no mean limitative: some of them may be optional , or additional calls to OWS may be necessary. We tried to illustrate most of them by a client to our e-Commerce service example.

The client first requests a service profile and model (transactions 1 and 2). For the sake of clarity, it combines transactions 1 to 4 of Figure 1. At this point, the client still has to process the semantic description provided by OWL-S.

Using OWS for assessing the relevance of the Web Services In order to decide if the Web Services is relevant, the client relies on a description of what the server does (represented in OWL-S by the *profile*), and possibly on a description of how the server computes its result (represented in OWL-S by the *model*).

For example, let us imagine that the client wants to know if the e-Commerce service sells manga DVD for children, without looking for a particular title. This would require a view about DVD from an ontology of culture media, and that this view has to be translated into a suitable representation formalism. The client would then call a translation service provided by a remote server (transactions 3 and 6), and pass as argument the result of the view retrieval (transactions 4 and 5), obtained from an OWS server. Although this step is not represented in the Figure 2, note that looking for an OWS server providing the view or the translation may in turn require the client to process the OWL-S description of these servers.

Using OWS for semantic interoperability Let us assume that the previous process was successful and that the client has determined that the server was adequate. From the semantic description of the server (the *grounding* in OWL-S), the client is able to infer the expected types of parameters that it has to provide. For example, the client may need to comply with the server-specific way of representing the shipping and the billing address. Transforming these parameters from its internal representation to the one expected by the server requires some ontology mapping, that the client delegates to a third OWS server (transactions 7 and 10). Without the client being aware of it, the mapping server calls a remote reasoner before returning its result (transactions 8 and 9). Again, these steps may involve the processing of the servers' OWL-S descriptions.

Now, the client is able to actually call the expected Web Service (transactions 11 and 12).

Eventually, the client still has to interpret the result, which involve a supplementary call to a remote OWS server (transactions 13 and 14).

4 Conclusions and perspectives

In this paper, we have identified a set of ontology-manipulation functions that enable and facilitate the use of semantic information by applications on the Semantic Web. We proposed the ontology Web Services (OWS) architecture that implements these functions as Semantic Web Services. This integration not only allows the ontology services to be first-class members among Semantic Web Services but also facilitates the process of web-service composition and invocation by providing some of the functionalities that automate this process.

Our work relies on existing standard technologies for ontologies (*e.g.*, RDF(S) and OWL), as well as for Web Services (*e.g.*, SOAP, WSDL, OWL-S). This is an indication that we are switching from an era of design to an era of implementation and that Web Services and ontology Web Services are ready for a

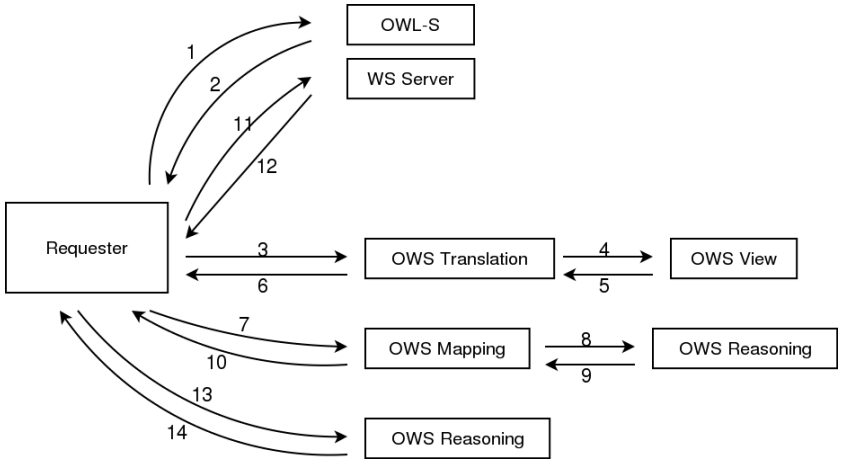

Fig. 2. Using OWS services for processing the semantic description of a Web Services. OWS can be used for assessing the Web Services relevance (transactions 3 to 6) or for achieving the semantic interoperability between the client and the server (transactions 7 to 10 and 13-14)

widespread adoption. We believe that OWS will contribute to this phenomenon by providing access to some of the functionalities that will enable applications to decide dynamically which Web Services to contact and to address the semantic heterogeneity of the web.

We have begun implementing prototypes of some of these services, but most of the implementation is future work. However, our main goal is to identify the types of ontology Web Services and their interfaces, enabling others to implement such services on the Semantic Web.

References

1. P. A. Bernstein, A. Y. Halevy, and R. A. Pottinger. Model management: Managing complex information structures. *SIGMOD Record*, 29(4):55–63, 2000.
2. M. H. Burstein. The many faces of mapping and translation for semantic web services. In *International Conference on Web Information Systems Engineering (WISE 03)*, pages 261–268, 2003.
3. M. H. Burstein. Ontology mapping for dynamic service invocation on the semantic web. In *AAAI Spring Symposium*, 2004.
4. D. Dou, D. McDermott, and P. Qi. Ontology translation on the semantic web. In *Proceedings of the International Conference on Ontologies, Databases and Applications of Semantics*, 2003.
5. T. R. Gruber. *Formal Ontology in Conceptual Analysis and Knowledge Representation*, chapter Toward Principles for the Design of Ontologies used for Knowledge Sharing. Kluwer Academic Publishers, 1993.

6. A. Y. Halevy. Answering queries using views: a survey. *VLDB Journal*, pages 270–294, 2001.
7. I. Horrocks and S. Tessaris. Querying the semantic web: a formal approach. In I. Horrocks and J. Hendler, editors, *Proc. of the International Semantic Web Conference (ISWC 2002)*, pages 177–191. Springer-Verlag, 2002.
8. G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and Scholl. RQL: A declarative query language for RDF. In *Proceedings of the Eleventh International World Wide Web Conference*, pages 592–603, 2002.
9. G. Karvounarakis, A. Magkanaraki, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl, and K. Tolle. Querying the semantic web with RQL. *Computer Networks and ISDN Systems Journal*, 42(5):617–640, 2003.
10. M. Klein and N. F. Noy. A component-based framework for ontology evolution. In *Workshop on Ontologies and Distributed Systems at IJCAI-03*, 2003.
11. A. Magkanaraki, V. Tannen, V. Christophides, and D. Plexousakis. Viewing the semantic web through RVL lenses. In D. e. a. Fensel, editor, *Proceedings of the Second International Semantic Web Conference (ISWC 2003)*, pages 96–112, 2003.
12. C. J. Matheus, M. M. Kokar, K. Baclawski, and J. Letkowski. Constructing RuleML-based domain theories on top of OWL ontologies. In *Rules and rule markup languages for the semantic web Workshop, ISWC*, LNCS, pages 81–94, 2003.
13. P. Mika, M. Sabou, A. Gangemi, and D. Oberle. Foundations for OWL-S: Aligning OWL-S to DOLCE. In *AAAI Spring Symposium*, 2004.
14. P. Mitra, G. Wiederhold, and M. Kersten. Graph-oriented model for articulation of ontology interdependencies. In *Conference on Extending Database Technology*, 2000.
15. N. F. Noy and M. A. Musen. Ontology versioning as an element of an ontology-management framework. *IEEE Intelligent Systems*, 2003.
16. N. F. Noy and M. A. Musen. The PROMPT suite: Interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003.
17. N. F. Noy and M. A. Musen. Specifying ontology views by traversal. In *Proceedings of the Third International Semantic Web Conference (ISWC2004)*, 2004. In Press.
18. N. F. Noy, M. A. Musen, J. J. L.V. Mejino, and C. Rosse. Pushing the envelope: Challenges in a frame-based representation of human anatomy. *Data and Knowledge Engineering*, 48(3):335–359, 2004.
19. C. Rosse and J. L. V. Mejino. A reference ontology for bioinformatics: The foundational model of anatomy. *Journal of Biomedical Informatics.*, 2004.
20. M. Sintek and S. Decker. Triple – a query, inference , and transformation language for the semantic web. In *Proceedings of the International Semantic Web Conference (ISWC 02)*, LNCS 2342, 2002.
21. R. Volz, D. Oberle, and R. Studer. Views for light-weight web ontologies. In *ACM Symposium on Applied Computing*, pages 1168–1173, 2003.