Toward an integrated ontology for Web services

Yassin Chabeb, Samir Tata and Djamel Belaïd
Institut TELECOM; TELECOM & Management SudParis; UMR CNRS SAMOVAR
9 rue Charles Fourier 91011 Evry France
{Yassin.Chabeb,Samir.Tata,Djamel.Belaid}@it-sudparis.eu

Abstract—The lack of semantics in Web Services Description Language (WSDL) prevents automatic discovery and hence automatic invocation and composition. In our work, we are interested in extending existing approaches for the description of Semantic Web Services. Previously, we have extended the W3C recommendation on Semantics for Web Services (SAWSDL) and have proposed the use of two types of ontologies: a Technical Ontology type containing concepts defining semantics of services, their QoS, etc. and a Domain Ontology type containing the concepts defining the semantics of the business domain. The aim of this paper is to present Yet Another Semantic Annotation for YASA4WSDL. This ontology integrates useful concepts of YASA4WSDL description was enriched by references to other efforts made for semantic Web services description. Section IV we present our proposed approach and the mapping techniques used for the construction of a technical ontology based on the integration of ontologies. The last section concludes this paper and presents our future work.

II. RELATED WORK

We present in this section the most related work to our contribution and review the approaches for describing and modeling Web services. We rely on this review to identify the needs to define a technical ontology for Web services.

In [8], SAWSDL suggests how to add semantic annotations to various parts of a WSDL document like input and output message structures, interfaces, and operations. This extension is in line with WSDL extensibility framework. SAWSDL defines a new namespace called “sawSDL” and adds an extension attribute called “modelReference” so that relationships between WSDL components and concepts in another business semantic model (e.g. ontology) are handled. On one hand, in SAWSDL “modelReference” can only annotate concepts in the WSDL metamodel. For example, when a “modelReference” attribute of an operation contains a list of references to some concepts in a business domain one can not know the nature of each reference whether it is a precondition, effect or some thing else. This makes more difficult automatic service discovery and composition based on concepts that do not belong to the metamodel of WSDL. On the other hand, SAWSDL does not require any changes neither in existing WSDL and XML Schema [3] documents nor in the way in which they had been used previously.

OWL-S approach proposes an upper ontology for Web services motivated by the need to provide three essential types of knowledge about a Web service: (i) what does the service provide for prospective clients, (ii) how it is used and (iii) how does one interact with it? The answer to (i) is given in the “profile” which is used to advertise the service. The service profile elements include: preconditions, inputs, outputs, results and service category. The answer to (ii) is given in the “process model” which includes: inputs, outputs, preconditions, effects and the behavior of the service (data and control flow). The answer to (iii) is given in the “grounding”. A grounding provides the needed details about transport protocols.

WSMO [9] is an ontology of services based on WSMF (Web Service Modeling Framework) [10, 6]. WSMO defines four top level elements as the main concepts which have to be described in order to describe Semantic Web services: ontologies, services, mediators and goals. A service is
described in terms of non-functional properties, a provided interface and a provided capability. A goal contains non-functional properties, a requested interface and a requested capability. An interface describes messages sent to/between a service and the visible behavior of that service. A capability includes non-functional properties, pre-conditions, assumptions, post-conditions, and effects.

Another effort in enhancing Web services description with semantics, related to WSMO, is WSMO-Lite [11]. Contrary to WSMO and OWL-S, it adopts a bottom-up approach for modeling services and is built on SAWSDL. WSMO-Lite identifies vocabulary for semantic description and proposes a bridge between WSDL, SAWSDL and domain-specific ontologies. But WSMO-Lite does not prescribe concrete language for functional semantics and as effects and conditions are language-dependent, it cannot specify semantics for them [11].

In [12], the authors propose the use of OWL-S constructs (service profile and process model) as references of SAWSDL annotations. The idea is to continue employing the OWL-S grounding and to adopt a SAWSDL-based perspective. This approach starts from the assumption that atomic processes in OWL-S correspond to SAWSDL’s operations. Then it proposes to add a “modelReference” attribute with message elements as it is not defined by SAWSDL, and finally the authors propose to refer a message’s “modelReference” to an OWL class.

In [13], the Composite Capabilities/Preference Profiles (CC/PP) standard vocabulary [2] is extended and semantic information is added to the description. This W3C proposed standard has a description profile constructed as a two-level hierarchy of a set of components each one having at least one attribute. So in order to describe more service information, new CC/PP components and attribute vocabularies (type of service, quality of service, location, and additional information, etc.) are introduced in [14]. Actually capability is a useful concept but does not satisfy all the user needs for service description such as process aspect of services or operation effects.

In [15, 16, 17], DIANE Service Description (DSD) propose object-oriented service description in order to put into practice additional requirements that are not fulfilled by semantic service description such as WSMO and OWL-S. In [18, 17], the authors describe how DSD is enhancing semantic description by using an ontology to express concepts like precondition and effect.

III. REQUIREMENTS AND MOTIVATIONS

Semantic Web services use business domain ontologies to provide their descriptions in terms of concepts. A business concept is used to annotate service description parts (e.g. operation, interface, etc.). The main objectives of semantic service description approaches are the automation and enhancement of discovery, composition and invocation of services. These objectives are met by OWL-S and WSMO since these approaches describe the service profiles for discovery, the service behavior for composition and the grounding for invocation. But these description languages are “closed” approaches: they only handle respectively OWL and WSMO[19] as type of ontologies. Moreover, the set of concepts they defined is limited and is difficult to extend. Some WSMO concepts do not appear in OWL-S ontology and vice versa. In SAWSDL there is no explicit mention of pre-condition and effects that one can find in WSMO and OWL-S. In addition SAWSDL focuses on the discovery and automatic invocation of services and is not dedicated to describe Web service behavior which is essential for automatic service invocation and composition. This shortage is mainly due to the limited expressiveness of the WSDL meta-model.

SAWSDL is an approach independent of the semantic representation language thanks to the separation of semantic annotation mechanism from the representation of the semantic descriptions. This gives flexibility to the developers’ community to select their favorite semantic representation language, to reuse semantic domain models and annotate descriptions using multiple ontologies. Semantic annotation in SAWSDL use an extended attribute called “modelReference” so that relationships between WSDL components and concepts in another business semantic model (e.g. ontology) are handled. It allows using all types of ontology (OWL, WSML, UML, etc.). And since it is close to WSDL, SAWSDL does not require more effort for developers accustomed to WSDL.

In order to enhance automatic service discovery, composition and invocation, we have identified some requirements for the description of semantic Web services. The SAWSDL specification states that a “modelReference” may be used with each element within WSDL and XML schema. However, SAWSDL defines its meaning only for interface, operation, fault, element, complexType, simpleType and attribute [20]. “It should be noted that the guidance given regarding the uses of “modelReference” for each of these elements has much more the flavor of suggestions than definitions” [5]. For example, the material usage with interfaces mentions that “modelReference” can be used to categorize them according to some model, specify behavioral aspects or other semantic definitions, and similarly for operations. Consequently, when an operation is annotated using several semantic concepts in an ontology, one is not able to differentiate these concepts: which one annotates category, which one annotate the behavior, which one annotate the QoS? And so on. Similar observation can be made for any semantic annotation of any WSDL element. Therefore, we are convinced that it is necessary to be able to differentiate the semantic description of all services elements. Differentiation of semantic annotation of WSDL elements can be used to enhance Web services discovery. Instead of using “modelReference” to associate one or more semantic properties to a WSDL element, we should use means to differentiate each semantic property that can be associated to a WSDL element. Indeed, one can consider discovering Web services using one specific semantic property such as Web services effects or can consider composing/invoking Web services using one specific semantic property such as Web service behavior.

In [12], there are many issues needing clear decision and more precision, having to do with the mapping of the inputs,
outputs, operations, interfaces, and faults. It is related with difficulties in mapping between Message Exchange Pattern in SAWSDL and inputs/outputs in atomic processes on one hand and the fact that OWL-S does not have constructs that provide direct correspondence to “interface” elements in SAWSDL on the other hand.

All these limitations of OWL-S, WSMO, metamodel of WSDL, and other approaches are the origin of our motivation for the definition of Yet Another Semantic Annotation for WSDL (YASA4WSDL) that annotates WSDL description not only by business concepts but also service concepts of OWL-S, WSMO, metamodel of WSDL and any other technical service ontology.

The aim of this paper is to present YASA4WSDL and define technical service ontology for YASA4WSDL that integrate concepts of WSDL meta-model, the OWL-S upper ontology and WSMO.

IV. CONTRIBUTION

In this section we present YASA4WSDL, our semantic annotation for WSDL according to the overview given in Section II.

Compared to WSMO and OWL-S, YASA4WSDL offers multiple advantages. First of all, users can describe, in an upwardly compatible way, both the semantic and the syntactic functional and non functional properties in WSDL, a language that the developers’ community is familiar with. Second, by externalizing the semantic of business domain models and the technical concepts (service and its QoS, context…), one can allow Web service developers to annotate their Web services with their ontology language of choice (such as UML or OWL) unlike in OWL-S or WSMO. In addition, OWL-S and WSMO define their own Service ontologies unlike in YASA4WSDL, which can integrate any technical ontology (Context, QoS or service ontology).

In addition, regarding to the work presented in [12], YASA4WSDL allows the developer community to specify the correspondence between service concepts (interfaces, operation, fault…) and business concepts. This explicit correspondence can be used to bridge YASA4WSDL and OWL-S, WSMO, and many other description languages for semantic Web services [1].

According to DIANE approach [15, 16, 17], our approach proposes almost the same contributions but it focuses on up to date standards (SAWSDL) and common WSDL-oriented tools.

As an extension of WSDL, YASA4WSDL can use all types of ontologies. It facilitates its adoption by developers accustomed to WSDL.

A. YASA4WSDL presentation

The main objective of YASA4WSDL is to extend SAWSDL for enhancing expressiveness of service description. In SAWSDL for a given WSDL element one can use many references to concepts in a domain ontology but there is no specification of the semantic information nature: is it a precondition, an effect or a result? etc. In our description, we propose a new attribute called serviceConcept to give references to the technical concepts corresponding, in the same order, to the domain concepts listed in the original SAWSDL “modelReference” attribute. Indeed our approach for semantic Web services description is based on the use of two types of ontologies. The first one, called Technical Ontology, contains several concepts defining semantics of services' concepts and concepts describing their non functional properties (QoS, context…). The second type of ontologies, called Domain Ontology, contains the semantics of the service domain concepts (e.g. tourism, health, trade…).

Figure 1. Annotation system in YASA4WSDL.

A service described in YASA4WSDL can define for each WSDL element two attributes providing semantic description. The first attribute serviceConcept, contains a set of URI referencing the corresponding concepts in one or several Technical Ontologies. The second attribute contains a set of URI corresponding to the first list and which define the semantics in one or several Domain Ontologies. Let’s consider the example presented in Figure 1. There are two ontologies: the first is Technical (Service Ontology) that describes the semantics of some service concepts: precondition and result. The second ontology is a Domain ontology (called TransportOntology) that describes some concepts in the travel domain: validFlightInfo and reservationInfo. The example presented the semantic annotation of an operation named reserveFlight. The modelReference attribute references two concepts: validFlightInfo and reservationInfo. The importance of the extended attribute serviceConcept is to distinguish the role played by each concept referenced by modelReference attribute. The serviceConcept references two service concepts: precondition and effect that correspond to the two domain concepts validFlightInfo and reservationInfo. The order of the references is important, it associates the first technical concept with the first domain concept, the second technical concept with the second domain concept and so on.

Another advantage provided by this new approach in YASA4WSDL is that we can extend the Service Ontology by new concepts (new description element, more precise concepts…) that has no impact on the annotation system. There will be no limited service ontology. This makes the system of annotation independent of the used semantic business domain representation language and ontology and the used semantic service representation language and ontology. It gives flexibility to the developers’ community to
select their favorite semantic representation language and their technical ontology, to reuse semantic domain models and annotate descriptions using multiple ontologies.

<table>
<thead>
<tr>
<th>OWL-S</th>
<th>WSMO</th>
<th>WSDL</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effect</td>
<td>Capability</td>
<td>Service</td>
<td>Community</td>
</tr>
<tr>
<td>Precondition</td>
<td>Goal</td>
<td>Interface</td>
<td>Version</td>
</tr>
<tr>
<td>Process</td>
<td>WebService</td>
<td>Operation</td>
<td>Status</td>
</tr>
<tr>
<td>Result</td>
<td>Precondition</td>
<td>Input</td>
<td>Business</td>
</tr>
<tr>
<td>Input</td>
<td>Postcondition</td>
<td>Output</td>
<td>Actor</td>
</tr>
<tr>
<td>Service</td>
<td>Assumption</td>
<td>Description</td>
<td>Provider</td>
</tr>
<tr>
<td>Participant</td>
<td>Effect</td>
<td>Binding</td>
<td>Capacity</td>
</tr>
<tr>
<td>Category</td>
<td>Orchestration</td>
<td>Fault</td>
<td>Constraint</td>
</tr>
</tbody>
</table>

TABLE I. MAIN CONCEPTS TO MAP

B. Integrated ontology for Web services

In this subsection, we present how we define an integrated ontology containing service concepts of OWL-S, WSMO, metamodel of WSDL and any other technical service ontology. Yet Another Semantic Annotation for WSDL (YASA4WSDL) annotates WSDL description using technical concepts from the integrated ontology and business concepts from domain ontologies. To build the technical ontology, we start mapping concepts from ontologies of OWL-S, WSMO, WSDL meta-model and some other technical service ontologies. This corresponds with identifying the connections between the entities of ontologies to match. For the mapping, we use matching techniques to calculate the best matching between a pair of entities. Indeed, the matching techniques can maximize the detection of similar couples of concepts and reduce the number of those who are dissimilar. Table I shows the main concepts to map.

TABLE II. EXTRACT FROM RESULTS OF TERMINOLOGICAL RULES

We attach to the terminological matcher a trust degree: Conf = 1 and define rules for calculating the similarity value SV, returned by the matcher, for a couple of comparable concepts C1 and C2 (0 for the symbols ! and #).

We use a combination of parallel matchers; the matching techniques can maximize the detection of similar concepts and reduce the number of those who are dissimilar. Table I shows the main concepts to map.

TABLE III. EXTRACT FROM RESULTS OF TERMINOLOGICAL RULES

Calculation of similarity and mapping: We use here two types of matcher: terminological and linguistic.

a) Terminological matching: After normalization of terms (deleting unnecessary character, normalization of compound words, etc.), we adopt here a definition of matching based on Levenshtein distance. It measures the similarity between two strings and finds the minimum number of insertions and deletions of characters required to convert a string into another. Table II shows the result of matching between some concepts of the Table I. It shows only the values of distance which are less than the sum of lengths of two terms. If we replace all the letters in a word with all the letters of another, then these two words are not near in terminological point of view. We use the symbol “!” for the great values and the symbol “#” for the concepts of the same ontology.

SV=1; if Distance=0
SV=1/1+Distance-Min(nom(C1),nom(C2));
if Distance>Min(nom(C1),nom(C2))
SV=0.7; otherwise

We attach to the terminological matcher a trust degree: Conf = 1 and define rules for calculating the similarity value SV, returned by the matcher, for a couple of comparable concepts C1 and C2 (0 for the symbols ! and #).

TABLE III. EXTRACT FROM RESULTS OF TERMINOLOGICAL RULES

b) Auxiliary information-based linguistic matching: We adopt here the definition of WordNet dictionary-based matcher: we rely on the semantic relationships between words in natural language to calculate the similarity value SV: synonyms are equivalent entities (SV = 1), hyponyms are recovered entities (SV = 0.5), otherwise there is no relation (SV = 0). We attach to this matcher a trust degree: Conf = 2. Table IV presents the similarity values.

c) Combining matchers and generation of hypothesis mapping: We use a combination of parallel matchers; the result of the various individual matchers will be combined (see Table V) to identify only one mapping candidate (also called mapping hypothesis) between each pair of concepts (as described in [21]). Each of these mapping hypothesis (Hpi) contains five-tuples: Hp <R, c, c', Conf Hp, SVHp> with:

- R: relationship identified by the matchers between the couple of concepts c and c'.

TABLE IV. EXTRACT FROM RESULTS OF LINGUISTIC MATCHING
The trust degree of the mapping hypothesis $H_p$ is:
\[
Conf_{H_p} = \sum Conf_i
\]

The similarity value produced by a combination of matchers:
\[
SV_{H_p} = \frac{\sum Conf_i \times SV_i}{\sum Conf_i}
\]

### TABLE V. 
**Extract from results of combined matching**

<table>
<thead>
<tr>
<th>Concepts</th>
<th>Precondition</th>
<th>Result</th>
<th>Capacity</th>
<th>Input</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Interface</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Precondition</td>
<td>1</td>
<td>0</td>
<td>0.08</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Postcondition</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.56</td>
</tr>
<tr>
<td>WebService</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Capability</td>
<td>0</td>
<td>0.03</td>
<td>0.9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Output</td>
<td>0</td>
<td>0.1</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The mapping is based on Table V as the mapping hypothesis depend on it: $H_p<=$, Input, Result, 3, 0.1>

Algorithm:

If $\exists$ Map<=$,c5,d4,Conf_1,SV2>$

Then If $\exists$ $H_p<=$,c3,d2,Conf_2,SV2>$

- $SV2 = SV2+(SV1/\min(|V(c5)|_R,|V(d4)|_R))$
- $Conf_2 = Conf_2+Conf_{sem-R}$

Else

- Generate $H_p<=$,c3,d2,Conf_2,SV2>$

$SV2 = SV2+(SV1/\min(|V(c5)|_R,|V(d4)|_R))$
- $Conf_2 = Conf_{sem-R}$

Figure 2. «Top-Down» Semantic matcher of a non-symmetric relation.

2) Filtering: There are three types of filters: structural, semantic and threshold-based filter:

- With structural filters, we can use mappings generated earlier (to make a crease with a mapping), or make a cross-crease mapping hypothesis or eventually eliminate divergences with the existing mapping,
- With semantic filters, we also exploit semantic relationships in ontologies based on their type.
- With threshold-based filters, we define a threshold (Th) on the similarity values below which mapping hypothesis are not considered (see Rule (4)), the value of Th could be amended by the system administrator through a graphical interface.

For each: $H_p<=$, c, d, Conf, SV$>,$

\[
if (SV<Th) then (eliminate Hp)
\]

Finally we apply all the filters needed, on the mapping hypothesis. For example, with a threshold-based filter, we sort in descending order, according to the SV similarity value in mapping hypothesis and we eliminate less relevant mapping hypothesis. At this stage, it only remains to validate the mapping and export resulting ontology. We were unable to insert the tree technical services ontology given the size of the figure. The filtering and validation of mapping hypothesis allow to generate mappings as:

\[
Map<=$,Input WSDDL,Input OWL-S>$
\]

\[
Map<=$,Interface WSDDL,Interface OWL-S>$
\]

\[
Map<=$,Precondition OWL-S,Precondition WSMO>$
\]

\[
Map<=$,Output OWL-S,Output WSDDL>$
\]

\[
Map<=$,Capability,Capacity>$
\]

\[
…
\]
Here is a list of main concepts of the technical ontology after validation of mappings:

<table>
<thead>
<tr>
<th>Concept</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effect</td>
<td>ServiceModel</td>
</tr>
<tr>
<td>Precondition</td>
<td>Process</td>
</tr>
<tr>
<td>AtomicProcess</td>
<td>SimpleProcess</td>
</tr>
<tr>
<td>CompositeProcess</td>
<td>Parameter</td>
</tr>
<tr>
<td>Profile</td>
<td>Local</td>
</tr>
<tr>
<td>Participant</td>
<td>Input</td>
</tr>
<tr>
<td>Output</td>
<td>Result</td>
</tr>
<tr>
<td>Service</td>
<td>Choreography</td>
</tr>
<tr>
<td>Orchestration</td>
<td>Capability</td>
</tr>
<tr>
<td>Goal</td>
<td>Interface</td>
</tr>
<tr>
<td>Effect</td>
<td>Assumption</td>
</tr>
<tr>
<td>Postcondition</td>
<td>Description</td>
</tr>
<tr>
<td>InterfaceFault</td>
<td>InterfaceOperation</td>
</tr>
<tr>
<td>Operation</td>
<td>BindingOperation</td>
</tr>
<tr>
<td>Endpoint</td>
<td>BindingFault</td>
</tr>
<tr>
<td>Binding</td>
<td>Documentation</td>
</tr>
<tr>
<td>Comment</td>
<td>Category</td>
</tr>
<tr>
<td>Community</td>
<td>Version</td>
</tr>
<tr>
<td>Status</td>
<td>Business</td>
</tr>
<tr>
<td>Actor</td>
<td>Provider</td>
</tr>
<tr>
<td>Constraint</td>
<td>Port</td>
</tr>
</tbody>
</table>

V. CONCLUSION AND FUTURE WORK

In this paper, we presented a brief review of services ontologies, and techniques and systems of mapping, useful for the integration of ontologies. We have built a technical service ontology by integrating ontologies of OWL-S, WSMO and WSDL meta-model and other concepts, through the identification of correspondences between the concepts of these ontologies.

In YASA4WSDL, the service description annotation can be made “manually” or through a graphical interface to assist the user by importing ontologies and presenting their concepts. Any description item can be annotated by any concept, a set of annotation constraints remains to define and to implement on the annotation interface. Our future work is to formalize these constraints and implement a tool for the annotation descriptions YASA4WSDL.

REFERENCES