CEUR Workshop Proceedings
http://ceur-ws.org
ISSN 1613-0073

Vol-596
urn:nbn:de:0074-596-3

ORES-2010
Ontology Repositories and Editors for the Semantic Web

Proceedings of the 1st Workshop on Ontology Repositories and Editors for the Semantic Web

Hersonissos, Crete, Greece, May 31st, 2010.

Edited by

Mathieu d'Aquin, **The Open University, UK**
Alexander García Castro, **Universität Bremen, Germany**
Christoph Lange, **Jacobs University Bremen, Germany**
Kim Viljanen, **Aalto University, Helsinki, Finland**

10-Jun-2010: submitted by Christoph Lange
11-Jun-2010: published on CEUR-WS.org

On the Use of Transformation and Linked Data Principles in a Generic Repository for Semantic Web Services

Barry Norton[1], Mick Kerrigan[2], and Adrian Marte[2]

[1] AIFB, Karlsruhe Institute of Technology, Germany
barry.norton@kit.edu
[2] Semantic Technology Institute, University of Innsbruck, Austria
first.last@sti-innsbruck.at

Abstract. As yet, despite many years of research into Semantic Web Services, there is no standard Semantics-based service model. SA-WSDL, a W3C recommendation, provides links only to an unspecified semantic model from WSDL, itself merely an XML syntax. Ontology-based approaches such as OWL-S and WSMO have never advanced beyond the stage of submission to standards bodies. In this environment one approach to moving forward is exemplified by WSMO-Lite, which can be viewed as an 'intersection' of the features of OWL-S and WSMO as applied a semantic representation of the relevant structure of WSDL, the so-called 'minimal service model'. A relatively newer standards submission, named the Semantic SOA Reference Ontology and the subject of this paper, takes instead a 'union' approach to the features considered so far. We shall show that among the advantages of such an approach is its ability to mediate between the existing models. The major contribution of this paper, alongside detailing the Reference Ontology, is to show how this mediation can be effected simply using SPARQL and exemplify the practicality of this solution as the basis for a RESTful service repository based on Linked Data principles.

1 Introduction

Semantic Web Services (SWS) provide a means for creating richer descriptions for Web Services, where explicit ontology-based semantics increase automation in service creation and consumption tasks via reasoning. Semantic Web Services form a layer on top of existing Web Service technologies and not a replacement for them.

In order to use the semantic descriptions present in a so-called Semantic SOA (SSOA), to automate the tasks associated with Service-Oriented Architectures (SOA), a set of reasoning-based platform services are required within the SSOA. These services are collectively termed a Semantic Execution Environment (SEE) and form the core of a SSOA-based implementation. There are a number of different implementations of SEEs currently under development in the research community, which have some common features. Examples of such Semantic Execution Environments are WSMX [1], IRS-III [2], and METEOR-S [3].

The OASIS Semantic Execution Environment Technical Committee (SEE-TC)[1], which is co-chaired by the authors, was established in 2005, with the aim to:

> "provide guidelines, justifications and implementation directions for an execution environment for Semantic Web services. The resulting infrastructure will incorporate the application of semantics to service-oriented systems and will provide intelligent mechanisms for consuming Semantic Web services."

The SEE-TC is in the process of standardizing the types of platform services that exist within a Semantic Execution Environment for Semantic Web Services, their black box behaviour, and their interfaces. In order to consistently define these platform service the SEE-TC have first defined a Semantic SOA Reference Ontology (SSOA-RO), which provides a description of the elements that need to be modeled in order to effectively provide semantic description for services.

Currently under public review, on the route towards OASIS standardisation, one of the main requests was that the Reference Ontology be made available in the form of RDFS. Work towards this has opened up the possibility, motivated particularly the SEALS project[2], the ability to transform descriptions in existing service models into the SSOA-RO, and to use this as the basis to produce definitions in these different models. It has been found that this is largely feasible using the 'CONSTRUCT' syntax for the semantic query language SPARQL.

The authors specified, for the SEALS project, which is concerned with the benchmarking and evaluation of semantic technologies, RESTful APIs for repositories containing tools, test data (including SWS descriptions), and evaluation results[3]. The transformation of service descriptions into different service models was included as an instance of 'synthetic test data generation', where descriptions stored in the Semantic SOA Reference Ontology are transformed via SPARQL.

In the meantime, the SOA4All project [4] has pointed out that such a RESTful API for the management of such descriptions, where the descriptions are exposed in RDF as they are in the SEALS repository, can be the basis of exposing service descriptions as Linked Data. In particular, each service is managed via a unique URI, deferencable via HTTP, and linking to related datasets. The transformations presented in this paper are applied within a discovery-enabled repository, Discovery Cloud[5], interface-compatible with the SOA4All repository, iServe, for the storage of services in the minimal service model common to microWSMO and WSMO-Lite, but which also allows retrieval and inter-translation of WSMO and OWL-S services and goal/template-based discovery.

The paper is arranged as follows: the SSOA-RO is reviewed in Section 2; transformations from existing models into this ontology are considered in Section 3; transformations back out to the existing models are exemplified in Section 4; the extension of a Linked Data-compliant API to these transformations is considered in Section 5; finally, conclusions are drawn and further work discussed in Section 6.

[1] `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=semantic-ex`

[2] Semantic Evaluation at Large Scale: `http://www.seals-project.eu/`

[3] `http://about.seals-project.eu/downloads/category/1-`

[4] Service Oriented Architectures for All: `http://www.soa4all.eu/`

[5] `http://km.aifb.kit.edu/services/DisCloud`

2 OASIS Semantic SOA Reference Ontology

The Semantic SOA Reference Ontology (RO), currently under public review on the route to OASIS standardisation, is an ontology expressed in RDFS that aims to combine the features of OWL-S [4], WSMO [5] and WSMO-Lite [6], and be consistent with the terminology and concepts of the OASIS SOA Reference Model (SOA-RM) [7]. To this end the starting point for the Reference Ontology is the top-level concepts of the SOA-RM shown in Figure 1. 'Execution Context' and 'Contract & Policy' are shown dotted as the work on their semantic description is considered too early for standardisation.

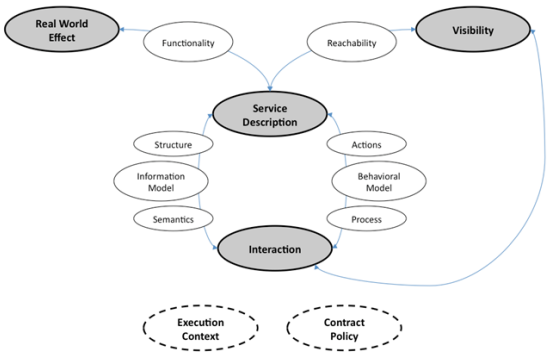

Fig. 1. Top-Level Concepts of OASIS SOA Reference Model

Figure 2 shows how these concepts are refactored and supplemented in the Semantic SOA Reference Ontology.

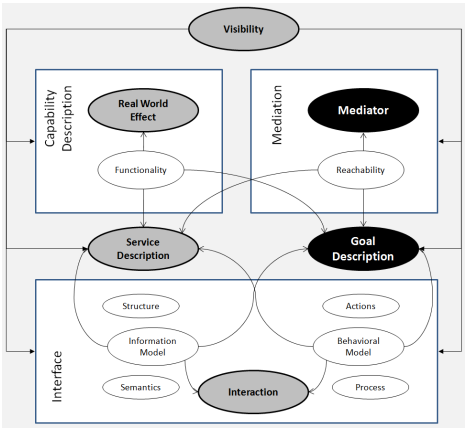

Fig. 2. Top-Level Concepts of OASIS Semantic SOA Reference Ontology

In particular, the concept of 'Visibilty' becomes a global underlying concern via the expression of the whole service model, not just the information model, in ontology-based semantics. The associated concept of 'Reachability' is subsumed into the WSMO-inspired notion of 'Mediation' (shown in black as a new top-level concept), by which top-level concepts can be connected together with a specification of the means to overcome any heterogeneities. Also novel is the WSMO-inspired concept of 'Goal', by which client requirements of a service interaction are documented. The concepts of 'CapabilityDescription' and 'Interface' are shown as groupings.

We consider now the RDFS definitions for these parts of the SSOA-RO model. The relevant 'top-level' RDFS descriptions are reproduced in Figure 3.

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix ro: <http://docs.oasis-open.org/semanticsoa/referenceontology/v1.1#> .


ro:TopLevelElement rdf:type rdfs:Class .

ro:ServiceDescription rdfs:subClassOf ro:TopLevelElement .

ro:GoalDescription rdfs:subClassOf ro:TopLevelElement .

ro:Ontology rdfs:subClassOf ro:TopLevelElement ;
            rdfs:subClassOf owl:Ontology .

ro:importsOntology rdf:type rdf:Property ;
                   rdfs:domain ro:TopLevelElement ;
                   rdfs:range ro:Ontology .

ro:usesMediator rdf:type rdf:Property ;
                rdfs:domain ro:TopLevelElement ;
                rdfs:range ro:Mediator .

ro:Mediator rdfs:subClassOf ro:TopLevelElement .

ro:hasSource rdf:type rdf:Property ;
             rdfs:domain ro:Mediator ;
             rdfs:range ro:TopLevelElement .

ro:hasTarget rdf:type rdf:Property ;
             rdfs:domain ro:Mediator ;
             rdfs:range ro:TopLevelElement .
```

Fig. 3. Semantic SOA Reference Ontology Top-Level Elements in RDF(S)

Definitions, from the Reference Ontology, relevant for the 'heavyweight' definition of service capabilities, and the functional requirements of clients via goals, are shown in Figure 4. It should be noted that the Reference Ontology, and SEALS in turn, makes no particular stipulation about which language should be used to encode rules, using instead a subclass of RDF literal. When RIF[6] [8] becomes a standard it might be possible to rely on this as a common interchange for rules.

[6] http://www.w3.org/2005/rules/wiki/RIF_Working_Group

```
ro:offersCapability rdf:type rdf:Property ;
                    rdfs:domain ro:ServiceDescription ;
                    rdfs:range ro:Capability .

ro:requiresCapability rdf:type rdf:Property ;
                      rdfs:domain ro:GoalDescription ;
                      rdfs:range ro:Capability .

ro:Capability rdfs:subClassOf ro:FunctionalDescription .

ro:hasPrecondition rdf:type rdf:Property ;
                   rdfs:domain ro:FunctionalDescription ;
                   rdfs:range ro:RuleLiteral .

ro:hasAssumption rdf:type rdf:Property ;
                 rdfs:domain ro:FunctionalDescription ;
                 rdfs:range ro:RuleLiteral .

ro:hasSharedVariable rdf:type rdf:Property ;
                 rdfs:domain ro:FunctionalDescription ;
                 rdfs:range rdfs:RuleVariableLiteral .

ro:hasPostcondition rdf:type rdf:Property ;
                    rdfs:domain ro:FunctionalDescription ;
                    rdfs:range ro:RuleLiteral .

ro:hasEffect rdf:type rdf:Property ;
             rdfs:domain ro:FunctionalDescription ;
             rdfs:range ro:RuleLiteral .

ro:RuleLiteral rdfs:subClassOf rdfs:Literal .

ro:RuleVariableLiteral rdfs:subClassOf rdfs:Literal .
```

Fig. 4. Service and Goal Capability Definitions from Reference Ontology

Ranking and lightweight service discovery can be based on the properties shown in Figure 5. As well as these functional and non-functional descriptions of services, another important part of Semantic Web Service models concerns the interfaces of services. The Reference Ontology's high-level definitions for these are shown in Figure 6.

```
ro:hasNonFunctionalParameter rdf:type rdf:Property ;
                             rdfs:domain ro:TopLevelElement ;
                             rdfs:range ro:NonFunctionalParameter .

ro:requiresClassification rdf:type rdf:Property ;
                          rdfs:domain ro:GoalDescription ;
                          rdfs:range rdfs:Class .

ro:hasClassification rdf:type rdf:Property ;
                     rdfs:domain ro:ServiceDescription ;
                     rdfs:range rdfs:Class .

ro:ClassificationRoot rdfs:subClassOf rdfs:Class .
```

Fig. 5. Further (Non)Functional Definitions from Reference Ontology

```
ro:supportsInterface rdf:type rdf:Property ;
                     rdfs:domain ro:ServiceDescription ;
                     rdfs:range ro:Interface .

ro:requiresInterface rdf:type rdf:Property ;
                     rdfs:domain ro:GoalDescription ;
                     rdfs:range ro:Interface .

ro:Choreography rdfs:subClassOf ro:Interface .

ro:Orchestration rdfs:subClassOf ro:Interface .

ro:hasGlobalActionModel rdf:type rdf:Property ;
                        rdfs:domain ro:Interface ;
                        rdfs:range ro:ActionModel .
```

Fig. 6. Interface Definitions from Reference Ontology

The Reference Ontology also defines a property for Process Models for interfaces, but no standard has yet been defined for this (this is a topic for future consideration in the Technical Committee):

```
ro:hasProcessModel rdf:type rdf:Property ;
                   rdfs:domain ro:Interface ;
                   rdfs:range ro:ProcessModel .
```

Since the transformations detailed in this paper do not currently include orchestration, since this is a highly heterogeneous part of existing semantic service models, and furthermore consider only atomic processes (from a client perspective), it is sufficient to reproduce only the definitions for (choreography) Action Models, shown in Figure 7.

```
ro:hasInputAction rdf:type rdf:Property ;
                  rdfs:domain ro:ActionModel ;
                  rdfs:range ro:Action .

ro:hasOutputAction rdf:type rdf:Property ;
                   rdfs:domain ro:ActionModel ;
                   rdfs:range ro:Action .

ro:hasSharedAction rdf:type rdf:Property ;
                   rdfs:domain ro:ActionModel ;
                   rdfs:range ro:Action .

ro:communicatesConcept rdf:type rdf:Property ;
                               rdfs:domain ro:Action ;
                               rdfs:range rdfs:Class .

ro:communicatesMessage rdf:type rdf:Property ;
                               rdfs:domain ro:Action ;
                               rdfs:range rdfs:Resource .
```

Fig. 7. Action Model from Reference Ontology

3 Transformations to the Reference Ontology

In order to produce instances of the Reference Ontology from existing service collections, some form of transformation is necessary. For the most part it has been established that SPARQL is sufficient to achieve the structural transformations. A set of CONSTRUCT queries are provided for transformation into — and, as described in Section 4, out of — the Reference Ontology from the other models considered. These all rely on the following prefixes:

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl:<http://www.w3.org/2002/07/owl#>
PREFIX oservice:<http://www.daml.org/services/owl-s/1.1/Service.owl#>
PREFIX oprofile:<http://www.daml.org/services/owl-s/1.1/Profile.owl#>
PREFIX oprocess:<http://www.daml.org/services/owl-s/1.1/Process.owl#>
PREFIX ogrounding:<http://www.daml.org/services/owl-s/1.1/Grounding.owl#>
PREFIX ro:<http://docs.oasis-open.org/semanticsoa/referenceontology/v1.1#>
PREFIX part:<http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl#>
PREFIX wsml:<http://www.wsmo.org/wsml/wsml-syntax#>
PREFIX sawsdl:<http://www.w3.org/ns/sawsdl#>
PREFIX wsl:<http://www.wsmo.org/ns/wsmo-lite#>
```

These queries are exemplified with the 'OWLS2RO' transform, whose body is given in Figure 8.

```
WHERE
 {{?service rdf:type oservice:Service} .
  {?service oservice:presents ?profile} .
  {?service oservice:describedBy ?process} .
  {?process rdf:type process:AtomicProcess} .
  {?service oservice:supports ?grounding} .
  {?grounding ogrounding:hasAtomicProcessGrounding ?processGrounding} .
  {{{?profile oprofile:hasInput ?input} .
     {?input oprocess:parameterType ?inputType} .
     OPTIONAL
      {{?processGrounding ogrounding:wsdlInput ?inputMessage} .
       {?inputMessage ogrounding:owlsParameter ?input} .
       {?inputMessage ogrounding:wsdlMessagePart ?inputPart} .
       OPTIONAL {?inputMessage ogrounding:xsltTransformationString ?inputTransform}}}
    UNION
    {{?profile oprofile:hasOutput ?output} .
     {?output oprocess:parameterType ?outputType} .
     OPTIONAL
      {{?processGrounding ogrounding:wsdlOutput ?outputMessage} .
       {?outputMessage ogrounding:owlsParameter ?output} .
       {?outputMessage ogrounding:wsdlMessagePart ?outputPart} .
       OPTIONAL {?outputMessage ogrounding:xsltTransformationString ?outputTransform}}}} .
  OPTIONAL {?profile oprofile:serviceClassification ?classification}}
```

Fig. 8. OWLS2RO Transformation Body in SPARQL

It should be noted that, since OWL-S sticks deliberately to the DL fragment of OWL, it is unable to give the range of input and output types as classes (as this would require a metaclass), and represents these simply as URIs.

This can be seen in the result of executing the body with the head 'SELECT *' over a knowledge base containing the 'BookNonMedicalFlight' service from the current 1.1 version of the OWL-S Test Collection[7], where rows with a binding of '?service' to:

```
<http://127.0.0.1/services/1.1/BookNonMedicalFlight_service.owls#
                                        BookNonMedicalFlightService>
```

include one with a binding of '?input' to:

```
<http://127.0.0.1/services/1.1/BookNonMedicalFlight_service.owls#
                                       BookNonMedicalFlight_Account>
```

and '?inputType' to:

```
"http://127.0.0.1/ontology/NonMedicalFlightCompanyOntology.owl#
                  Account"^^<http://www.w3.org/2001/XMLSchema\#anyURI>
```

In post-processing the construct query, therefore, we make direct reference (since there is no problem doing so in RDFS, or in other target languages such as WSML) to the class, but preserve the URI-based reference too, in order to recover the OWL-S description. Figure 9 shows the simple query head where just the Reference Ontology triples are created.

```
CONSTRUCT
 {?service rdf:type ro:ServiceDescription .
  ?process rdf:type ro:Choreography .
  ?process ro:hasGlobalActionModel ?processGrounding .
  ?processGrounding ro:hasInputAction ?input .
  ?input ro:communicatesConcept ?inputType .
  ?input ro:communicatesMessage ?inputMessage .
  ?inputMessage part:hasPart_directly ?inputPart .
  ?inputMessage sawsdl:loweringSchema ?inputTransform .
  ?processGrounding ro:hasOutputAction ?output .
  ?output ro:communicatesConcept ?outputType .
  ?output ro:communicatesMessage ?outputMessage .
  ?outputMessage part:hasPart_directly ?outputPart .
  ?outputMessage sawsdl:liftingSchema ?outputTransform .
  ?service ro:hasClassification ?classification .
  ?service rdfs:subClassOf ro:ClassificationRoot}
```

Fig. 9. OWLS2RO Transformation Head in SPARQL

Note that the Reference Ontology follows WSMO in grounding a class to an entire WSDL message, whereas OWL-S uses the specific WSDL 1.1 mechanism of 'parts' to map individual inputs. The transform therefore creates such a class using the W3C simple partonomy vocabulary to form the links to individual concepts. As well as replacing URIs with class references in the 'communicatesConcept' properties, the post-processing will concatenate a set of membership tests across the input types to form a WSML precondition, and form a WSML postcondition by doing the same for output types (such conditions and effects are implicit in OWL-S). Where other pre- and post-conditions exist in WSML, RIF will be used, when this becomes a standard, to express these in the most generic way possible. This is unlikely to be possible with SWRL conditions in OWL-S descriptions, but these are very scarce in practice.

[7] OWLS-TC3/htdocs/services/1.1/BookNonMedicalFlight_service.owls

Finally note that the same problem of references to classes complicates the use of functional classification in OWL-S, which is further compounded by the way in which service descriptions express the dependency on ontologies. The transform makes the associated concept a subclass of the functional classification root in the Reference Ontology, which should be propagated up the subsumption hierarchy, but this is impossible for OWL-S (though perfectly possible in WSMO and WSMO-Lite descriptions).

4 Transformation from the Reference Ontology

In order to exemplify the transformations from the SSOA-RO into existing service models, the example from the OWL-S Test Collection considered in the previous section (i.e., the OWL-S v1.1 BookNonMedicalFlight service) will be transformed to WSMO-Lite. We elide the common features and show a fragment of the RO2WSMOLite transform, primarily to show the result of the postprocessing described above, in Figure 10.

```
CONSTRUCT
  {?service rdf:type wsl:Service .
   ?precondition rdf:type wsl:Condition .
   ?precondition rdf:value ?preconditionValue .
   ?service sawsdl:modelReference ?precondition .
   ?precondition rdf:value ?preconditionValue .
   ?postcondition rdf:type wsl:Condition .
   ?postcondition rdf:value ?postconditionValue .
   ?service sawsdl:modelReference ?postcondition .
   ?postcondition rdf:value ?postconditionValue .
}
WHERE
 {?service rdf:type ro:ServiceDescription .
  ?service ro:hasCapability ?capability .
  ?capability ro:hasPrecondition ?precondition .
  ?precondition rdf:value ?preconditionValue .
  ?capability ro:hasPostcondition ?postcondition .
  ?postcondition rdf:value ?postconditionValue}
```

Fig. 10. OWLS2RO Transformation Head in SPARQL

The result of applying this query to the post-processed version of the example considered in the previous section is shown, as a screenshot of the execution of the query in the Sesame Workbench, in Figure 11.

In many cases the SAWSDL for service descriptions has already been produced, though may be updated to include references such as those shown to new WSML expressions. Where they do not exist they will be created by extension of the existing WSDL. In each case these will be stored, alongside the RDF representation in the Reference Ontology in the Test Data Repository.

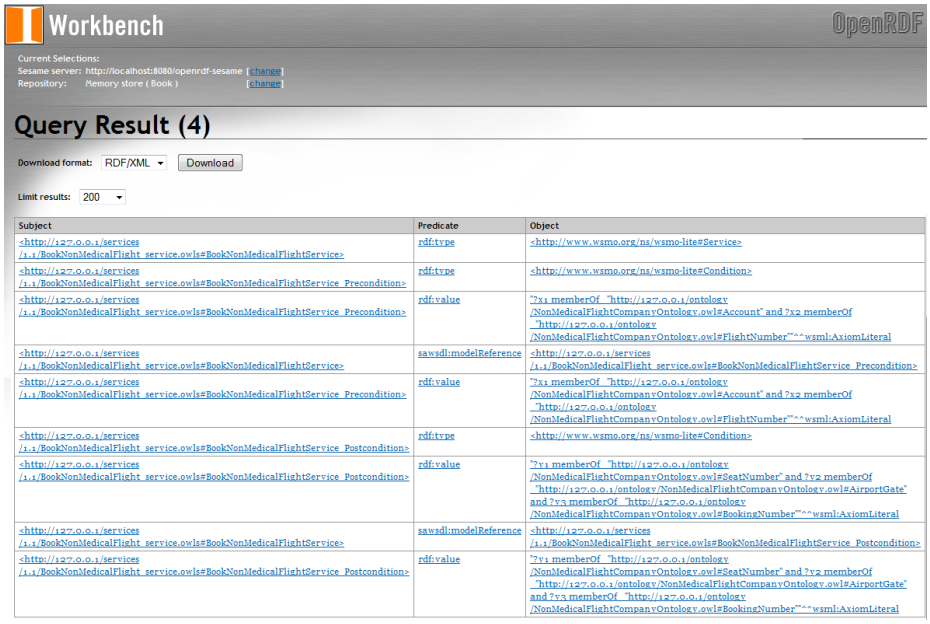

Fig. 11. Transformation of OWL-S TC 1.1 Flight Booking Service to WSMO-Lite

5 Service Descriptions as Linked Data

The four so-called 'Linked Data Principles' are originally stated[8] as follows:

1. Use URIs as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
4. Include links to other URIs. so that they can discover more things.

In a resource-oriented, i.e, truly RESTful, service interface it is natural that service descriptions are identified by resolvable URIs (principles 1 and 2), and this was the approach taken for repositories in the SEALS project and later SOA4All. Furthermore, as has been shown, all service models can be represented in RDF (principle 3) and this is the approach taken in the repository interface in iServe and here. Finally services link to ontologies used in their description, in particular domain ontology import is exposed as a subproperty to RDF's 'seeAlso' (principle 4 — further means to meet this principle are considered in Section 6.

[8] `http://www.w3.org/DesignIssues/LinkedData.html`

In order to respect Linked Data and REST principles, and remain compatible with the iServe API used in the SOA4All project, the extensions described in this paper are encoded as an extension to content negotiation. In HTTP, URIs identify *resources* but these might have different *representations* that can be retrieved. A common example is in the format used for a picture — a client might ask (preferentially) for a JPG over a GIF encoding. Similarly in Linked Data we might allow retrieval of RDF content in RDF/XML, Turtle (n3) etc. The repository interface supported here allows exactly specification of the RDF format required in the usual 'accept' field to the request header.

The service model in which the result should be returned, however, is orthogonal to the RDF format. One might request a WSMO description in N3, or an OWL-S description in RDF/XML, for instance. For this reason we introduce a second header field, which acts like a simplified version of accept headers and negotation, with the key 'service_model' and values that are URIs for each of the service models.

In both cases, i.e. the service model and the RDF encoding, furthermore, there should be some default in case the client makes no specific request. In order to remain compatible with iServe the default service model is the 'minimal service model' of WSMO-Lite/microWSMO. Since it is the Web representation, the default RDF encoding is RDF/XML.

6 Conclusions and Further Work

This paper has detailed the Semantic SOA Reference Ontology, its definition in RDFS, and its utility in transforming between existing service models to increase interoperability in the usage of semantic repositories for service descriptions. Concretely it has shown how SPARQL is, for the most part, sufficient to achieve these transformations. Finally it has been shown how an API, compatible with Linked Data principles, can be formed to expose these transformations over a service repository in a RESTful fashion.

On-going work on the repository considers the treatment of templates, used as the basis for discovery, as permanent resources, i.e. uploaded to the repository just as service descriptions, where discovery is carried out on an on-going basis as new service descriptions are found by crawling and/or uploaded. A 'GET' retrieval on the service template can therefore also return a dynamically-ranked set of services that can be used to achieve the template behaviour, providing further justification for the claims with respect to the fourth Linked Data principle.

Future work on the Reference Ontology will consider process models sufficient for orchestration and it is hoped that these will be derivable from the OWL-S process model, and that restricted processes will be capable of transformation into the OWL-S process model, WSML Abstract State Machine-based orchestrations and semantic BPEL extensions, as already considered — with transformations based on semantic rule languages — in [9].

Acknowledgements: The work is supported by the EU FP7 ICT projects SOA4All (IP 215219) and SEALS (e-Infrastructures 238975). We gratefully acknowledge the insights of project participants Barry Bishop, Reto Krummenacher and Carlos Pedrinaci, as well as the contributions of all members of the OASIS SEE Technical Committee.

References

1. Haller, A., Cimpian, E., Mocan, A., Oren, E., Bussler, C.: WSMX - A Semantic Service-Oriented Architecture. In: Proceedings of the International Conference on Web Services (ICWS2005), Orlando, Florida, USA (July 2005)
2. Cabral, L., Domingue, J., Galizia, S., Gugliotta, A., Norton, B., Tanasescu, V., Pedrinaci, C.: IRS-III: A broker for semantic web services based applications. In: Proceedings of the 5th International Semantic Web Conference (ISWC2006), Athens, Georgia, USA (Nov 2006)
3. Verma, K., Gomadam, K., Sheth, A., Miller, J., Wu, Z.: The METEOR-S Approach for Configuring and Executing Dynamic Web Processes. Technical report, LSDIS (24 June, 2005)
4. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: Semantic markup for web services. Available at `http://www.daml.org/services/owl-s/1.1/overview/` (November 2004)
5. Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services. Springer (2006)
6. Vitvar, T., Kopecky, J., Viskova, J., Fensel, D.: Wsmo-lite annotations for web services. In Hauswirth, M., Koubarakis, M., Bechhofer, S., eds.: Proceedings of the 5th European Semantic Web Conference. Number 5021 in LNCS, Springer Verlag (2008)
7. OASIS SOA Reference Model TC: Reference model for service oriented architecture 1.0. Technical report, OASIS (October 2006)
8. Kifer, M.: Rule interchange format: The framework. In: RR '08: Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems. Number 5341 in LNCS, Springer (2008) 1–11
9. Norton, B., Cabral, L., Nitzsche, J.: Ontology-based translation of business process models. In: Proceedings of 4th International Conference on Internet and Web Applications and Services (ICIW 2009), IEEE Computer Society (2009) 481–486