Comprehensive service semantics and light-weight Linked Services: towards an integrated approach

Stefan Dietze, Neil Benn, Hong Qing Yu, Carlos Pedrinaci,
Bassem Makni, Dong Liu, Dave Lambert, John Domingue

Knowledge Media Institute, The Open University, MK7 6AA, Milton Keynes, UK
{s.dietze, n.j.l.benn, h.q.yu, c.pedrinaci, b.makni, d.liu, d.j.lambert, j.b.domingue}@open.ac.uk

Abstract. Semantics are used to mark up a wide variety of data-centric Web resources but, are not used in significant numbers to annotate online services—that is despite considerable research dedicated to Semantic Web Services (SWS). This is partially due to the complexity of comprehensive SWS models aiming at automation of service-oriented tasks such as discovery, composition, and execution. This has led to the emergence of a new approach dubbed *Linked Services* which is based on simplified service models that are easier to populate and interpret and accessible even to non-experts. However, such *Minimal Service Models* so far do not cover all execution-related aspects of service automation and merely aim at enabling more comprehensive service search and clustering. Thus, in this paper, we describe our approach of combining the strengths of both distinct approaches to modeling Semantic Web Services – "lightweight" Linked Services and "heavyweight" SWS automation – into a coherent SWS framework. In addition, an implementation of our approach based on existing SWS tools together with a proof-of-concept prototype used within the EU project NoTube is presented.

Keywords: Semantic Web Services, Linked Services, Linked Data, IPTV.

1 Introduction

The past decade has seen a wide range of research efforts in the area of Semantic Web Services (SWS), mainly aiming at the automation of Web service-related tasks such as discovery, orchestration or mediation via broker-based approaches. Building on formal service semantics, several conceptual models, such as OWL-S [14] and WSMO [9], and also standards such as SAWSDL [18] have been proposed which aim at formalizing semantic service descriptions usually covering aspects such as service capabilities, interfaces and non-functional properties. Besides, a considerable research community evolved around these SWS frameworks, providing, for instance, related annotation and execution tools [7].

While semantics are used to mark up a wide variety of data-centric resources on the Web, that does not apply to online services in significant numbers. The reasons for this are two-fold. Firstly, SWS research has for the most part targeted WSDL [22] or SOAP-based [21] Web services, which are not prevalent on the Web [4]. Secondly, due to the inherent complexity required to fully capture computational functionality, creating SWS descriptions has represented an important knowledge acquisition bottleneck and has required the use of rich knowledge representation languages and complex reasoners. There exists an inherent conflict between the need to capture comprehensive and meaningful service semantics – to allow reasoning-based automation of any sort – and the requirement to keep the costs for providing services descriptions low in order to simplify the modeling process and to ensure that efficient and scalable solutions can be implemented [17]. Hence, despite considerable amount of research dedicated to the SWS vision and the existence of a range of SWS-related projects, tools and specifications, so far there has been little take up of SWS technology within non-academic environments.

The prevalent lack of impact of SWS technology is particularly concerning since Web services – nowadays including a range of often more light-weight technologies beyond the WSDL/SOAP approach, such as RESTful services, HTTP GET-style requests or XML-feeds – are in widespread use throughout the Web where applications use distributed requests to combine and mash-up data from a variety of open data sources. Hence, the challenges SWS attempted to tackle are of even more crucial importance for today's highly distributed Web applications. These issues have led to the emergence of more simplified SWS approaches to which we shall refer here as "lightweight", such as WSMO-Lite [19] or the Micro-WSMO/hRESTs [10] approach which replace "heavyweight" service semantics with less comprehensive and less costly to produce service models that are represented in RDF(S), and hence, comply with the infrastructure of the growing *Semantic Web* [2]. Analogous to the *Linked (Open) Data (LOD)* term [3], this approach was recently dubbed as the *Linked Service* approach [17]. Due to the fact that such service annotations are much easier to produce and can be populated with references to widely established LOD vocabularies, they address a much wider audience and allow even non-SWS experts to describe and annotate services. However, while those models are easier to produce [4], they merely aim at enabling structured, semantics-enabled search by humans or automated service clustering. More expressive solutions are required to achieve greater levels of automation, for instance, dealing with matching service requests with extensive capability representations of available services, or with handling of data-level mismatches when executing a set of services in an orchestrated manner.

Therefore, here, we aim to combine the strengths of both distinctive SWS approaches – lightweight Linked Services and more heavyweight broker-based SWS automation – into a coherent SWS framework. By integrating collaborative and user-driven Web-scale service annotations with comprehensive SWS specifications, we benefit from both low cost for providing annotations and a high level of automation. This also has the benefit of enabling a range of matchmaking scenarios (from user-driven keyword matching to automated capability matchmaking).

Section 2 introduces work related to our research. Section 3 gives an overview of our approach and describes the approach and tools that were developed to support our two-stage approach, while Section 4 describes the deployment and evaluation of our work within in an EU research project.

2 Related work & background

The landscape of SWS is characterized by a number of conceptual models that, despite a number of common characteristics, remain essentially incompatible due to the different representation languages and expressivity utilized as well as because of conceptual differences. The main conceptual frameworks and specifications devised thus far include for instance WSMO [20], OWL-S [14]. SAWSDL [18], which in turn derives from WSDL-S [18]. The vast majority of the SWS initiatives were built upon the enrichment of WSDL Web services with semantics. It is only recently that researchers have started focusing on Web APIs and RESTful services. The main outputs of this recent research are SA-REST [18] and MicroWSMO [12]

Over the last few years, a significant portion of research on the SW has been devoted to creating what is referred to as LOD [3] which is based upon a set of principles, including the usage of HTTP URIs to provide information and allow access based on RDF and SPARQL. Since these principles were outlined, there has been a large uptake, most notably through DBpedia [1] and others that have produced a vast amount of linked datasets. While the great potential of this massive data space still remains largely unexploited, service-oriented computing has been argued to be a suitable approach to supporting the construction of advanced applications based on linked data [16].

2.1. Lightweight service annotation: the iServe Linked Services approach

In order to support annotation of a variety of services, such as WSDL services as well as REST APIs, the EC-funded project SOA4ALL[1], has developed iServe[2] a novel and open platform for publishing semantic annotations of services based on a direct application of linked data principles [17]. iServe supports publishing service annotations as linked data—Linked Services—expressed in terms of a simple conceptual model that is suitable for both human and machine consumption and abstracts from existing heterogeneity around service kinds and annotation formalisms. In particular iServe provides:

- Import of service annotations in a range of formalisms (e.g., SAWSDL, WSMO-Lite, MicroWSMO, OWL-S) covering both WSDL services and Web APIs;
- Means for publishing semantic annotations of services which are automatically assigned a resolvable HTTP URI;
- Support for content negotiation so that service annotations can be returned in plain HTML or in RDF for direct machine consumption;
- SPARQL endpoint allowing querying over the services annotations;
- REST API to allow remote applications to consume and provide annotations.
- Support for linking service annotations to existing vocabularies on the Web.

In order to cater for interoperability, iServe uses what can be considered the maximum common denominator between existing SWS formalisms which we refer to

[1] http://www.soa4all.eu/
[2] http://iserve.kmi.open.ac.uk

as the Minimal Service Model (MSM). The MSM, first introduced together with WSMO-Lite and hRESTS [19], is thus a simple RDF(S) ontology able to capture (part of) the semantics of both Web services and Web APIs in a common model. MSM is extensible to benefit from the added expressivity of other formalisms. The MSM, denoted with the 'msm' namespace in Fig. 1, defines *Services* as having a number of *Operations* each of which have an *Input*, *Output MessageContent*, and *Faults*. In turn, a MessageContent may be composed of *MessageParts* which may be *mandatory* or *optional*. iServe additionally uses the SAWSDL, WSMO-Lite and hRESTS vocabularies. The SAWSDL vocabulary captures in RDF the three main kinds of annotations over WSDL and XML Schema, including *modelReference*, *liftingSchemaMapping* and *loweringSchemaMapping* that SAWSDL supports. WSMO-Lite builds upon SAWSDL by extending it with a model specifying the semantics of the particular service annotations. It provides a simple RDFS ontology together with a methodology for expressing functional and non-functional semantics, and an information model for WSDL services based on SAWSDL's modelReference hooks. The hRESTS vocabulary extends the MSM with specific attributes for operations so as to allow modeling additional details necessary for Web APIs.

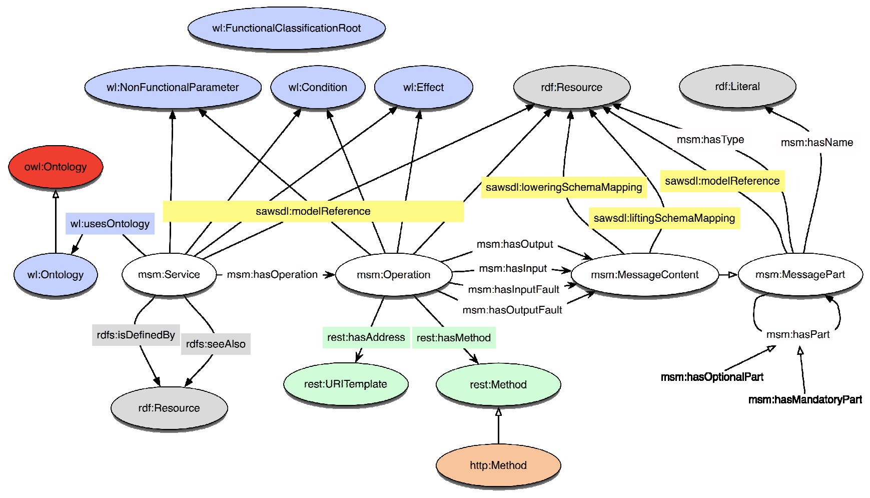


Fig. 1. iServe conceptual model for services – The Minimal Service Model and WSMO-Lite.

In order to support users in creating semantic annotations for services two editors have been developed: SWEET [12] (SemanticWeb sErvices Editing Tool) and SOWER (SWEET is nOt a Wsdl EditoR), which support users in annotating Web APIs and WSDL services respectively. However, SWEET and SOWER build on the assumption that either HTML documentation of services/APIs (SWEET) or WSDL files (SOWER) are available as starting point for annotation. In addition, while the iServe approach enables uptake of SWS technology by a wider audience, the automation and matchmaking scenarios which it facilitates are actually limited. The reason for that being that the MSM deliberately excludes execution aspects for the sake of simplicity and the lack of a commonly prescribed capability representation model.

2.2. Automated services brokerage: the IRS-III framework

IRS-III[3] [7] is a SWS execution environment which acts as a service broker – mediating between the goals of a client and relevant services that are deployed on the Web – striving for a high level of service automation. IRS-III adopts the WSMO conceptual model of services. The ultimate aim of the WSMO model of Web services is to be able to provide a well-defined semantics, which can then be interpreted by a reasoner to enable automatic discovery, selection, composition, mediation, execution, and monitoring of services [10]. As opposed to MSM, IRS-III directly covers execution-related aspects.

The WSMO conceptual model of services consists of the following core elements: *goal*, *mediator*, and *Web service*. These are described in a formal representation language, for instance, OCML [15] in the case of IRS-III. The functionality offered by a Web Service is captured by its *capability* description, which defines necessary *pre-* and *postconditions* as well as underlying *assumptions* and *effects* of the service. These are usually formalized as logical expressions. The means to interact with the Web service is captured by its *interface* definition.

Given that IRS-III directly aims at automating service execution related aspects, the interface covers *choreography* and *orchestration* descriptions. Choreography addresses the communication between the IRS-III broker and a Web service, and is described as so-called *grounding.* The IRS-III grounding mechanism supports REST-based, SOAP-based, and XML-RPC based services [11]. Grounding involves two processes referred to as *lifting* and *lowering*. Lowering involves transforming input parameters at the semantic level to data input to the service at the syntactic level. Lifting involves the opposite transformation, i.e. transforming the data output from the service at the syntactic level into an ontological object at the semantic level.

Orchestration addresses the problem of how to model functionality that is composed of several Web services. At the semantic level the orchestration is represented by a workflow model expressed in OCML, that describes the flow of control between the Web services. The IRS-III orchestration model supports the main control-flow primitives of sequence, selection, and repetition.

At runtime, IRS-III automatically discovers and invokes Web services suitable for a given client request, formulated as a goal instance, by selecting suitable services and executing these whilst adhering to any data, control flow and Web service invocation constraints. In principle, selection is based on comparing the capability descriptions of the request with the ones of the relevant SWS. Such matchmaking is currently supported, for instance, via (a) comparison and evaluation of logical expressions used in the capability descriptions, or (b) a hybrid approach [6] which combines similarity-computation via vector representations of SWS instances with (a). The IRS-III functionalities are exposed through a Java API[4] (details in [7]), and an HTTP-based

[3] http://technologies.kmi.open.ac.uk/irs - IRS: Internet Reasoning Service
[4] http://technologies.kmi.open.ac.uk/irs/irs3docs/api/index.html

REST API, which applications use to interact with IRS-III.

3 Two-stage service annotation and reasoning

In order to tackle the challenges introduced in Section 1, we aim at combining the two distinct SWS representation approaches

(R1) lightweight Linked Services (as facilitated by MSM), and
(R2) heavyweight SWS automation (as facilitated by WSMO).

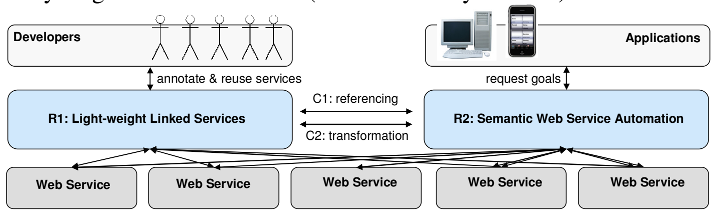


Fig. 2. From lightweight service annotations to heavyweight SWS automation - overall approach.

While these approaches currently co-exist without a well-defined relationship, we propose two different bi-directional correlations, which are under investigation:

(C1) service model cross-referencing,
(C2) service model transformation and augmentation.

Under (C1), we subsume all kinds of references between models across (R1) and (R2) as depicted in Fig. 2. For instance, a lightweight service annotation (MSM) could point to a heavyweight WSMO description that models the same service more exhaustively or vice versa. That would allow semantics to be exploited in (R1) as well as (R2) for reasoning of different sorts, for instance, to perform some clustering or filtering based on (R1) to reduce the amount of potentially interesting services for a given query in (R2). In addition, (C2) considers the transformation between models across (R1) and (R2), either manually or (semi-)automatically. Our current implementation builds upon existing SWS research namely WSMO and WSMO-Lite/MSM by integrating iServe and IRS-III. The remainder of this section describes the two approaches - (C1) and (C2) - in more detail.

3.1. Services model cross-referencing

Service model cross-referencing involves the formal definition of relationships between service models. The two main types of relationship are depicted in Figure 3.

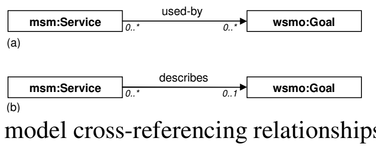


Fig. 3. Supported service model cross-referencing relationships.

(a) MSM instances referring to WSMO goal instances:
This involves specifying a link between an MSM instance and a corresponding WSMO goal description. Links of this kind define that the respective goal (*wsmo:Goal*) makes use of the service described by the respective *msm:Service*, i.e. for instance, the goal is linked to the service and potentially allows its discovery and execution as part of a more complex orchestration. Following that reference, developers are able to query the iServe repository via SPARQL or its API to (i) discover suitable services described via MSM, and then (ii) use a corresponding goal invocation URI to execute the service via IRS-III execution facilities. However, one assumption for such use cases is the existence of service models in both, iServe as well as IRS-III, which describe the same underlying service.

(b) MSM instances describing WSMO goal instances:
An additional link between MSM (iServe) and WSMO (IRS-III) is established by annotating the interface for achieving a particular goal (*wsmo:Goal* within IRS-III) itself as a minimal service description (*msm:Service*) within iServe. This is feasible and useful since WSMO goals within IRS-III are exposed via a REST-API and hence, each goal constitutes a particular service itself, which makes use of one or more actual Web services/APIs to provide a specific functionality. This has the benefit of allowing developers to query the MSM knowledge base in order to keep track of and discover WSMO goals. In that, complex functionalities – which might make use of a number of services – can be exposed via IRS-III and then be annotated within iServe as (higher level) services themselves.

3.2. Service model transformation and augmentation

Here, we consider the transformation and augmention of models across (R1) and (R2), either manually or (semi-)automatically. This involves transforming service descriptions based on one conceptual model of services (e.g. the MSM) into the other, e.g., WSMO and vice versa.

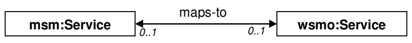


Fig. 4.Transformation between service representations across both conceptual models.

As can be seen from the previous sections, there is some overlap between the elements of a service description according to the MSM and the elements of a service description according to the WSMO conceptual model. This applies in particular to the service entity within both models. Here, we investigated the overlap between both schemas in order to establish potential mapping rules. The following figure depicts the core entities of WSMO and MSM, their relationships and their potential cross-model mapping. Please note, that for the sake of simplification, we left aside the WSMO elements goal and mediator, which have no expression in the MSM whatsoever.

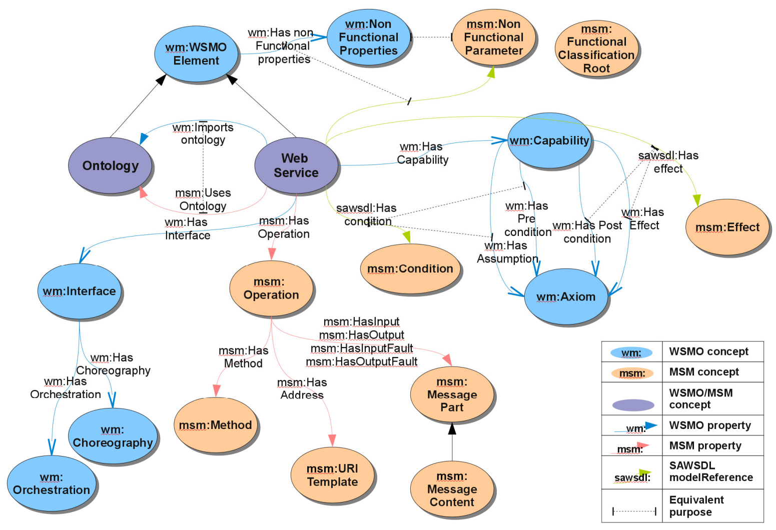


Fig. 5.MSM vs WSMO entities: relationships and mappings.

As depicted above, both models share a certain overlap, mainly relating to the core concepts such as *Ontology*, *Web Service* and *Non-Functional Parameter (Property)* and a number of properties which are equivalent. We foresee a bi-directional semi-automated transformation strategy between WSMO and MSM consisting of the following steps:

S1. Generating raw target model from source model.
S2. Semi-automatic augmentation of target model.

This transformation is making use of the mentioned model schema overlap and aims at generating raw target models (e.g. a WSMO service instance) from a given source model (e.g. a MSM service instance) as part of *S1*. *S2* then aims at semi-automatically enriching the generated service instance in order to create a fully target schema compliant service instance.

3.3. Implementation: service annotation and integration via SmartLink

In order to tackle some of the issues mentioned above and to approach integration of service models, a new services annotation and search tool was created, *SmartLink* ("SeMantic Annotation enviRonmenT for Linked services"). SmartLink allows annotation of REST-ful services based on the MSM from scratch, that is, without any pre-existing services documentation such as WSDL or HTML files, as assumed by existing iServe annotation tools (Section 2.1). Besides, SmartLink exploits an extension of the MSM schema including a number of additional non-functional properties. MSM-schema properties are directly stored in iServe, while additional properties are captured in a complementary RDF store based on OpenRDF Sesame[5]. Due to these extensions, we refer in the following to our service RDF store as *iServe+*. These non-functional properties are, for instance, *contact person, developer name, Quality of Service* (*QoS), development status, service license,* and *WSMO goal reference*. The latter property directly contributes to facilitate our vision of allowing MSM models to refer to existing WSMO goals which utilize the same service entity; that is, it facilitates our model referencing vision (Section 3.1) between MSM and WSMO models. In addition, by allowing developers to directly annotate existing REST-ful services and APIs, SmartLink directly provides another contribution to enable our service model integration vision (Section 3.1) based on allowing the annotation of WSMO goal requests – which in fact are REST-ful services themselves – as MSM service instances.

SmartLink currently provides mechanisms which enable the export of particular (MSM) service instances as RDF or human-readable HTML. In order to facilitate service model transformation and augmentation between MSM and WSMO as proposed in Section 3.2, current research deals with the establishment of an export mechanism of MSM service models as WSMO instances. While current implementation work is concerned with adding corresponding export facilities to SmartLink, model transformation is just enabled on a manual basis at the moment.

4 Case study: two-fold service annotation within NoTube

This section describes a first application of our approach in the context of the NoTube project[6] where the ultimate goal is to develop a network of services, connected through the use of semantics, to personalize consumption of digital (IP)TV content.

4.1. NoTube challenges

In order to illustrate the challenges with respect to service-related tasks, we describe one of the main use cases driven by the TV broadcast industry partners within the NoTube project – namely the requirement to provide personalized content and metadata delivery to users. Here, the basic feature is the matching of heterogeneous users' profiles, e.g. including interests, preferences and activity data, and user contexts (e.g. current location and viewing device) to filter and deliver TV content from a variety of sources. Addressing this particular use case in a service-oriented manner involves selecting and orchestrating between numerous services that provide various functionality, for instance, to aggregate users' topic interests based on their social networking activities, retrieve electronic program guide (EPG) data from various sources, and provide recommendations based on a dedicated algorithm. To support the highly service-oriented nature of the project, two major goals need to be supported: (1) support of distributed developers with lightweight service annotations,

[5] http://www.openrdf.org/
[6] http://www.notube.tv

and (2) support of application automation with Semantic Web Service brokerage. To support these goals, we deploy and adapt iServe and IRS-III as SWS frameworks.

4.2. Two-fold service semantics: implementation and integration within NoTube

Supporting lightweight NoTube service annotations via SmartLink and iServe
While the NoTube development takes place in a highly distributed setting and follows service-oriented principles, NoTube developers need to be provided with the means to document and search for appropriate services and data sources in order to build applications and higher-level services.

```
<rdf:RDF xmlns:so="http://www.purl.org/vocabularies/service-ontology#"
xmlns:msm="http://cms-wg.sti2.org/ns/minimal-service-model"
xmlns:saw="http://www.w3.org/ns/sawsdl#"...>

<rdf:Description
rdf:about="http://lupedia.ontotext.com/lookup#text2rdfa">
   <so:hasContactPerson>Stefan Dietze</so:hasContactPerson>
   <so:hasGoal>GET-LUPEDIA-ENTITIES-GOAL</so:hasGoal>
   <msm:hasInput
   rdf:resource="http://lupedia.ontotext.com/lookup/input#lookupText"/>
   <msm:hasOutput
   rdf:resource="http://lupedia.ontotext.com/lookup/output#lookupResult"/>
  …
   <so:hasOneLiner>Lookup of free text in DBPedia based on entity recognition and
   DBPedia lookup.</so:hasOneLiner>
   <msm:hasOperation
   rdf:resource="http://lupedia.ontotext.com/lookup/#text2rdfa"/>
   <sa:modelReference rdf:resource="http://www.service-
   finder.eu/ontologies/ServiceCategories#Multimedia"/>
   <sa:modelReference rdf:resource="http://www.service-
   finder.eu/ontologies/ServiceCategories#Content"/>
  …
</rdf:Description>
…
<rdf:Description
rdf:about="http://lupedia.ontotext.com/lookup/output#lookupResult">
   <sa:modelReference rdf:resource="http://dbpedia.org/data/Entity"/>
</rdf:Description
…
</rdf:RDF>
```

Listing 1. RDF-excerpt of LUPedia service description based on MSM.

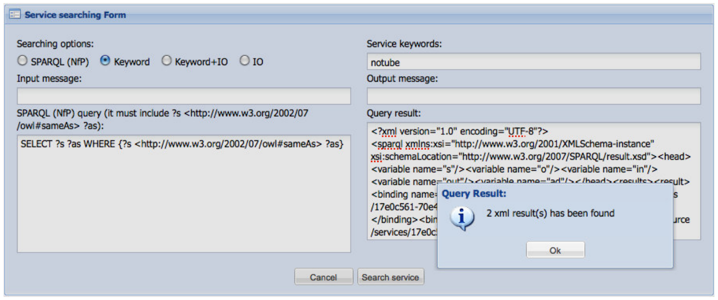
Hence, as an initial step, lightweight service semantics need to be generated, stored and exposed in an explorable way to support the NoTube developers in finding and reusing appropriate services. NoTube adopts the iServe environment by utilising the iServe+ and SmartLink tools which cater for additional NoTube-specific requirements (Section 3.3) which operates on top of the iServe RDF store. In addition, the general-purpose service taxonomy used by iServe (ServiceFinder ontology[7]) was extended with a service classification specific to the NoTube domain.

Listing 1 depicts an extract of the RDF description of a particular NoTube service (LUPEDIA[8]) which performs a lookup of free text in DBPedia in order to allow enrichment of EPG metadata with additional DBPedia entities. Besides the utilisation

[7] http://www.service-finder.eu/ontologies/ServiceCategories
[8] http://lupedia.ontotext.com/

of model references to external vocabularies – please note the highlighted reference (`<sa:modelReference>`) at the bottom – the listing also highlights some of the integrative elements which had been utilized within NoTube. For instance, the `<so:hasGoal>`-property refers to a particular WSMO goal instance within IRS-III to cater for our model referencing approach (Section 3.1).

The following screenshot depicts the query interface of SmartLink, which allows to query for services. Service matchmaking is being achieved by matching a set of core properties (input, output, keywords) or submitting more comprehensive SPARQL queries.



Searching options: ( ) SPARQL (NfP) [✓ Keyword] ( ) Keyword+IO ( ) IO

Service keywords: notube

Input message:

Output message:

SPARQL (NfP) query (it must include ?s <http://www.w3.org/2002/07/owl#sameAs> ?as):

```
SELECT ?s ?as WHERE {?s <http://www.w3.org/2002/07/owl#sameAs> ?as}
```

Query result:

```
<?xml version="1.0" encoding="UTF-8"?>
<sparql xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2007/SPARQL/result.xsd"><head>
<variable name="s"/><variable name="o"/><variable name="in"/>
<variable name="out"/><variable name="ad"/></head><results><result>
<binding name=                                                  s
/17e0c561-70e4
</binding><bin                                                  urce
/services/17e0c
```

Query Result:

2 xml result(s) has been found

Fig. 6. SmartLink service query interface as utilized in NoTube.

Support of service automation with Semantic Web Service brokerage

The IRS-III acts a middleware component for the NoTube project with the purpose of automatically finding, combining and invoking relevant Web Services based on goals specified by NoTube application developers. By annotating existing services via WSMO, we abstract from the Web service implementations, ensuring a high level of autonomy and flexibility. That is, service consumers treat goals as black boxes which provide abstract functionalities achievable by IRS-III in terms of reasoning on WSMO service instances to discover and orchestrate suitable services. Goals are requested via the IRS-III REST API (Section 2.2), and, as such, each individual goal achievement request constitutes a service itself.

The following code excerpt shows the WSMO description (in OCML) of the same NoTube service (LUPedia) and a corresponding WSMO goal (`GET-LUPEDIA-ENTITIES-GOAL`). This code has been obtained by manually applying our transformation strategy from Section 3.2. Besides the I/O definitions ("`has-text`" and "`has-lupedia-entities`") the listing also shows the grounding definitions that determine how the WSMO goal invocation instance is grounded to the underlying Web service. The grounding consists of three key definitions (highlighted in the Listing):

- The definition of the service listener (`GET-LUPEDIA-ENTITIES-WS-PUBLISHER-INFORMATION`);
- The lowering definition defining the lowering from the semantic level (WSMO/OCML instances) into the input parameters of the Web service (`LOWER-FOR-GET-LUPEDIA-ENTITIES-GOAL`, not shown in full detail);
- The lifting definition which describes the lifting of service execution results into WSMO/OCML instances (`LIFT-FOR-GET-LUPEDIA-ENTITIES-GOAL`, not shown in full detail). The lifting defines a rule for parsing and handling the XML result of

the LUPedia service (see also [11])

```
(DEF-CLASS GET-LUPEDIA-ENTITIES-GOAL (GOAL)
              ((HAS-INPUT-ROLE :VALUE has-text)
               (HAS-OUTPUT-ROLE :VALUE has-lupedia-entities)
               (has-text :TYPE String)
               (has-lupedia-entities :TYPE List)))
  …
(DEF-CLASS GET-LUPEDIA-ENTITIES-WS-PUBLISHER-INFORMATION (PUBLISHER-INFORMATION)
              ((HAS-WEB-SERVICE-HOST :VALUE "lupedia.ontotext.com")
     (HAS-WEB-SERVICE-LOCATION :VALUE "/lookup/text2xml")))

(DEF-RULE LOWER-FOR-GET-LUPEDIA-ENTITIES-GOAL
     …)
(DEF-RULE LIFT-FOR-GET-LUPEDIA-ENTITIES-GOAL
      …
      (extract-lupedia-entities-from-xml ?xml ?list-of-lupedia-entities)
      if
      (= ?list-of-lupedia-entities
         (setofall ?lupedia-entity
                 (and
                 (#_xml:rootElement ?xml ?rootEl)
                 (#_xml:contents ?rootEl ?rootContents)
                 (member ?lookupsEl ?rootContents)
                 (#_xml:tag ?lookupsEl "lookups")
                 (#_xml:contents ?lookupsEl ?lookupsContents)
                 (member ?instanceURIEl ?lookupsContents)
                 (#_xml:tag ?instanceURIEl "instanceUri")
                 (#_xml:contents ?instanceURIEl (?instanceURIContents))
                 (#_xml:value ?instanceURIContents ?instance-uri)
                 (member ?classURIEl ?lookupsContents)
                 (#_xml:tag ?classURIEl "instanceClass")
                 (#_xml:contents ?classURIEl (?classURIContents))
                 (#_xml:value ?classURIContents ?class-uri)
                 (= ?lupedia-entity (#_LUPediaEntity
                                         ?instance-uri
                                         ?class-uri)))))))
```

Listing 2. WSMO/OCML-code of LUPedia service.

Integration aspects between MSM and WSMO within NoTube

Section 3 introduced two methods for integrating the MSM and WSMO approaches: (a) Service model cross-referencing, and (b) Service model transformation. Within NoTube, the service model cross-referencing approach as described in Section 3.1 was implemented in two ways: by including a property in the extended MSM schema that provides a link to a corresponding WSMO goal description in the IRS-III execution environment (as illustrated by Listing 1). Furthermore, each WSMO goal invocation URI, that is the REST API goal achievement request which itself represents a REST-ful service for invoking some particular functionality, is also represented as a service following the extended MSM. That allows to expose higher level functionalities – achieved by orchestrating a number of heterogeneous services – as services themselves. Due to a lack of automated model transformation mechanisms so far and the lack of use cases requiring models being used in both representations, service instances had so far been transformed manually between WSMO and MSM. For instance, the service description in Listing 1 was generated by following the transformation procedure introduced in Section 3.2 to generate the service instance illustrated by Listing 2.

4.3. Lessons learned

From our initial use case, a few observations have been made which will shape our future efforts related to our two-fold services annotation and reasoning approach. While it was fairly easy to gather lightweight service semantics within NoTube by encouraging developers in the project to directly annotate their services via SmartLink, the lack of service automation and execution support provided by our extended MSM models, and, more importantly, the current tool support, made it necessary to transform and augment these models to expose them via IRS-III, i.e. as WSMO models within IRS-III, in order to perform more execution-oriented tasks. While transformation currently was achieved manually, future work will be dedicated to minimize this effort by striving for (semi-)automated transformation as sketched in Section 3.2.

The recommendation of LOD model references via open APIs – SmartLink currently uses WATSON[9] – proved very useful to aid the population of our iServe+ store. However, due to the increasing number of LOD datasets – strongly differing in terms of quality and usefulness – it might be necessary in the future to select recommendations only based on a controlled subset of the LOD cloud in order to reduce available choices.

With respect to service automation and brokerage, WSMO and IRS-III provide certain facilities to define service orchestrations or to achieve automated service selection [5]. However, while SWS frameworks strive for fully automated service brokerage, current tools and technologies do not facilitate that vision and allow only a very limited degree of actual automation. Still, the execution-oriented nature of WSMO/IRS-III provided a number of benefits when dealing with highly heterogeneous services. For instance, NoTube benefited from applying our rule-based definition of lifting- and lowering mechanisms [11] to map between heterogeneous service input and output schemas – e.g., based on JSON, RDF or XML – and the knowledge-level representations of services, to allow some further reasoning-based processing of data.

However, while our integrative approach proved useful in the sense that it supported required services discovery and automation scenarios within NoTube, maintaining services models following two distinct representation approaches turned out to be a costly task triggering the need for further investigation.

5 Conclusions

We have described a two-stage approach to semantic service representation and reasoning, aiming at a combination of the strengths of two distinctive methods – lightweight Linked Services and more heavyweight broker-based SWS automation – into a coherent SWS framework. The paper argued that by integrating collaborative and user-driven Web-scale service annotations with comprehensive SWS specifications, application developers benefit from both low cost for providing annotations and a high level of automation. In that, while taking advantage of service

[9] http://watson.kmi.open.ac.uk/WatsonWUI/

models produced by a large non-expert audience, both structured search for service instances by humans as well as automation of service tasks is supported to some extent.

In our vision, integration between lightweight service annotations and comprehensive SWS specifications is achieved by different means of (a) model cross-referencing and (b) model transformation and augmentation. Based on this vision we proposed a consistent approach of integrating a set of SWS-related tools and service models aiming at interoperability between lightweight service annotations and heavyweight service specifications. Besides, an application of our approach within the EU research project NoTube was presented as a proof-of-concept prototype.

While the current solution provides an overall framework for integrated service models which support different levels of automation, future work needs to investigate automated model transformation mechanisms in order to support the seamless integration of instances across distinct service models schemas. However, it might be argued, that there exists only an insufficient overlap between MSM and WSMO which does not support a more automated means of transformation as such. Besides, as mentioned above, maintaining services models following two distinct representation paradigms leads to additional effort. As additional downside, we like to point out that existing SWS brokerage technologies, such as IRS-III, support automation only to a certain extent.

In these respects, our future work will also investigate on (a) different levels of services automation, ranging from simple I/O matchmaking to capability matchmaking and execution handling, (b) their feasibility and usefulness and (c) possibilities to extend light-weight approaches, such as MSM, in order to support higher levels of automation.

6 Acknowledgments

This work is partly funded by the European project NoTube. The authors would like to thank the European Commission as well as all partners of the NoTube project for their support.

7 References

[1] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2008). Dbpedia: A nucleus for a web of open data. In Proceedings of 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference (ISWC+ASWC 2007), pages 722–735.

[2] Berners-Lee, T., Hendler, J., Lassila, O (2001). "The Semantic Web". Scientific American Magazine. retrieved March 29, 2009.

[3] Bizer, C., T. Heath, et al. (2009). "Linked data - The Story So Far." Special Issue on Linked data, International Journal on Semantic Web and Information Systems (IJSWIS).

[4] Davies, J., Domingue, J., Pedrinaci, C., Fensel, D., Gonzalez-Cabero, R., Potter, M., Richardson, M., and Stincic, S. (2009). Towards the open service web. BT Technology Journal, 26(2).

[5] Dietze, S., Benn, N., Domingue, J., Conconi, A., and Cattaneo, F.: "Interoperable Multimedia Metadata through Similarity-based Semantic Web Service Discovery". In Proceedings of 4th International Conference on Semantic and Digital Media Technologies (SAMT '09), 2--4 December 2009, Graz, Austria.

[6] Dietze, S., Benn, N., Domingue, J., Conconi, A., and Cattaneo, F. (2009) Two-Fold Semantic Web Service Matchmaking – Applying Ontology Mapping for Service Discovery, 4th Asian Semantic Web Conference, Shanghai, China.

[7] Domingue, J., Cabral, L., Galizia, S., Tanasescu, V., Gugliotta, A., Norton, B., Pedrinaci, C.: "IRS-III: A broker-based approach to Semantic Web Services", Jounal of Web Semant, pp. 109-132. Elsevier Science Publishers B. V, 2008.

[8] Farrell, J., and Lausen, H. 2007. Semantic Annotations for WSDL and XML Schema. http://www.w3.org/TR/sawsdl/. W3C Candidate Recommendation 26 January 2007.

[9] Fensel, D.; Lausen, H.; Polleres, A.; de Bruijn, J.; Stollberg,, M.; Roman, D.; and Domingue, J. 2007. Enabling Semantic Web Services: The Web Service Modeling Ontology. Springer.

[10] Kopecky, J.; Vitvar, T.; and Gomadam, K. 2008. MicroWSMO. Deliverable, Conceptual Models for Services Working Group, URL: http://cms-wg.sti2.org/TR/d12/v0.1/20090310/d12v01_20090310.pdf.

[11] Lambert, D., and Domingue, J. (2008) Grounding semantic web services with rules, Workshop: Semantic Web Applications and Perspectives, Rome, Italy

[12] Maleshkova, M., Pedrinaci, C., and Domingue, J. (2009). Supporting the creation of semantic restful service descriptions. In Workshop: Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2) at 8th International Semantic Web Conference.

[13] Maleshkova, M., Kopecky, J., and Pedrinaci, C. (2009). Adapting SAWSDL for semantic annotations of restful services. In Workshop: Beyond SAWSDL at OnTheMove Federated Conferences & Workshops.

[14] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K. (2004). OWL-S: Semantic Markup for Web Services. Member submission, W3C. W3C Member Submission 22 November 2004.

[15] Motta, E., Reusable Components For Knowledge Modelling: Case Studies in Parametric Design Problem Solving. IOS Press, ISBN I 58603 003 5, 1999.

[16] Pedrinaci, C., Domingue, J., and Reto Krummenacher (2010) Services and the Web of Data: An Unexploited Symbiosis, Workshop: Linked AI: AAAI Spring Symposium "Linked data Meets Artificial Intelligence".

[17] Pedrinaci, C., Liu, D., Maleshkova, M., Lambert, D., Kopecky, J., and Domingue, J. (2010) iServe: a Linked Services Publishing Platform, Workshop: Ontology Repositories and Editors for the Semantic Web at 7th Extended Semantic Web Conference.

[18] Sheth, A. P., Gomadam, K., and Ranabahu, A. (2008). Semantics enhanced services: Meteor-s, SAWSDL and SA-REST. IEEE Data Eng. Bul l., 31(3):8–12.

[19] Vitvar, T.; Kopecky, J.; Viskova, J.; and Fensel, D. 2008. Wsmo-lite annotations for web services. In Hauswirth, M.; Koubarakis, M.; and Bechhofer, S., eds., Proceedings of the 5th European SemanticWeb Conference, LNCS. Berlin, Heidelberg: Springer Verlag.

[20] WSMO Working Group (2004), D2v1.0: Web service Modeling Ontology (WSMO). WSMO Working Draft, (2004). (http://www.wsmo.org/2004/d2/v1.0/).

[21] World Wide Web Consortium, W3C: Simple Object Access Protocol, SOAP, Version 1.2 Part 0: Primer, (2003). (http://www.w3.org/TR/soap12-part0/).

[22] World Wide Web Consortium, W3C: WSDL: Web services Description Language (WSDL) 1.1, (2001). (http://www.w3.org/TR/2001/NOTE-wsdl20010315)