Semantic Web Service Selection with SAWSDL-MX

Matthias Klusch and Patrick Kapahnke

German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, Saarbrücken, Germany
klusch@dfki.de, patrick.kapahnke@dfki.de

Abstract. In this paper, we present an approach to hybrid semantic Web service selection of semantic services in SAWSDL based on logic-based matching as well as text retrieval strategies. We discuss the principles of semantic Web service description in SAWSDL and selected problems for service matching implied by its specification. Based on the result of this discussion, we present different variants of hybrid semantic selection of SAWSDL services implemented by our matchmaker called SAWSDL-MX together with preliminary results of its performance in terms of recall/precision and average query response time. For experimental evaluation we created a first version of a SAWSDL service retrieval test collection called SAWSDL-TC.

1 Introduction

As a W3C recommendation dated August 28, 2007, the SAWSDL[1] specification proposes mechanisms to enrich Web services described in WSDL[2] (Web Service Description Language) with semantic annotations. However, there is no SAWSDL semantic service matchmaker publicly available to the community yet. To fill this gap, we initially adopt the ideas of semantic Web service matching of our hybrid matchmakers OWLS-MX and WSMO-MX (see [8, 6]), for service description languages OWL-S[3] and WSML respectively, to this environment. A detailed discussion of the SAWSDL specification, particularly addressing the problems arising for semantic Web service selection, is also given.

In this paper, we present the first version of our hybrid SAWSDL Web service matchmaker called SAWSDL-MX. It exploits both crisp logic-based matching (subsumption reasoning) and IR-based (text retrieval) matching. Our preliminary experimental analysis shows, that in line with OWLS-MX and WSMO-MX, hybrid matching can outperform both variants applied stand-alone in terms of recall and precision.

The remainder of this paper is structured as follows. After a brief introduction to SAWSDL and discussion of implied challenges of semantic service selection in

[1] http://www.w3.org/TR/sawsdl/
[2] http://www.w3.org/TR/wsdl/ and http://www.w3.org/TR/wsdl20/
[3] http://www.daml.org/services/owl-s/1.1/

section 2, the hybrid matching approach of SAWSDL-MX is described in detail in section 3. Section 4 presents the architecture and implementation details of SAWSDL-MX. Preliminary results of our experimental evaluation of SAWSDL-MX over a initial test collection SAWSDL-TC1 in terms of recall, precision and average query response time are shown in 5. We comment on related work in section 6 and conclude in section 7.

2 SAWSDL Services

In the following, a brief introduction of the semantically enabled service description language SAWSDL is given. Language specific problems for semantic service discovery arising from the W3C recommendation and methods of resolution and assumptions for avoiding them respectively are also discussed.

SAWSDL is designed as extension of WSDL enabling service providers to enrich their service descriptions with additional semantic information. For this purpose, the notion of *model reference* and *schema mapping* have been introduced in terms of XML attributes that can be added to already existing WSDL elements as depicted in figure 1. More precisely, the following extensions are used for annotation:

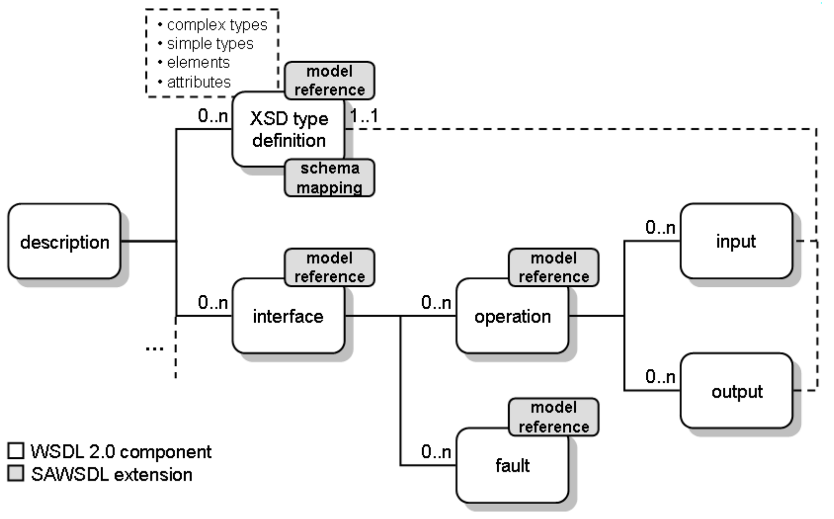

Fig. 1. SAWSDL extensions of WSDL interface components

– *modelReference*: A *modelReference* points to one ore more concepts with equally intended meaning expressed in an arbitrary semantic representation language. They are allowed to be defined for every WSDL and XML Schema element, though the SAWSDL specification defines their occurrence only

in WSDL interfaces, operations, faults as well as XML Schema elements, complex types, simple types and attributes. The purpose of a model reference is mainly to support automated service discovery.
- *liftingSchemaMapping*: Schema mappings are intended to support automated service execution by providing rules specifying the correspondences between semantic annotation concepts defined in a given ontology (the "upper" level) to the XML Schema representation of data actually required to invoke the Web service using SOAP (the "lower" level), and vice versa. A *liftingSchemaMapping* describes the transformation from the "lower" level in XML Schema up to the ontology language used for semantic annotation.
- *loweringSchemaMapping*: The reference tag *loweringSchemaMapping* describes the transformation from the "upper" level of a given ontology to the "lower" level in XML Schema.

Since the specification of SAWSDL does not restrict the developer of a semantic service in SAWSDL to a particular ontology language, any service selection has to cope with the implied semantic interoperability problem of both heterogeneous ontologies and heterogeneous ontology languages. Therefore, as an initial starting point, we restricted our inital SAWSDL service matchmaker to "understand" only the standard OWL[4]. More concrete, we assume for SAWSDL-MX 1.0 that model references in SAWSDL service offers and requests are pointing to ontological concepts exlcusively defined in OWL-DL. That allows to apply standard subsumption reasoning used for OWL-S matchmaking such as in [14, 4, 8]. Besides, there is no retrieval test collection for SAWSDL publicly available yet, but for OWL-S, namely OWLS-TC, which we converted semi-automatically into SAWSDL services such that we could use the resulting SAWSDL-TC for initially evaluating our matchmaker.

Another problem with the SAWSDL specification with respect to service matching is that so-called *top-level annotation* and *bottom-level annotation* are defined as to be considered independent from each other. The term *top-level annotation* describes the case, where a complex type or element definition of a message parameter is described by a model reference as a whole. A *bottom-level annotation* pursues the idea of semantically annotating the parts that are contained inside the definition of a complex type or element. However, it remains unclear how to evaluate matching *between* top-level and low-level annotated parameters, or which one to prefer if both levels are available. To circumvent this problem, we decided to rely on top-level annotations of upper parameter type definitions, and ignore bottom-level annotation in the first version of our matchmaker. In addition to that, element and type definition specifying a message component can be annotated at the same time. The specification does not imply a solution for this case either, so we decided to rely on the annotation directly attatched to the referenced XML Schema object if available.

Further, multiple references to multiple ontologies defined in different languages and formats such as logic theories, plain text documents or structured

[4] http://www.w3.org/2004/OWL/

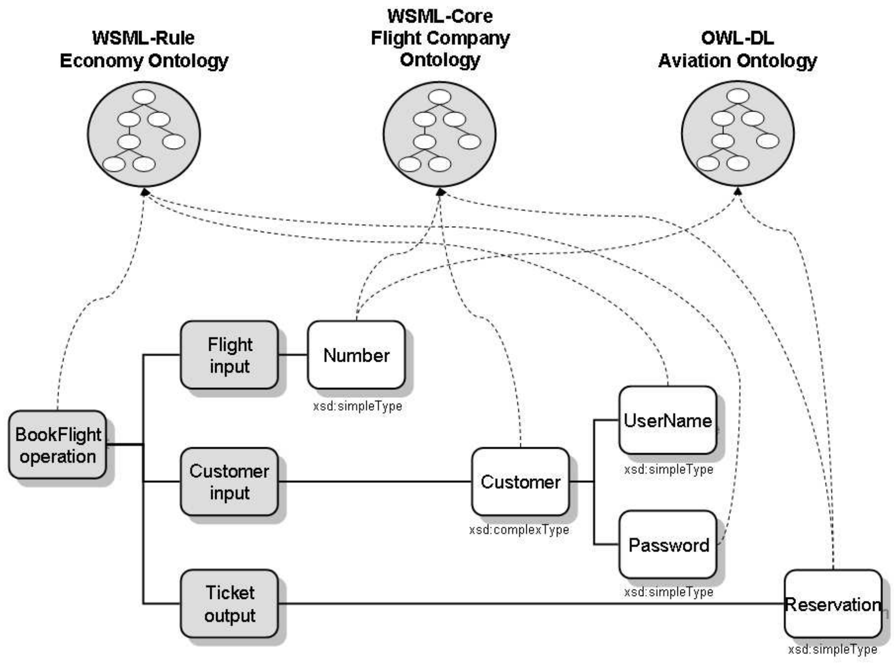

Fig. 2. SAWSDL service example

thesauri can be used to describe the semantic of even the *same* element. Therefore, a matchmaker, in principle, cannot know whether these different types of semantic descriptions of the element are intended to be treated as complementary or equivalent. In the first case, how to aggregate the complementing descriptions, in the latter case, which one to select best for further processing? This opens up a wide range of pragmatic approaches to deal with this for service matching. SAWSDL-MX 1.0 checks only the first model reference of an element. However, different variants dealing with multiple model references connected to a single object are topic of further development, since they are to be treated as sets without order. One possible approach would be to check every combination of request and service offer reference part and perform some kind of aggregation afterwards.

To illustrate this problem by example, consider figure 2: A flight company offers a WSDL Web service with different operations concerning flight booking (*BookFlight* operation), account administration (omitted in the picture), and so on. The *BookFlight* operation is defined to take information of the desired flight (*Flight* input) and customer information in form of a tuple containing a user name and appropriate password (*Customer* input) as input parameters and delivers information about the ticket reservation (*Ticket* output).

To support automated Web service selection, this service is semantically annotated in compliance with the SAWSDL specification as shown in the figure.

In particular, the service developer of the flight company uses WSML-Core, WSML-Rule and OWL-DL concept descriptions for service element annotation. As a consequence, a matchmaker agent cannot perform single language-specific reasoning and matching mechanisms but has to apply an appropriate combination of them instead. This problem can be straight-forwardly solved by use of language mappings available for WSML-Core and OWL-DL[5] but remains hard to solve for comparing concepts in WSML-Rule and OWL-DL.

Further, in the example, the XML Schema description attached to the service input element *Customer* contains annotations for the compound complex type as well as the simple types (referenced by the elements contained in the complex type, element nodes are omitted in the picture). How to handle this situation? Selecting only one annotation level may neglect additional information while looking at all references as a conjunction of ontological concepts can lead to either logical inconsistencies, or is not possible due to incomparable description languages. This problem is exaggerated in the example by providing multiple references (multiple levels of annotations) for the same element. SAWSDL-MX 1.0 only checks the top-level annotation of the most generic element of the XML Schema description of a service parameter.

3 Service Matching with SAWSDL-MX

In the following, we describe one approach to SAWSDL-service selection which we implemented in an initial version of a matchmaker called SAWSDL-MX based on the assumptions stated above. SAWSDL-MX performs service selection in terms of logic-based, syntactic (text similarity-based) and hybrid matching of I/O parameters defined for potentially multiple operations of a Web service interface (signature matching)[6]. As service requests, standard SAWSDL Web service definition documents are used. This approach is particularly inspired by the hybrid semantic service matchmakers OWLS-MX [8] and WSMO-MX [6] for OWL-S and WSML.

3.1 Service Interface Matching

The matching process of SAWSDL-MX on the service interface level is performed as follows. For every pair of service offer O and service request R, every combination of their operations is evaluated by either logic-based matching, text

[5] http://www.wsmo.org/TR/d16/d16.1/v0.21/

[6] For SAWSDL-MX 1.0, we assume only one interface but multiple operations per service. Extending the proposed service matching algorithm to services with even multiple interfaces only requires additionally combined valuation of the respective interface matching results. The restriction to signature matching for SAWSDL-MX 1.0 is due to the fact that, in SAWSDL, preconditions and effects can be added as input and output model references only, which makes it hard for any matchmaker to identify them as such in general, and before actually analyzing the name and content of referenced models in particular.

retrieval-based matching, or both. The matching of operations is described in more detail later.

In order to compute an optimal injective mapping of operations for service offer and request, SAWSDL-MX applies bipartite graph matching, where nodes in the graph represent the operations and the weighted edges are built from possible one-to-one assignments with their weights derived from the computed degree of operation match. If there exists such a mapping, then it is guaranteed that there exists an operation provided by the service offer for every operation a requester defined in her query. That is, there exists no request operation that cannot be provided by the service offer, disregarding the quality of match at this point.

As an example, consider the service request and service offer given in figure 3. Every request operation RO_i (with $i \in \{1, 2\}$) is compared to every advertisement operation O_j (with $j \in \{1, 2, 3\}$) with respect to logic-based filters defined in the next section. In this example, RO_1 exactly matches with O_1, but fails for O_2 and O_3. O_3 is a weaker plug-in match for RO_2 (the subsumed-by match of RO_2 with O_2 is even weaker than a plug-in match). The best (max) assignment of matching operations is $\{\langle RO_1, O_1\rangle, \langle RO_2, O_3\rangle\}$.

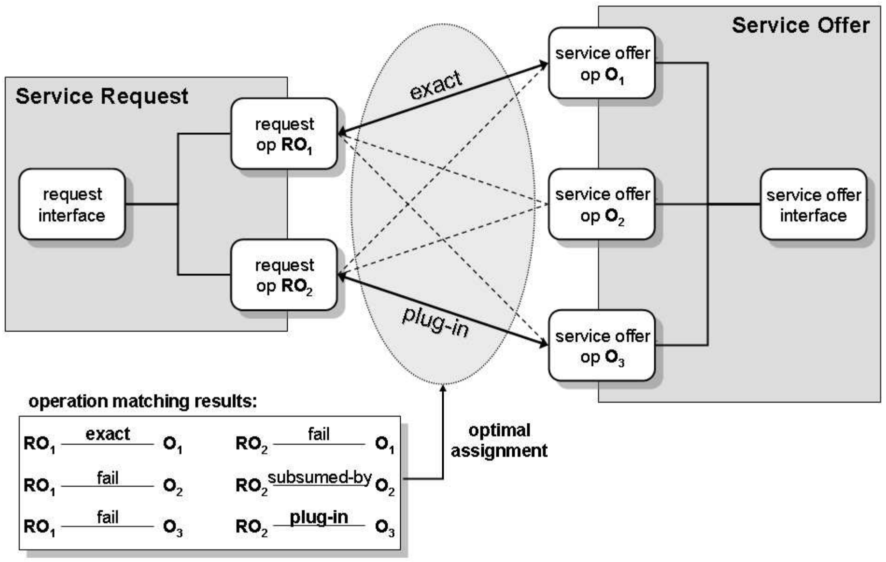

Fig. 3. Interface level matching of SAWSDL-MX

One conservative (min-max) option of determining the matching degree between service offer and request based on their pairwise operation matchings is to assume the worst result of the best operation matchings, to guarantee a fixed *lower* bound of similarity for *every* requested operation. This is what

SAWSDL-MX 1.0 is doing, so in this example shown in figure 3, the service offer is considered a *plug-in* match for the request. Other possibilities are to merge the operation matching results based on, for example, their average syntactic similarity values, and to provide more detailed feedback to the user on the operation matchings involved.

Please note that SAWSDL-MX aims at finding service matches solely based on single service offer documents. The problem of semantic Web service composition is somehow related, but additional state-based planning strategies have to be applied to solve this problem, which is out of the scope of this work. To accomplish on that, a Web service composition planner like e.g. OWLS-XPlan or SHOP2 could be considered (see [18, 19] for details).

3.2 Logic-based Operation Matching

As mentioned above, we assume for SAWSDL-MX 1.0 that model references in SAWSDL service offers and requests are pointing to ontological concepts exlcusively defined in OWL-DL or WSML-DL. That allows to apply standard subsumption reasoning for description logics (see [20]). Therefore, the logic-based operation matching part of SAWSDL-MX computes the degree of logic-based match for a given pair of service offer operation O_O and service request O_R by successively applying four filters of increasing degree of relaxation: *Exact*, *Plug-in*, *Subsumes* and *Subsumed-by*, which are, in essence, adopted from those of OWLS-MX 2.0 but modified in terms of an additional bipartite concept matching to ensure an injective mapping between offer and request concepts, if required. The reason of this modification is that previous experiments with OWLS-MX showed that many logic-based only failures could have been avoided by this additional constraint.

Exact match: Service operation O_O *exactly* matches service operation $O_R \Leftrightarrow$ ($\exists$ injective assignment $M_{in} : \forall m \in M_{in} : m_1 \in in(O_O) \wedge m_2 \in in(O_R) \wedge m_1 \equiv m_2) \wedge$ ($\exists$ injective assignment $M_{out} : \forall m \in M_{out} : m_1 \in out(O_R) \wedge m_2 \in out(O_O) \wedge m_1 \equiv m_2$). There exist a one-to-one mapping of perfectly matching inputs as well as perfectly matching outputs. Assuming that an operation fullfills a requesters need if every input can be satisfied and every requested output is provided, the assignments only require to be injective (but not bijective), thus additional available information not required for service invocation and additional provided outputs not explicitly requested are tolerated.

Plug-in match: Service operation O_O *plugs into* service operation $O_R \Leftrightarrow$ ($\exists$ injective assignment $M_{in} : \forall m \in M_{in} : m_1 \in in(O_O) \wedge m_2 \in in(O_R) \wedge m_1 \sqsupseteq m_2) \wedge$ ($\exists$ injective assignment $M_{out} : \forall m \in M_{out} : m_1 \in out(O_R) \wedge m_2 \in out(O_O) \wedge m_2 \in lsc(m_1)$). The filter relaxes the constraints of the *exact* matching filter by additionally allowing input concepts of the service offer to be arbitrarily more general than those of the service request, and advertisement output concepts to be direct child concepts of the queried ones.

Subsumes match: Service operation O_O *subsumes* service operation $O_R \Leftrightarrow$ ($\exists$ injective assignment $M_{in} : \forall m \in M_{in} : m_1 \in in(O_O) \wedge m_2 \in in(O_R) \wedge m_1 \sqsupseteq m_2) \wedge$ ($\exists$ injective assignment $M_{out} : \forall m \in M_{out} : m_1 \in out(O_R) \wedge m_2 \in$

$out(O_O) \wedge m_1 \sqsupseteq m_2$). This filter further relaxes constraints by allowing service offer outputs to be arbitrarily more specific than the request outputs (as opposed to the *plug-in* filter, where they have to be direct children). Thus, a *plug-in* can be seen as special case of a *subsumes* match resulting in a more fine-grained view at the overall service ranking.

Subsumed-by match: Service operation O_O is *subsumed by* service operation $O_R \Leftrightarrow$ ($\exists$ injective assignment $M_{in} : \forall m \in M_{in} : m_1 \in in(O_O) \wedge m_2 \in in(O_R) \wedge m_1 \sqsupseteq m_2$) $\wedge$ ($\exists$ injective assignment $M_{out} : \forall m \in M_{out} : m_1 \in out(O_R) \wedge m_2 \in out(O_O) \wedge m_2 \in lgc(m_1)$). The idea of the *subsumed-by* matching filter is to determine the service offers that the requester is able to provide with all required inputs and at the same time deliver outputs that are at least closely related to the requested outputs in terms of the inferred concept classification.

At this filtering step, services that offer equivalent or more specific outputs already have been discovered. The *subsumed-by* filter additionally returns service offers that provide more general output concepts, namely direct parents. These may be of value for a user to know, though it depends on the granularity of the matchmaker ontology. For example, it would not make sense to return a service operation providing information on vehicles, if the user explicitly requested information on a very special brand of a car which concept is inappropriately modelled as a direct child of the concept vehicles in the ontology.

The overall algorithm for logic-based matching of operations considers the filters in the following order based on the degree of relaxation: *exact* $>$ *plug-in* $>$ *subsumes* $>$ *subsumed-by* $>$ *fail*. The notion of *fail* applies to cases where none of the filtering tests succeeded.

3.3 Syntactic Operation Matching

In addition, SAWSDL-MX can perform syntactic-based matching based on selected token-based text similarity measures. That is, a syntactic similarity value is computed for every pair of service offer and request operation which is used to rank operations with same logic-based matching degree. The implemented similarity measures for SAWSDL-MX 1.0 are the same as for OWLS-MX, that are the *Loss-of-Information*, the *Extended Jaccard*, the *Cosine* and the *Jensen-Shannon* similarity measures. The architecture of SAWSDL-MX allows the integration of other text similarity measures such as those provided by SimPack[7] which is also used in the iMatcher matchmaker [7].

The weighted keyword vectors of inputs and outputs for every operation are generated by first unfolding the referenced concepts in the ontologies (as defined for standard tableaux reasoning algorithms). The resulting set of primitive concepts of all input concepts of a service operation is then processed to a weighted keyword vector based on TFIDF weighting scheme, the same is done with its output concepts. The text similarity of a service offer operation and a request operation is the average of the similarity values of their input and output vectors according to the selected text similarity measure.

[7] http://www.ifi.uzh.ch/ddis/research/semweb/simpack/

3.4 Hybrid Operation Matching

Inspired by OWLS-MX [8], SAWSDL-MX combines logic-based and syntactic-based matching to perform *hybrid* semantic service matching. There are different options of combination: A *compensative* variant using syntactic similarity measures in cases where none of the logic-based filters applies helps to improve the service ranking with respect to logic-based false negatives by re-considering them again in the light of their computed syntactic similarity. An *integrative* variant deals with problems concerning logic-based false positives by not taking the syntactic similarity of concepts into account only when a logical matching fails, but as a conjunctive constraint in each logical matching filter. Our experiments showed that OWLS-MX 2.0 using the integrative variant performs better than the original one with the complementary use of syntactic similarity. However, SAWSDL-MX 1.0 inherited the compensative variant from OWLS-MX 1.0, that is, only the logic-based subsumed-by filter is modified to a hybrid filter by integrative checking of syntactic simliarity of concepts, and the syntactic nearest-neighbour filter is compensative in the sense that it is only performed in case all other filters fail.

Subsumed-by match: Service operation O_O is *subsumed by* service operation $O_R \Leftrightarrow$ ($\exists$ injective assignment $M_{in} : \forall m \in M_{in} : m_1 \in in(O_O) \wedge m_2 \in in(O_R) \wedge m_1 \sqsupseteq m_2$) $\wedge$ ($\exists$ injective assignment $M_{out} : \forall m \in M_{out} : m_1 \in out(O_R) \wedge m_2 \in out(O_O) \wedge m_2 \in lgc(m_1)) \wedge sim_{IR}(O_R, O_O) \geq \alpha$. A *subsumed-by* match computed by hybrid matching additionally requires the IR-based similarity computed using one of the measures from $IR = \{LOI, ExtJacc, Cos, JS\}$ to be above a given threshold α. This helps to avoid logic-based false positives to be introduced by the pure logic-based variant of this filter.

Nearest-neighbour match: This filter compensates logic-based false negatives as described above. Its condition is $sim_{IR}(O_R, O_O) \geq \alpha$ and thus considers all services not already catched in previous filter steps whose IR-based similarity is above the threshold.

4 SAWSDL-MX Implementation

SAWSDL-MX 1.0 has been fully implemented in Java using the sawsdl4j[8] API (handling SAWSDL for WSDL 1.1) and the OWL API[9] for access to SAWSDL and OWL files, the DIG 1.1[10] as standard interface to handle SHOIQ knowledge base queries, and the Pellet[11] reasoner as inference engine for logic-based matchmaking.

Figure 4 gives an broad overview of the overall system architecture. Basically, SAWSDL-MX consists of the following components: *SAWSDL Matching Engine*, *Service Registry*, *Ontology Handlers*, *Local Matchmaker Ontology* and *Similarity*

[8] http://knoesis.wright.edu/opensource/sawsdl4j/
[9] http://owlapi.sourceforge.net/
[10] http://dig.sourceforge.net/
[11] http://pellet.owldl.com/

Measures. These are described in more detail in the following. From the perspective of service providers, SAWSDL-MX allows the registration of SAWSDL Web service offers at the service registry. For requesters, SAWSDL-MX provides an interface for submitting queries by means of a SAWSDL document specifying details about the desired service interface. After the service discovery process, the SAWSDL-MX matching engine returns a ranked list of service offers that match the query.

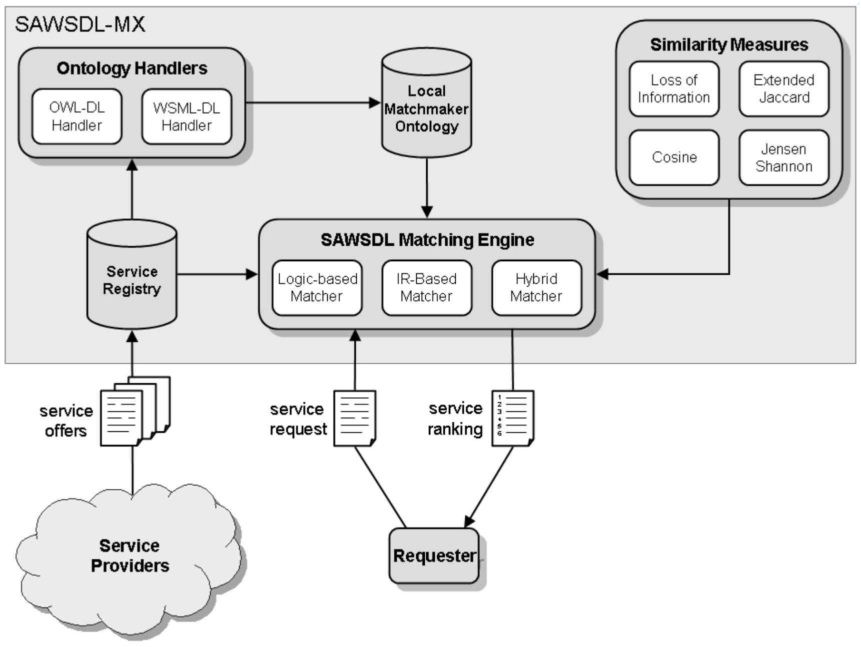

Fig. 4. SAWSDL-MX architecture

SAWSDL Matching Engine: The *SAWSDL Matching Engine* is the core component of SAWSDL-MX. It provides several matching variants of SAWSDL-MX 1.0 as described in previous sections: The *Logic-based Matcher* computes service ranking by means of crisp-logic subsumption reasoning and the logic-based matching filters described in section 3.2. The *IR-based Matcher* produces the ranked results using syntactic similarity measures as described in section 3.3. Finally, the *Hybrid Matcher* performs the combined approach of logic-based reasoning and syntactic similarity comparison as described in 3.4. The matching engine component is designed to provide easy integration of additional matching variants by means of Java interface implementation.

Service Registry: This component is the storage for service offers provided by service providers. It is accessed by the matching engine to produce the ranked results for a query.

Ontology Handlers: After the service registration process, the semantic annotations of a SAWSDL service (by means of model references) are processed using *Ontology Handlers*. Therefore, an appropriate handler able to parse and reason about the referenced ontology is selected and the concepts are stored locally to facilitate logic-based reasoning as well as concept unfolding for IR-based matching at query time. As for the matching engine component, the *Ontology Handlers* package is designed to allow the proper integration of additional knowledge representation formalisms by means of Java interfaces.

Local Matchmaker Ontology: This component is in fact part of the *ontology handlers* in the actual implementation but depicted as seperate component for reasons of clarity. The Local Matchmaker Ontology is a storage for all relevant concepts referenced by registered service offers as proposed in [8]. However, since SAWSDL allows the use of various knowledge representation formalisms, parts of the component relevant for certain ontology handlers are directly covered inside the handlers. In case of our current implementation of SAWSDL-MX, it consists of the Pellet reasoner, which is accessed by handlers able to process description logic based ontology languages via DIG 1.1. Currently, only the *OWL-DL Handler* is actually implemented, but expanding the system to WSML-DL is straight-forward, since they rely on subsets of the SROIQ description language, which is addressed by Pellet[12].

Similarity Measures: This package currently contains the four similarity measures *loss-of-information*, *extended Jaccard*, *cosine* and *Jensen-Shannon*. However, adding more variants for IR-based matching can be easily accomplished again via interfaces. An proprietary document indexing structure based on hash tables is also provided. The integration of additional syntactic similarity measures (e.g. from *SimPack*) and better indexing strategies is intended for following versions of SAWSDL-MX.

5 Evaluation of Performance

The experimental evaluation of the retrieval performace of the first version SAWSDL-MX focuses on measuring its recall and precision based on a first SAWSDL test collection semi-automatically derived from OWLS-TC 2.2[13] using the OWLS2WSDL[14] tool, as there is currently no standard test collection for SAWSDL matchmaking available. OWLS2WSDL transforms OWL-S service descriptions (and concept definitions relevant for parameter description) to WSDL through syntactic transformation. The collection consists of 894 Web services covering different application domains: education, medical care, food, travel, communication, economy and weaponry. For this set of service offers, 26 queries have been selected and relevance sets have been created for each of them. These where subjectively defined as relevant according to the standard TREC definition of binary relevance [16]. As the creation of this test collection has been done

[12] With exception of n-ary datatypes

[13] http://projects.semwebcentral.org/projects/owls-tc/

[14] http://projects.semwebcentral.org/projects/owls2wsdl/

by transforming OWL-S services contained in OWLS-TC 2.2, which provides services containing only one atomic process per description, every SAWSDL advertisement only contains a single interface with a single operation (but possibly multiple I/O's). Therefore and because all automatically derived model references exclusively point to OWL ontologies, this test collection can only be seen as a first attempt towards a commonly agreed testing environment for SAWSDL service discovery and our evaluation has to be considered as preliminary. The performance measures used for evaluation are defined as follows:

$$Recall = \frac{|A \cap B|}{|A|}, Precision = \frac{|A \cap B|}{|B|},$$

where A is the set of all relevant documents for a request and B the set of all retrieved documents for a request. The so-called *F1-measure* equally weights recall and precision and is defined as:

$$F1 = \frac{(2 \cdot Precision \cdot Recall)}{(Recall + Precision)}.$$

We adopt the prominent macro-averaging of precision. That is, we compute the mean of precision values for answer sets returned by the matchmaker for all queries in the test collection at standard recall levels $Recall_i$ ($0 \leq i < \lambda$). Ceiling interpolation is used to estimate precision values that are not observed in the answer sets for some queries at these levels; that is, if for some query there is no precision value at some recall level (due to the ranking of services in the returned answer set by the matchmaker) the maximum precision of the following recall levels is assumed for this value. The number of recall levels from 0 to 1 (in equidistant steps $\frac{n}{\lambda}, n = 1 \ldots \lambda$) we used for our experiments is $\lambda = 20$. Thus, the macro-averaged precision is defined as follows:

$$Precision_i = \frac{1}{|Q|} \times \sum_{q \in Q} \max\{P_o | R_o \geq Recall_i \wedge (R_o, P_o) \in O_q\},$$

where O_q denotes the set of observed pairs of recall/precision values for query q when scanning the ranked services in the answer set for q stepwise for true positives in the relevance sets of the test collection. For evaluation, the answer sets are the sets of all services registered at the matchmaker which are ranked with respect to their (totally ordered) matching degree.

The performance tests have been conducted on a machine with Windows 2000, Java 6, 1,7 GHz CPU and 2 GB RAM using SME^2 [15] as evaluation environment.

As can be seen in figure 5(a), the *hybrid* variant utilizing *cosine* measure performs best in both finding correct results among the top of the ranking as well as returning positives at high precision towards full recall. It is followed by pure IR-based service discovery (also using *cosine* measure), which is surprisingly at first glance, since it is assumed by the semantic Web community that

[15] http://projects.semwebcentral.org/projects/sme2/

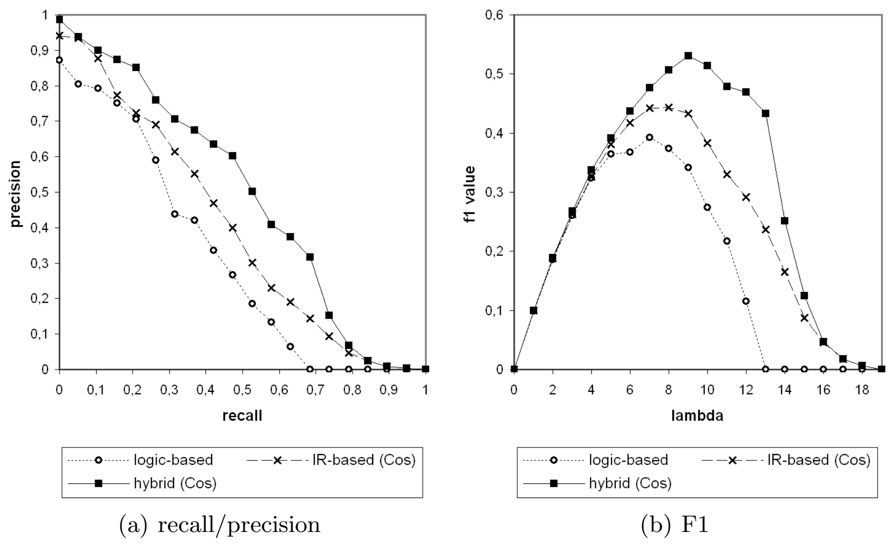

Fig. 5. Performance of SAWSDL-MX

semantically enabled ressource retrieval should be able to outperform standard information retrieval purely relying on syntactic information in general. However, as Wang et al. show in [17] exemplarily for OWL, the currently established Web ontology landscape provides mainly poor specification of concepts in terms of the used expressivity of description languages. In fact, many ontologies currently available are just simple taxonomies that do not rely on advanced features provided by for example OWL-DL, thus IR-based matching techniques are often good enough to compare service parameters. The crisp logic-based variant of SAWSDL-MX performs worst with respect to precision. This is mainly due to the problem with ontologies just described and due to the coarse-grained concept descriptions available. Equal consideration of recall and precision using the F1 measure yields the results given in figure 5(b), which recapitulates the observations. Regarding query response times, IR-based matching performs best, namely 1,7 seconds on average per query, while crisp logic-based matching takes 4,7 seconds on average and their combination in hybrid matching is the slowest (6,4 seconds). These evaluation results are in line with the performance of OWLS-MX and WSMO-MX and thus fortify the proposition that hybrid matching outperforms pure logic-based as well as IR-based matching in terms of recall and precision.

6 Related Work

To the best of our knowledge, there exist only very few implemented semantic service discovery systems for SAWSDL. [10] presents a solution to SAWSDL Web service discovery using UDDI registries called FUSION. In FUSION, any service description is classified at the time of its publishing and then mapped to UDDI to allow for fast lookups. In case of unknown semantic service requests reasoning has to be done at query time. In contrast to SAWSDL-MX, each service offer has only to satisfy one matching condition based on subsumption relationships inferred by a reasoner, thus the ranking is not affected by different degrees of logic-based match, neither does FUSION perform a syntactic or hybrid semantic match. Like SAWSDL-MX 1.0, FUSION is strictly bound to OWL-DL, since for each service, a semantic representation in terms of an individual of a predefined OWL concept is constructed. Lumina [11] developed in the METEOR-S project[16] follows a similar approach based on a mapping of WSDL-S (and later on SAWSDL respectively) to UDDI but performs syntactic service matching only. For a survey of semantic service matchmakers in general, we refer the interested reader to [9].

7 Conclusion

SAWSDL-MX performs hybrid semantic Web service matching for SAWSDL operations based on both logic-based reasoning and IR-based syntactic similarity measurement, and combines the results to provide a matching result for service interfaces with multiple operations. The requester formulates queries in terms of SAWSDL service interface descriptions and is presented a service ranking containing service offers from the local registry. The version SAWSDL-MX 1.0 presented in this paper has been implemented and evaluated in terms of recall and precision using a preliminary SAWSDL test collection called SAWSDL-TC1 which we derived from the existing collection OWLS-TC 2.2. As the experimental results show, hybrid matching of SAWSDL services can outperform both logic-based and IR-based matching in terms of precision at the cost of increased average query response time.

We are currently working on several aspects of SAWSDL service discovery and extensions of SAWSDL-MX. As SAWSDL is not restricted to semantically represent service components using a fixed knowledge representation formalism, the integration of additional ontology language support is intended. While description logics have already been discussed for the first version SAWSDL-MX 1.0, the support for languages originating from logic programming such as WSML-Flight and WSML-Rule is subject to our future work.

Besides, inspired by the monolithic logic-based semantic service matchmaker MaMaS [1, 2], we are currently working on an adaptive variant called SAWSDL-MXA which exploits means of ontology patching such as concept contraction and abduction combined with machine learning based on implicit feedback [5].

The semantic interoperability problem induced by the inevitable occurrence of heterogeneous ontologies used for semantic service annotation can be addressed by appropriate ontology alignment techniques [13]. In SAWSDL-MX, one option is to perform an additional matching of concept primitives (that

[16] http://lsdis.cs.uga.edu/projects/meteor-s/

are left undefined in the matchmnaker ontology) in unfolded concepts to be compared using a *shared minimum vocabulary* of requesters and providers like WordNet[17], or by consistent introduction of additional equivalence axioms to the local knowledge base of SAWSDL-MX [12].

SAWSDL-MX 1.0 and SAWSDL-TC1 are both publicly available at *semweb-central.org.*

References

1. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F. M., Mongiello, M.: Concept Abduction and Contraction in Description Logics. Proceedings of the 16th International Workshop on Description Logics (DL'03), Volume 81 - Sept. 2003
2. Colucci, S., Coppi, S., Di Noia, T., Di Sciascio, E., Donini, F. M., Pinto, A., Ragone, A.: Semantic-Based Resource Retrieval using Non-Standard Inference Services in Description Logics. Proceedings of Thirteenth Italian Symposium on ADVANCED DATABASE SYSTEMS Sistemi Evoluti per Basi di Dati (SEBD-2005), pp. 232-239, 2005
3. Euzenat, J., Valtchev, P.: Similarity-based ontology alignment in OWL-Lite. Proceedings of the European Conference on Artificial Intelligence ECAI, 333-337, 2004
4. Jaeger, M. C., Rojec-Goldmann, G., Liebetruth, C., Mühl G., Geihs K.: Ranked Matching for Service Descriptions Using OWL-S. KiVS 2005: 91-102, 2005
5. Joachims, T., Radlinski, F.: Search Engines that Learn from Implicit Feedback. Computer Volume 40, Issue 8, Aug. 2007 Page(s):34 - 40, 2007
6. Kaufer, F., Klusch, M.: WSMO-MX: A Logic Programming Based Hybrid Service Matchmaker. Proceedings of the 4th IEEE European Conference on Web Services (ECOWS 2006), IEEE CS Press, Zurich, Switzerland, 2006
7. Kiefer, C., Bernstein, A.: The Creation and Evaluation of iSPARQL Strategies for Matchmaking. Proceedings of the 5th European Semantic Web Conference (ESWC), Lecture Notes in Computer Science, Vol. 5021, pages 463–477, Springer-Verlag Berlin Heidelberg, 2008
8. Klusch, M., Fries, B., Sycara, K.: Automated Semantic Web Service Discovery with OWLS-MX. Proceedings of 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Hakodate, Japan, ACM Press, 2006
9. Klusch, M.: Semantic Web Service Coordination. In: M. Schumacher, H. Helin, H. Schuldt (Eds.) CASCOM - Intelligent Service Coordination in the Semantic Web. Chapter 4. Birkhuser Verlag, Springer, 2008
10. Kourtesis, D., Paraskakis I.: Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery. Proceedings of the 5th European Semantic Web Conference (ESWC 2008), Lecture Notes in Computer Science (LNCS), vol. 5021, Springer-Verlag Berlin Heidelberg, pp. 614628, 2008
11. Li, K., Verma, K., Mulye, R., Rabbani, R., Miller, J. A., Sheth, A. P.: Designing Semantic Web Processes: The WSDL-S Approach. Chapter submitted to Semantic Web Processes and Their Applications. J. Cardoso, A. Sheth, Editors. Springer
12. Meilecke, C., Stuckenschmidt, H.: Applying Logical Constraints to Ontology Matching. Proceedings of KI 2007: Advances in Artificial Intelligence: 30th Annual German Conference on AI, 2007
13. Shvaiko, P., Euzenat, J.: A Survey of Schema-based Matching Approaches Journal on Data Semantics, 2005.
14. Sycara, K., Paolucci, M., Ankolekar, A., Srinivasan, N.: Automated discovery, interaction and composition of Semantic Web services. Journal of Web Semantics, vol 1, Elsevier, 2003
15. Toch, E., Gal, A., Reinhartz-Berger, I., Dori D.: A Semantic Approach to Approximate Service Retrieval. ACM Transactions on Internet Technology, 8(1), 2008
16. TREC. Text Retrieval Conference. http://trec.nist.gov/data/.
17. Wang, T. D., Parsia, B., Hendler, J.: A survey of the web ontology landscape. Proceedings of International Semantic Web Conference (ISWC), 2006
18. Klusch, M., Gerber, A., Schmidt, M.: Semantic Web Service Composition Planning with OWLS-Xplan. 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web, Arlington VA, USA, 2005
19. Sirin, E., Parsia, B., Wu, D., Hendler, J., Nau, D.: HTN planning for web service composition using SHOP2. Proceedings of the 2nd International Semantic Web Conference (ISWC), pages 20-23, Sanibel Island, Florida, USA, 2003
20. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press 2003, ISBN 0-521-78176-0

[17] http://wordnet.princeton.edu/