Evaluating Semantic Web Service Tools using the SEALS platform

Liliana Cabral[1], Ioan Toma[2]

[1] Knowledge Media Institute, The Open University, Milton Keynes, UK
[2] STI Innsbruck, University of Innsbruck, Austria

l.s.cabral@open.ac.uk, ioan.toma@sti2.at

Abstract. This paper describes the approach for the automatic evaluation of Semantic Web Service (SWS) tools, based on the infrastructure under development within the SEALS project. We describe the design of evaluations, considering existing test suites as well as repository management and evaluation measure services that will enable evaluation campaign organizers and participants to evaluate SWS tools. Currently, we focus on the SWS discovery activity, which consists of finding Web Services based on their semantic descriptions. Tools for SWS discovery or matchmaking can be evaluated on retrieval performance, where for a given goal, i.e. a semantic description of a service request, and a given set of service descriptions, i.e. semantic descriptions of service offers, the tool returns the match degree between the goal and each service, and the platform measures the rate of matching correctness based on a number of metrics.

Keywords: Semantic Web Services, automatic evaluation.

1 Introduction

The evaluation of Semantic Web Services is currently being pursued by a few initiatives using different evaluation methods. Although these initiatives have succeeded in creating an initial evaluation community in this area, they have been hindered by the difficulties in creating large-scale test suites and by the complexity of manual testing to be done. In principle, it is very important to create test datasets where semantics play a major role for solving problem scenarios; otherwise comparison with non-semantic systems will not be significant, and in general it will be very difficult to measure tools or approaches based purely on the value of semantics. Therefore, providing an infrastructure for the evaluation of SWS that supports the creation and sharing of evaluation artifacts and services, making them widely available and registered according to problem scenarios, using agreed terminology, can benefit evaluation participants and organizers.

In this paper we describe the approach for the automatic evaluation of Semantic Web Services using the SEALS platform, that is, the services of the SEALS platform

for SWS tools. The SEALS (Semantic Evaluation at Large Scale) project[1] aims to create a lasting reference infrastructure for semantic technology evaluation (the SEALS platform). The SEALS Platform will be an independent, open, scalable, extensible and sustainable infrastructure that will allow online evaluation of semantic technologies by providing an integrated set of evaluation services and a large collection of datasets. Semantic Web Services are one of the technologies which are supported by SEALS. By the time of this writing, the SEALS project completed 15 months of its 36-month duration. Hence, the results presented in this paper are ongoing and under testing. An overview of existing SWS approaches, matchmaking algorithms or evaluation measures is not in the scope of this paper.

Within SEALS, we propose an approach that is informed by and improves existing SWS tool evaluation initiatives (Section 2.1). In this sense, our approach shares the goals and objectives of these initiatives. We describe the design of evaluations (Section 3), considering existing test suites (Section 2.2) as well as repository management and evaluation measure services (Section 4) that will enable evaluation campaign organizers and participants to evaluate SWS tools. Currently, we focus on the SWS discovery activity, which consists of finding Web Services based on their semantic descriptions. Tools for SWS discovery or matchmaking can be evaluated on retrieval performance, where for a given goal, i.e. a semantic description of a service request, and a given set of service descriptions, i.e. semantic descriptions of service offers, the tool returns the match degree between the goal and each service, and the platform measures the rate of matching correctness based on a number of metrics.

The evaluation of SWS tools uses metadata, described via ontologies, about the evaluation scenario, tools, testdata and results stored in repositories. The evaluation scenario metadata informs which test suites and tools participate in a specific evaluation event, and provides the evaluation workflow. The testdata metadata (Section 3.1) informs how the testdata is structured for consumption. More specifically, the metadata for SWS discovery test suites describes the set of service descriptions, the list of goals and the reference sets (expert's relevance values between a goal and a service). In addition, the evaluation of SWS tools produces two types of results: raw results, which are generated by running a tool with specific testdata; and interpretations, which are the results obtained after the evaluation measures are applied over the raw results. The format of the results (Section 3.1) is also described via ontologies.

In Section 5, we describe the interface of the SWS plugin API, which must be implemented for participating tools. In Section 6 we describe the evaluation workflow as well as examples of tools and available test suites for SWS discovery. The workflow performs access to the SEALS repositories and services. In the first phase of the project this workflow will be available to the evaluation campaign organizers only and executed over tools registered in a specific SWS evaluation campaign scenario. Finally in Section 7 we describe our conclusions and future work.

[1] http://www.seals-project.eu/

2 Related Work

Semantic Web Service (SWS) technologies enable the automation of discovery, selection, composition, mediation and execution of Web Services by means of semantic descriptions of their interfaces, capabilities and non-functional properties. SWS build on Web service standards such as WSDL, SOAP and REST (HTTP), and as such provide a layer of semantics for service interoperability. Current results of SWS research and industry efforts include a number of reference service ontologies (e.g. OWL-S, WSMO, WSMO-Lite) and semantic annotation extension mechanisms (e.g. SAWSDL, SA-REST, MicroWSMO).

The work performed in SEALS regarding SWS tools will be based upon the Semantic Web Service standardization effort that is currently ongoing within the OASIS Semantic Execution Environment Technical Committee (SEE-TC)[2]. A Semantic Execution Environment (SEE) is made up of a collection of components that are at the core of a Semantic Service Oriented Architecture (SOA). These components provide the means for automating many of the activities associated with the use of Web Services, thus they will form the basis for creating the SWS plugin APIs and services for SWS tools evaluation.

2.1 Existing SWS Evaluation Initiatives

In the following we provide information, extracted from the respective websites, about three current SWS evaluation initiatives: the SWS Challenge; the S3 Contest; and the WS Challenge (WSC).

The SWS Challenge[3] (SWSC) aims at providing a forum for discussion of SWS approaches based on a common application base. The approach is to provide a set of problems that participants solve in a series of workshops. In each workshop, participants self-select which scenario (e.g. discovery, mediation or invocation) and problems they would like to solve. Solutions to the scenarios provided by the participants are manually verified by the Challenge organising committee. The evaluation is based on the level of effort of the software engineering technique. That is, given that a certain tool can solve correctly a problem scenario, the tool is certified on the basis of being able to solve different levels of the problem space. In each level, different inputs are given that requires a change in the provided semantics. A report on the methodology for the SWSC has been published in the W3C SWS Testbed Incubator[4]. One of the important goals of the SWSC is to develop a common understanding of the various technologies evaluated in the workshops. So far, the approaches range from conventional programming techniques with purely implicit semantics, to software engineering techniques for modelling the domain in order to more easily develop application, to partial use of restricted logics, to full semantics annotation of the web services.

[2] www.oasis-open.org/committees/ex-semantics
[3] http://sws-challenge.org
[4] http://www.w3.org/2005/Incubator/swsc/XGR-SWSC-20080331

The Semantic Service Selection (S3) contest[5] is about the retrieval performance evaluation of matchmakers for Semantic Web Services. S3 is a virtual and independent contest, which runs annually since 2007. It provides the means and a forum for the joint and comparative evaluation of publicly available Semantic Web service matchmakers over given public test collections. S3 features three tracks: OWL-S matchmaker evaluation (over OWLS-TC); SAWSDL matchmaker evaluation (over SAWSDL-TC); cross evaluation (using JGD[6] collection). The participation in the S3 contest consists of: a) implementing the SME2[7] plug-in API for the participant's matchmaker together with an XML file specifying additional information about the matchmaker; and b) using the SME2 evaluation tool for testing the retrieval performance of the participant's matchmaker over a given test collection. This tool has a number of metrics available and provides comparison results in graphical format. The presentation and open discussion of the results with the participants is performed by someone from the organisational board at some event like the SMR2 workshop (Service Matchmaking and Resource Retrieval in the Semantic Web).

The Web Service Challenge[8] (WSC) runs annually since 2005 and provides a platform for researchers in the area of web service composition that allows them to compare their systems and exchange experiences. Starting from the 2008 competition, the data formats and the contest data are based on the OWL for ontologies, WSDL for services, and WSBPEL for service orchestrations. In 2009, services were annotated with non-functional properties. The Quality of Service of a Web Service is expressed by values expressing its response time and throughput. The WSC awards the most efficient system and also the best architectural solution. The contestants should find the composition with the least response time and the highest possible throughput. WSC uses the OWL format, but semantic evaluation is strictly limited to taxonomies consisting of sub and super class relationship between semantic concepts only. Semantic individuals are used to annotate input and output parameters of services. Four challenge sets are provided and each composition system can achieve up to 18 points and no less than 0 points per challenge set. Three challenge sets will have at least one feasible solution and one challenge set will have no solution at all.

2.2 Existing SWS Test Collections

The OWL-S Test Collection (OWLS-TC)[9] is intended to be used for evaluation of OWL-S matchmaking algorithms. OWLS-TC is used worldwide (it is among the top-10 download favourites of semwebcentral.org) and the de-facto standard test collection so far. It has been initially developed at DFKI, Germany, but later corrected and extended with the contribution of many people from a number of other institutions (including e.g. universities of Jena, Stanford and Shanghai, and FORTH). The OWLS-TC4 version consists of 1083 semantic web services described with

[5] http://www-ags.dfki.uni-sb.de/~klusch/s3/index.html
[6] http://fusion.cs.uni-jena.de/professur/jgd
[7] http://www.semwebcentral.org/projects/sme2/
[8] http://ws-challenge.georgetown.edu/wsc09/technical_details.html
[9] http://projects.semwebcentral.org/projects/owls-tc/

OWL-S 1.1, covering nine application domains (education, medical care, food, travel, communication, economy, weapons, geography and simulation). OWLS-TC4 provides 42 test queries associated with binary as well as graded relevance sets. The relevance sets were created with the SWSRAT (Semantic Web Service Relevance Assessment Tool) developed at DFKI. The graded relevance is based on a scale using 4 values: *highly relevant* (value: 3); *relevant* (value: 2); *potentially relevant* (value: 1); and *non-relevant* (value: 0). 160 services and 18 queries contain Precondition and/or Effect as part of their service descriptions.

The SAWSDL Test Collection (SAWSDL-TC) is a counterpart of OWLS-TC, that is, it has been semi-automatically derived from OWLS-TC. SAWSDL-TC is intended to support the evaluation of the performance of SAWSDL service matchmaking algorithms. The SAWSDL-TC3 version provides 1080 semantic Web services written in SAWSDL (for WSDL 1.1) and 42 test queries with associated relevance sets. Model references point to concepts described in OWL2-DL exclusively.

The Jena Geography Dataset (JGD)[10] is a test collection of about 200 geography services that have been gathered from web sites like seekda.com, xmethods.com, webservicelist.com, programmableweb.com, and geonames.org. JGD is available via the OPOSSum Portal[11]. The services are described using natural language. In addtition, the input and output parameter types have been manually linked to WordNet sense keys. The portal can also store ontologies to which service descriptions can refer. It is worth noting that the JGD collection has been used to support evaluations across formalisms. Semantic descriptions (including OWL-S and SAWSDL) for subsets of JGD have been created for evaluations within the context of the JGD cross-evaluation track at the S3 Contest.

3 SWS Tools Evaluation Design

Following on the SEALS infrastructure, SWS evaluation descriptions, test data, tool and evaluation results (including metadata) will be stored in respective repositories and used by the SEALS platform. The SEALS platform will basically run evaluations registered in the Evaluation Descriptions Repository. An Evaluation Description refers to the test data and tools participating in a specific evaluation scenario and provides the evaluation workflow for this scenario. Evaluation measures will be available as services, which can be used within evaluation workflows. The SEALS platform will also execute the SWS tool plugin, which must be implemented by tool providers (evaluation campaign participants).

We can summarize the goals of SWS tool evaluation as below:

- Provide a platform for the joint and comparative evaluation of publicly available Semantic Web service tools over public test collections.
- Provide a forum for discussion of SWS approaches.
- Provide a common understanding of the various SWS technologies.
- Award tools as a result of solving common problems.

[10] http://fusion.cs.uni-jena.de/professur/jgd
[11] http://fusion.cs.uni-jena.de/OPOSSum/

- Improve programmer productivity and system effectiveness by making semantics declarative and machine-readable.

We are interested in evaluating performance and scalability as well as solution correctness of application problems. We comment below on how we consider several evaluation criteria for SWS tools.

- Performance - This is specific to the type of SWS activity. For retrieval performance in discovery activities (i.e. service matchmaking), measures such as Precision and Recall are usually used. More generic performance measures are execution time and throughput.
- Scalability - Scalability of SWS tools are associated with the ability to perform an activity (e.g. discovery) involving an increasing amount of service descriptions. This can be measured together with performance (above), however, this is also related to the scalability of repositories.
- Correctness - This is related to the ability of a tool to respond correctly to different inputs or changes in the application problem by changing the semantic descriptions. This criterion is related to mediation and invocation of SWS. Messages resulting from the invocation or interaction of services should be checked against a reference set.
- Conformance - We are not concerned with measuring the conformance of a tool to a predefined standard. Instead, we will use a reference SWS architecture in order to define a SWS plugin API and a measurement API.
- Interoperability - As we are interested in evaluating SWS usage activities instead of the interchange of SWS descriptions, we are not concerned with measuring interoperability between tools.
- Usability - Although it might be useful to know which SWS tools have an easy to-use user interface or development environment, we consider that at this point in time due to the few number of front-ends for SWS development, a comparison would be more easily done using feedback forms. Therefore, we will not be concerned with measuring usability of SWS tools.

3.1 Metadata

In SEALS, artifacts such as test suites and results are stored in the SEALS Repositories. Every artifact is described using metadata in RDF/OWL. Generic metadata such as *artifact version*, *name* and *description* are associated with every artifact, however the metadata can be specialized for different types of tools. In particular, in this Section we describe the ontologies used to represent test suites and results for SWS discovery (see also [2]). The ontologies will be made publicly available at http://www.seals-project.eu/ontologies/.

The terminology used to describe a SWS discovery test data suite is provided by the *DiscoveryTestSuite* ontology, as graphically represented in Figure 1. The *DiscoveryTestSuite* ontology extends the generic *TestSuite* ontology defined in [3]. The main class *DiscoveryTestSuite* represents a discovery test suite and is a subclass of the *TestSuite* class. *DiscoveryTestSuite* can be described by properties of *TestSuite,* such as the *hasTestSuiteVersion*, modelled here using the *DiscoveryTestSuiteVersion*

class. The *DiscoveryTest* class represents a discovery test and is a subclass of the *Test* class. The property *belongsToDiscoveryTSV* indicates the discovery test suite version to which the discovery test belongs. The *MatchTest* class provides the goals and services for a match test using the properties *usesGoalDocument* and *usesService Document*. A discovery test includes a set of match tests. The property *belongsTo DiscoveryTest* indicates the discovery test suite to which the match test belongs. The *ServiceDocument* class represents a service description document, described by the properties *hasServiceName* (name of the service), *hasRepresentationLanguage* (language in which the service description is represented) and *isLocatedAt* (URI of the document). The *GoalDocument* class represents a goal document, which can be described by similar properties.

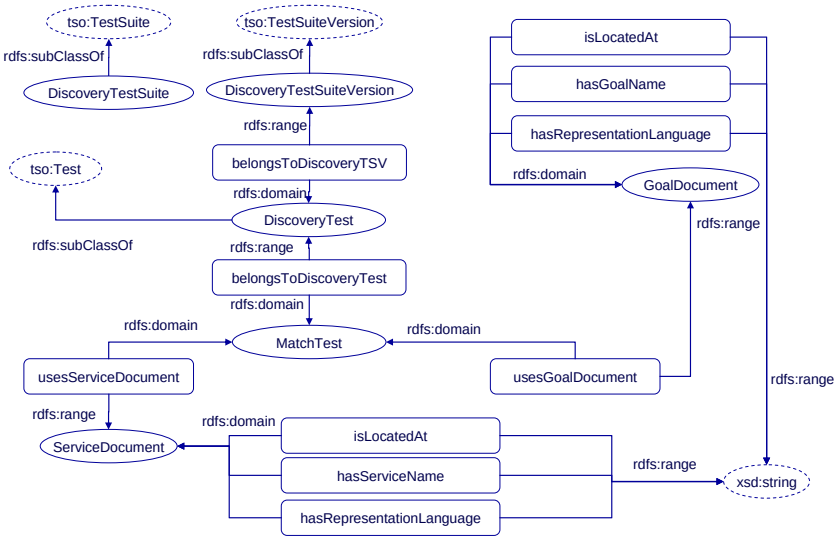

Figure 1 Graphical Representation of the Discovery TestSuite ontology

To describe a reference test suite we defined the *DiscoveryReferenceTestSuite* ontology (not shown here), which corresponds and extends the ontology in Figure 1 by adding the property *hasRelevance Value* to the *ServiceDocument* class.

In SEALS, the evaluation of SWS tools produces two types of results: *raw results*, which are generated by running a tool with specific testdata; and *interpretations*, which are the results obtained after the evaluation measures are applied over the raw results. In particular, for the evaluation of SWS discovery, raw results are represented according to the *DiscoveryResults* ontology as shown in Figure 2. A discovery result contains data produced by checking which services match a goal. The main class *DiscoveryResult* is a subclass of *TestRawResult* and together they specify to which discovery test suite and tool the match result belongs. In addition, they indicate whether any problems occurred. The *DiscoveryResultData* class is a subclass of *TestRawResultData* and represents the match result data that is described by the properties *hasGoalDescriptionURI, hasServiceDescription URI*, *hasMatchDegree* (the match degree, e.g. *None*, *Plugin*, *Exact*, *Subsumption*) and *hasConfidence*.

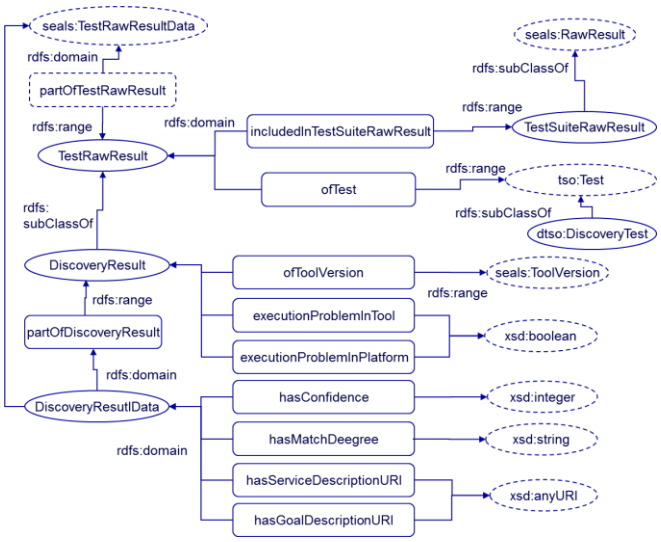

Figure 2 Graphical representation of Discovery Results ontology

Interpretations that are produced by interpreting the raw results produced by Semantic Web Services discovery tools are represented according to *Discovery Interpretation* ontology, as shown in Figure 3.

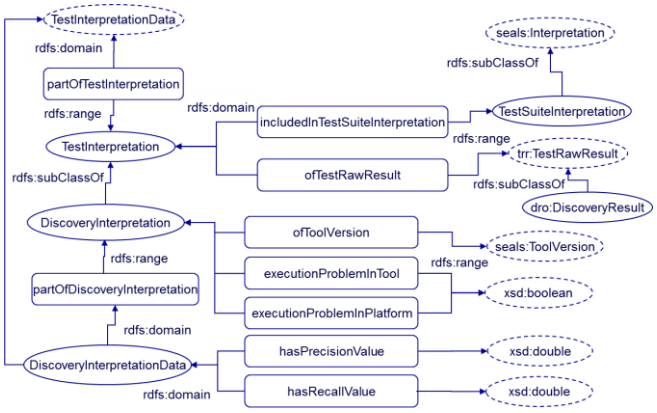

Figure 3 Graphical representation of the Discovery Interpretation ontology

This ontology has a format very similar to the Discovery Result ontology. The important difference is that the *DiscoveryInterpretation* class refers to the *DiscoveryResult* that generated this interpretation results. The *Discovery InterpretationData* class represents the discovery measurements from given a goal and a set of service descriptions. Currently, it is described by the properties *hasPrecisionValue* and *hasRecallValue*, which are decimal values representing the result of the respective measurements.

4 SWS Evaluation Services

In this section we will describe some of the services available from the SEALS platform for the evaluation of SWS tools. Currently, these services have been implemented as Java APIs, but in the future they will be available as Web Services. These services will be made publicly available at http://www.seals-project.eu/services/.

First, in SEALS we use repositories to store and retrieve test data, tools and results of an evaluation, namely the Test Data Repository, the Tools Repository and the Results Repository. Dedicated services called repository managers handle the interaction with the repositories and process metadata and data defined for SWS tool evaluation. More generic services (e.g. *retrieveTestDataSet*, *registerRawResult, registerInterpretation*) are used to access or store files (RDF or ZIP files) using REST clients; and more specific services (e.g. *extractGoals*, *extractServiceDescriptons*) are used to extract metadata content.

Second, we have developed a number of services in order to compute measurements for SWS discovery. Evaluation measures for SWS discovery will follow in general on the same principles and techniques from the more established Information Retrieval (IR) evaluation research area. Therefore we will use some common terminology and refer to common measures (as a reference see [4]). In the Java API, *DiscoveryMeasurement* is the main class, which returns metrics results for a given Discovery Result and Reference Set corresponding to the same goal. This class also returns overall measures for a list of goals. The class *Discovery Result* is part of the SWS plugin API (Section 5). The class *DiscoveryReferenceSet* contains the list of service judgments (class *DiscoveryJudgement*) for a specific goal, which includes the service description URI, and the relevance value. The relevance value is measured against the match degree returned by a tool. The class *MetricsResult* will contain a list of computed measure values such as *precision* and *recall* and also some intermediate results such as the as number of returned relevant services for a goal.

5 SWS Plugin API

In SEALS we provide the SWS plugin API, which must be implemented by tool providers participating in the SWS tool evaluation. As mentioned in Section 2, the SWS Plugin API (available from the campaign website) has been derived from the SEE API (see also [1] [2]) and works as a wrapper for SWS tools, providing a common interface for evaluation.

The *Discovery* Interface has 3 methods (*init(), loadServices(), discover()*) and defines a class for returning discovery results. The methods are called in different steps of the evaluation workflow (Section 6.1). The method *init()* is called once after the tool is deployed so that the tool can be initialized. The method *loadServices()* is called once for every dataset during the evaluation (loop) so that the list of services given as arguments can be loaded. The method *discover()* is called once for every goal in the dataset during the evaluation (loop) so that the tool can find the set of services that match the goal given as argument. The return type is defined by the class

DiscoveryResult. The class *DiscoveryResult* contains the goal and the list of service matches (class *Match*). The class *Match* contains the service description URI, the order (rank), the match degree ('NONE', 'EXACT', 'PLUGIN', 'SUBSUMPTION') and the confidence value. It is expected that the services that do not match are returned with match degree 'NONE' (assumed value is 0). 'EXACT' assumed value is 1.0. 'PLUGIN' and 'SUBSUMPTION' assumed value is 0.25.

6 SWS Discovery Evaluation Scenario

In the first SEALS evaluation campaign we will run the SWS discovery evaluation scenario[12]. Basically, a participant will register a tool via the Web interface provided in the SEALS website[13] and then he will be able to upload and edit his tool as part of an evaluation campaign scenario. Participants are also required to implement the SWS Tool plugin API as presented in Section 5. The organizers will make available instructions for the participants about the scenario and will perform the evaluation automatically by executing the workflow provided in the next section. The results will be available in the Results repository.

6.1 SWS Discovery Evaluation Workflow

In this section we describe the evaluation workflow for SWS discovery. The workflow performs access to the SEALS repositories in order to obtain the appropriate artifacts as well as access to available services for testing tools and applying measures. Also, the artifacts retrieved from the repositories can be metadata from which we extract the appropriate information.

The overall basic steps in the workflow are: find the evaluation description of a specific campaign scenario; extract the information about the tools and datasets from this description; then in a loop, execute each tool with the provided dataset; compute metrics (e.g. precision and recall) based on the provided reference set and raw results obtained, and finally the store both raw results and interpretations. In the following we describe in more details the service operations in the workflow. The high-level fragment of the actual java code implementation can be found in [2].

The *retrieveEvaluationDescription* operation accesses the Evaluation Repository and retrieves the discovery evaluation description corresponding to a given SWS Discovery evaluation campaign scenario. The *extractTools* operation extracts from the evaluation description metadata, the list of tools (ids) to be evaluated. We iterate over this list of tools, first checking whether the tool is deployed. The *extractTestDatasets* operation extracts from the evaluation description metadata, the list of datasets (ids) to be used in the evaluation. We iterate over this list of URIs, first retrieving each dataset from the testdata repository in operation *retrieveTestDataSet*. The *extractServices* operation extracts from the retrieved dataset, the list of service descriptions (URIs) to be used in the evaluation; and the extractGoals operation

[12] http://www.seals-project.eu/seals-evaluation-campaigns/semantic-web-services
[13] http://www.seals-project.eu/registertool

extracts the list of goals (URIs). The *runTool* operation runs the current tool with the current goal and services from the retrieved dataset. This operation will invoke the operations *loadServices* and *discover* from the SWS plugin implemented by the tool. The content of *DiscoveryResult* is serialized into the raw result for the current goal. This raw result is added to the list of raw results for the current dataset of the current tool in operation *addItemToRawResult*. The *extractReferenceSet* operation extracts from the retrieved dataset, the reference set to be used in the evaluation. The *computeMeasurements* operation computes all measurements (e.g. precision, recall) for the current goal using the current raw result and reference set. The content of *MetricsResult* is serialized into the interpretation for the current raw result. This interpretation is added to the list of interpretations for the current dataset of the current tool in operation *addItemToInterpretation*. The *registerRawResult* operation registers the accumulated list of raw results for the current dataset and tool into the results. The *registerInterpretation* operation registers the accumulated list of interpretations for the current dataset and tool into the results repository.

6.2 Tools and Test datasets

In Table 1 we list a number of tools that are publicly available and are candidates for evaluation using the SEALS platform under a SWS Discovery evaluation campaign.

Table 1 Candidate tools for SEALS SWS Discovery evaluation campaign

Tool Name	Provider	Webpage
OWLS-MX (with variants)	DFKI, Germany	http://projects.semwebcentral.org/projects /owls-mx/
SAWSDL-MX (with variants)	DFKI, Germany	http://projects.semwebcentral.org/projects /sawsdl-mx/
Glue2	CEFRIEL, Italy	http://sourceforge.net/projects/ glue2
IRS-III (Discovery)	KMI, The Open University, UK	http://kmi.open.ac.uk/ technologies/irs
WSMX (Discovery)	STI Innsbruck, Austria	http://www.wsmx.org/

We have registered two test suites to the Test Data Repository[14]. The latest collections corresponding to OWLS-TC and SAWSDL-TC are accessible at http:// seals.sti2.at/tdrs-web/testdata/persistent/OWLS-TC/4.0 and http://seals.sti2.at/tdrs-web/testdata/persistent/SAWSDL-TC/3.0 respectively. Depending on the value of the *HTTP Accept* for the two URLs above, either the metadata or the data is retrieved. To retrieve the metadata of the test suite version, set the *HTTP Accept* value to "application/rdf+xml". To retrieve the actual data as a ZIP file, set the *HTTP Header* value to "application/zip".

[14] http://seals.sti2.at/tdrs-web/

7 Conclusions

In this paper we have described the ongoing approach and services for SWS tools evaluation using the SEALS platform. We have described the implementation of the SWS Discovery Evaluation Workflow. As part of the workflow, we have implemented services (java code) for accessing the testdata and result repositories as well as services (java API) for performing measures for SWS Discovery evaluation. We have created metadata definitions (RDF/OWL) for testdata and results (raw results and interpretations) and corresponding services (generating and adding ontology instances to repositories). The SWS plugin API is currently very similar to SME's matchmaker plugin in what concerns discovery (matchmaking). However, the former will be extended in order to account for other activities such as composition, mediation and invocation. There are also many similarities in purpose between our approach and existing initiatives in that they all intend to provide evaluation services and promote discussion on SWS technologies within the SWS community. In this case, the main difference is that SEALS is investigating the creation of common and sharable metadata specifications for testdata, tools and evaluation descriptions as well as respective public repositories.

The work presented in this paper will be used during the SWS Tools Evaluation Campaign 2010, available at http://www.seals-project.eu/seals-evaluation-campaigns/semantic-web-services. This campaign will run the *SEALS Semantic Web Service Discovery Evaluation* scenario**.** Instructions for participants will be made available. In addition, we will release the SWS plugin API, which must be implemented for the participating tools. Currently, the last versions of the existing data collections OWL-S TC and SAWSDL-TC have been stored in the dataset repository. For future campaigns we plan to release datasets described using other languages such as WSMO-Lite as well as datasets for other problem scenarios such as the ones in the SWS Challenge. For the first campaign, the uploading of participant tools will be manual and there will be no access to the evaluation repository. The SWS evaluation workflow mentioned before will be performed by the organizers over the participating tools. Future work includes developing the APIs as Web Services and implementing the evaluation workflow as a BPEL process.

Acknowledgments This work has been partially funded by the European Commission under the SEALS project (FP7-238975).

References

1. Cabral, L., Kerrigan, M. and Norton, B.: D14.1. Evaluation Design and Collection of Test Data for Semantic Web Service Tools. Technical report, SEALS Project, March 2010.
2. Cabral, L., Toma, I., Marte, A.: D14.2. Services for the automatic evaluation of Semantic Web Service Tools v1. Technical report, SEALS Project, July 2010.
3. Garcia-Castro, R., Esteban-Gutierrez, M., Nixon, L., Kerrigan, M. and Grimm, S.: D4.2. SEALS Metadata. Technical report, SEALS Project, Feb 2010.
4. Küster, U., Koenig-Ries, B.: Measures for Benchmarking Semantic Web Service Matchmaking Correctness. In Proceedings of ESWC 2010. LNCS 6089. June, 2010.