Infrastructure and Workflow for the Formal Evaluation of Semantic Search Technologies

Stuart N. Wrigley
Dept. Computer Science
University of Sheffield, UK
s.wrigley@dcs.shef.ac.uk

Raúl García-Castro
Facultad de Informática
Universidad Politécnica de Madrid, Spain
rgarcia@fi.upm.es

Cássia Trojahn
INRIA & LIG
Montbonnot Saint Martin, France
cassia.trojahn@inrialpes.fr

ABSTRACT

This paper describes an infrastructure for the automated evaluation of semantic technologies and, in particular, semantic search technologies. For this purpose, we present an evaluation framework which follows a service-oriented approach for evaluating semantic technologies and uses the Business Process Execution Language (BPEL) to define evaluation workflows that can be executed by process engines. This framework supports a variety of evaluations, from different semantic areas, including search, and is extendible to new evaluations. We show how BPEL addresses this diversity as well as how it is used to solve specific challenges such as heterogeneity, error handling and reuse.

Categories and Subject Descriptors

D.2 [**Software Engineering**]: Software/Program Verification; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Experimentation, Measurement, Performance

Keywords

Semantic Technology Evaluation, Semantic Search, Evaluation Infrastructure, Workflows, BPEL

1. INTRODUCTION

Semantic technologies play a critical role in the recent advances in both web services and corporate knowledge management. Such developments are revolutionising the way information and knowledge are processed. Semantic technologies provide ways to express knowledge and data so that they can be properly exploited by computers in an automated way for different purposes such as information retrieval or data integration. The evaluation of such technologies is critical for their sustained improvement and adoption, allowing users to assess the suitability of current technologies to their needs. Some initiatives have already created a basis for semantic technology evaluation, such as those in the areas of ontology matching [3], ontology engineering [5, 6], ontology reasoning [7, 11], semantic search [9] or semantic web services [10, 13]. However, additional effort is required to accommodate the growth of the field, since evaluation is still costly, both in terms of reusing evaluation resources defined by others and of actually executing evaluations and analysing their results.

Manual evaluation of software does not scale when one tool has to be evaluated several times, when multiple tools have to be evaluated in exactly the same way, or when multiple tools have to be evaluated in more than one way. To ensure repeatability and allow comparison of tool results, such evaluations ought to be automated (and indeed use identical compute resources). However, such automation is a complex task that requires: a) the coordinated interaction in an evaluation workflow of all the involved resources, e.g., tools, test data and interpreters; b) the definition of such evaluation workflows in some machine-processable format; and c) the ability to cope with the heterogeneity of the different evaluation resources. The work conducted by the Semantic Evaluation at Large Scale (SEALS) project[1] has developed a comprehensive solution – the SEALS Platform [4] – which encompasses all of these factors.

The SEALS Platform is an open infrastructure for the evaluation of semantic technologies that offers independent computational and data resources for the evaluation of those technologies. To this end, the SEALS Platform provides a common evaluation framework, based on the reusability of evaluation resources, in which different types of semantic technologies can be automatically evaluated. It is responsible for all aspects of the evaluation: test data management; tool configuration and execution; result and interpretation generation and storage, etc. In order to ensure reproducibility and allow direct performance comparison, the entire evaluation is conducted within the SEALS Platform. In other words, all test data is stored locally as are the tools to be evaluated. The tools themselves are executed *within* the SEALS Platform (using virtual machine approaches to handle operating system dependencies) and once one or more tools have been evaluated, the generated results and any subsequent analyses are also stored locally and are made available for visualisation.

A major implication of this self-contained approach is that the tool to be evaluated must be controllable programmatically – all its operations must be accessible to the SEALS

[1] www.seals-project.eu

DESIRE'11, October 28, 2011, Glasgow, Scotland, UK.

Platform without human intervention. Therefore, in order for any tool to be run, it must meet certain integration criteria regarding the capabilities that have to be exposed to the platform – it must implement a specific API. Such functionality includes means of deploying and undeploying a tool as well as functionalities specific to a particular type of tool (for instance, 'match' for an ontology matching tool or 'executeQuery' for a search tool). The API is specified by the evaluation organiser and all participating tools must implement this API (usually a 'wrapper' around an existing tool).

This design allows the SEALS framework to deal with a variety of heterogeneous evaluations, from different semantic areas, and is extendible to encompass new evaluations. Naturally, the steps necessary to evaluate an ontology matching tool will be very different from those to evaluate a semantic search tool. Therefore, we use the Business Process Execution Language (BPEL) [1] to provide an efficient means of scripting the entire lifecycle of a particular evaluation (a workflow). Again, the creation of this workflow is the responsibility of the evaluation organiser.

Indeed, the versatility of the platform was demonstrated during the first worldwide SEALS evaluation campaign held during mid-2010 in which tools from five different semantic technology fields (ontology engineering, semantic search, semantic web services, ontology matching, storage and reasoning) were formally evaluated using this infrastructure [12] (see [15] for details of the search evaluation and its results).

State-of-the-art semantic search approaches are characterised by their high level of diversity both in their features as well as their capabilities. Such approaches employ different styles for accepting the user query (e.g., forms, graphs, keywords) [14] and apply a range of different strategies during processing and execution of the queries. They also differ in the format and content of the results presented to the user. All of these factors influence the user's perception of performance and usability.

Semantic search technologies can be evaluated on the basis of different criteria and metrics [16, 9]. At the core of any search task is the retrieval of pertinent information; search evaluations employ several questions which are applied to a particular ontology and dataset. Since the answer set for each question is finite and known *a priori*, the measures of precision (true positive/retrieved) and recall (true positive/expected) are used. We are also interested in how tools cope with increasingly large datasets (scalability) which, in turn, affects tool efficiency with respect to CPU load and memory usage. Since search is an inherently user-oriented task, evaluation must also consider metrics such as how long it takes for a query to be executed. In this paper, we focus on a semantic search evaluation workflow (see Sec. 3).

Our approach deals with the inherent heterogeneity when considering technologies from areas as diverse as semantic search, ontology engineering and semantic web service discovery to name but three. We achieve this at two levels: at a *logical level*, where we have specified a common framework for the evaluation of semantic technologies that allows defining common structures and behaviours for the different evaluation entities; and at a *technical level*, where we abstract out technology heterogeneity by defining a Service Oriented Architecture that includes all the evaluation services needed for particular semantic technology evaluations.

The goal of this paper is to present an evaluation framework that follows a service-oriented approach for evaluating semantic search technologies and how BPEL is used to define evaluation workflows that can be processed by the platform. The paper is organised as follows. Section 2 gives an overview of the SEALS Platform and presents the process that takes place when executing evaluations. Section 3 presents an abstract evaluation workflow and then shows how this has been implemented for semantic search technology evaluations. Finally, Section 4 draws conclusions from the work presented in this paper and proposes future lines of work.

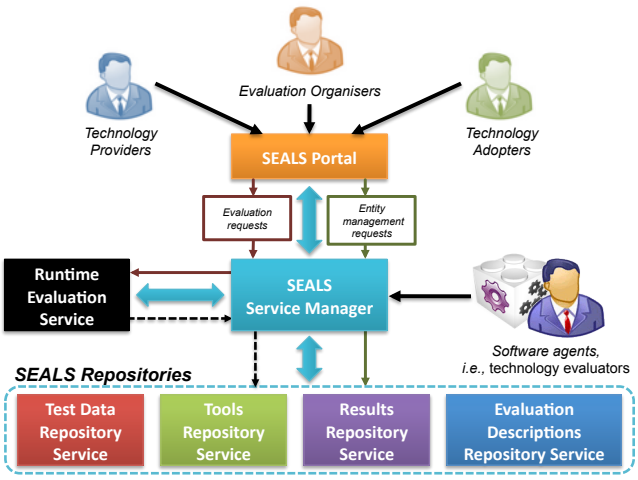

Figure 1: The SEALS evaluation platform.

2. EVALUATION EXECUTION PROCESS

The different entities that participate in an evaluation are the following: in any *evaluation* a given set of *tools* are evaluated, following a given *evaluation workflow* and using predetermined *test data* and evaluation *criteria*. As an outcome of this process, a set of *evaluation results* is produced. Evaluation results are classified[2] according to their provenance, differentiating *raw results* (those evaluation results directly generated by tools) from *interpreted results* (those generated from raw results when applying some evaluation criterion).

The SEALS Platform (Figure 1) has been developed around these evaluation entities; the service-oriented architecture of the platform comprises a number of components responsible for: coordination and consistency management (SEALS Service Manager), management of the platform entities (Test Data, Tools, Results, and Evaluation Descriptions Repositories), and evaluation execution (Runtime Evaluation Service).

Next, we describe the evaluation execution process that takes place in the SEALS Platform when a user requests the execution of an evaluation description (i.e., workflow) over a certain tool and using some test data. A detailed description of this process and of the management of computing resources and tools in the platform can be found in [2].

Figure 2 depicts the main interactions between the platform software components during an evaluation execution[3].

[2]The classification adopted in SEALS is in accordance with the approach followed in the IEEE 1061 standard for a software quality metrics methodology [8].

[3]We use BPMN, the standard notation for business processes, for illustrating the platform processes.

These interactions occur during three sequential stages that are described below.

2.1 Execution Request Analysis

The *SEALS Portal* is the entry point for users to launch evaluation executions. An evaluation execution request specifies the tool to be evaluated, the evaluation description to execute, and the test data to use as input. The SEALS Service Manager (SSM) is responsible for orchestrating the execution request. To this end, the Runtime Evaluation Service (RES) retrieves the evaluation description from the Evaluation Descriptions Repository (EDR), analyses the execution request in order to guarantee that it can be processed, and prepares all the information that is required for executing the rest of the evaluation process.

Among other things, the analysis includes checking the evaluation description (i.e., validating the syntax of the evaluation description and checking that the workflow described is well-defined) and checking if the execution request arguments satisfy the evaluation description contract (i.e., verifying the availability and type of the specified entities). If any of these verifications fails (syntax, semantics, or resources), the stage will fail and thus the processing of the execution request will terminate.

2.2 Execution Environment Preparation

Once the RES has checked that the execution request may be safely executed, the execution environment is prepared, that is, to prepare: a) the set of computing resources where the tools to be evaluated during the execution of the evaluation description are physically run and b) the evaluation workflow and any custom services required by it.

Tools available for the evaluation are stored in the Tools Repository (TR) and each tool may have its own computing requirements (i.e., operating system or third party applications). Therefore, in order to enable the reuse of the limited shared resources, the RES is in charge of preparing the computing resources according to the requirements of the tools under evaluation.

This preparation is carried out in two steps. First, the RES requests from the SSM the computing resources that are needed for executing the tools involved in the evaluation description. Then, once the computing resources have been acquired, the RES deploys the tools to be used during the execution of the evaluation description, as well as any third party application required by the tools.

More specifically, an evaluation description includes, besides the evaluation workflow, those evaluation-specific services required by the workflow; these services are different from those generic services provided by the platform, e.g., the repository services, and mainly deal with applying evaluation-specific metrics to evaluation raw results in order to obtain the corresponding interpretations.

The RES must deploy in the platform the evaluation workflow included in the evaluation description as well as those services that are invoked from it.

2.3 Evaluation Description Execution

At this stage the RES enacts the workflow defined in the evaluation description following the defined control flow and executing the activities specified within the workflow. The execution of these activities is composed of one or more

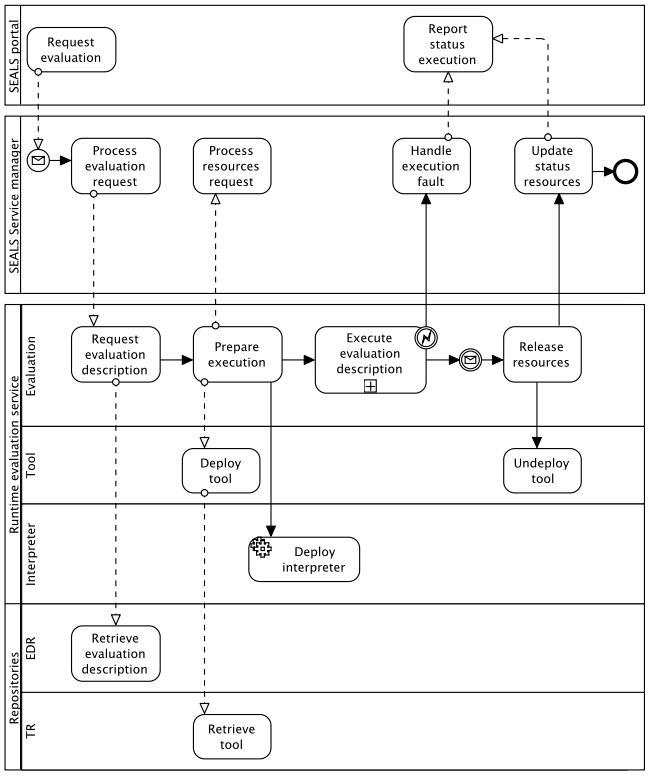

Figure 2: The evaluation execution process.

steps, depending on the specific activity and the current state of execution:

1. The first step is to stage-in all the data from the Test Data Repository (TDR) to be used in the activity; i.e., making the data involved available in the computing resource where the activity will be executed.

2. Once the data is available, it is time to execute the particular activity. The activity can imply invoking a tool's functionality or the interpretation of raw results by means of specific software artefacts, i.e., the interpreters.

3. Regardless of the specific activity executed, the next step consists of storing the results obtained in the Results Repository (RR). These results will be raw results if the activity executed was the invocation of a tool's functionality, and interpreted results otherwise.

4. Finally, any data that is not going to be further used in the computing resource should be deleted and thus the storage space freed. This final step will occur even if any previous step has failed to complete.

An important issue during the whole process is to handle eventual execution errors, either those that do not affect the normal flow of the process or those that are returned by services in response to a request that cannot be processed.

3. EVALUATION WORKFLOWS

In this section, we focus on evaluation descriptions and in particular their BPEL workflows. First we present a generic

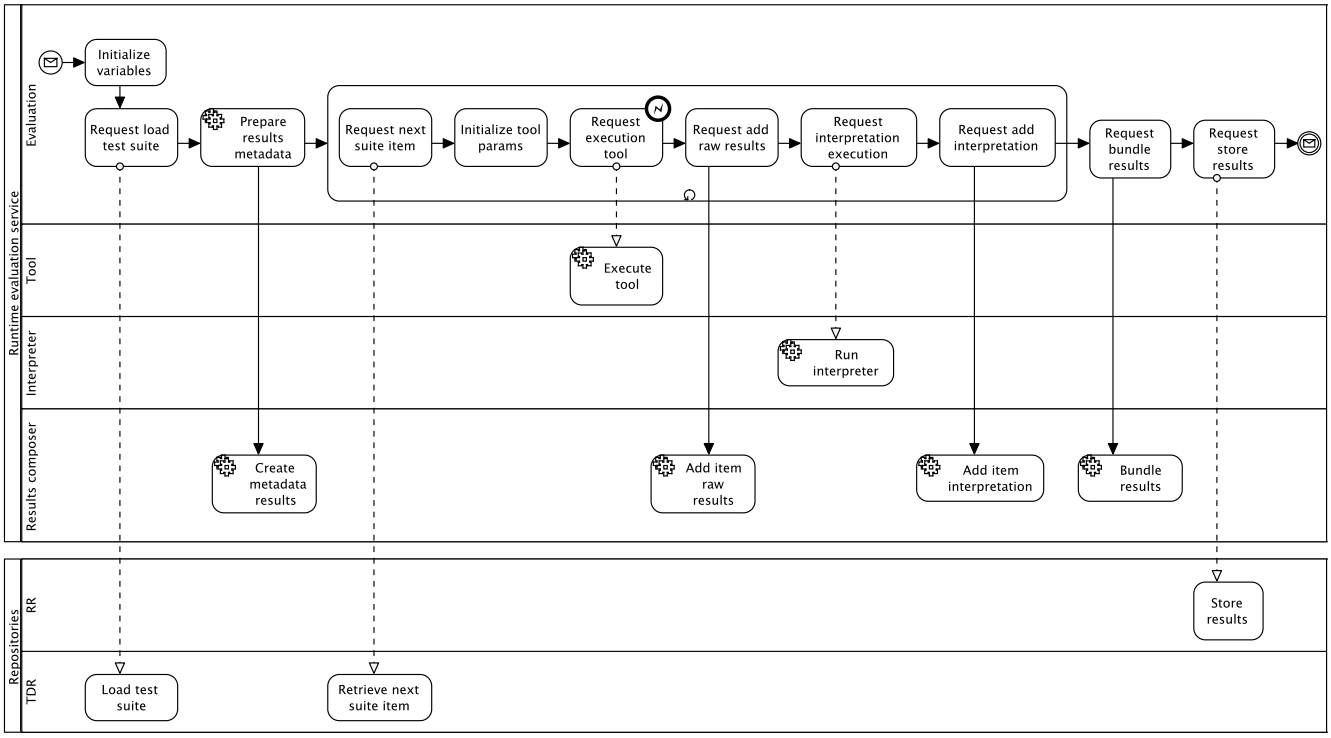

Figure 3: Generic evaluation workflow.

evaluation workflow and then show how it can be implemented for semantic search tool evaluations.

Figure 3 presents a generic evaluation workflow that corresponds to the *Execute evaluation description* process depicted in Figure 2. Any evaluation workflow requires interaction with the services provided by the Test Data and Results Repositories, as well as with those services that are specific to the evaluation (i.e., for accessing the deployed tool and interpreting raw results):

- The `test data repository service` provides access to the Test Data repository, in order to iterate over a test suite and retrieve its test items.
- The `tool service` provides access to a deployed tool.
- The `results repository service` provides access to the Results Repository, in order to add raw results and interpretations.
- The `results composer service` is a utility service that provides operations for bundling all the results generated during an evaluation execution (either raw results or interpretations).
- The `interpreter service` provides access to custom metrics implementations.

As Figure 3 shows, a typical evaluation workflow starts by receiving from the client process two input parameters: the identifier of the tool to be evaluated and the identifier of the test suite. Then, the execution environment is set up by loading the test suite from the Test Data Repository and by initialising the metadata that will describe the execution and its results. As presented in Figure 2, before the workflow execution, the Runtime Evaluation Service has already deployed the tool to be evaluated from the Tools Repository to a concrete computing resource.

The evaluation workflow iterates over all the test cases in the test suite and performs the following tasks in each iteration:

1. The *tool service* is invoked passing as parameters certain data items that are contained in the test case definition (which are particular to each evaluation).
2. The output of the tool invocation is the raw result that is sent to the *results composer service* to be bundled.
3. The *interpreter service* (which implements some evaluation metrics) derives from the raw result the interpretation of the test case execution and sends it to the *results composer service* to be bundled.

When all the test cases have been executed, the *results composer service* creates two bundles described with the corresponding metadata, one with the raw results of every test case and another with the interpretations, that are later stored in the Results Repository. Since the workflow is asynchronous, a callback notification message with a successful response is sent to the client consuming the workflow, when its execution finishes.

For semantic search evaluations, a test suite is composed of a single ontology, a set of questions and a corresponding set of groundtruths (answers). Within such a suite, a test case is a combination of the ontology, a question and that question's corresponding groundtruth.

The graphical representation of the semantic search BPEL workflow implementation is shown in Figure 4. A BPEL workflow allows interactions with one or more web services to be scripted; little computation is possible within the BPEL itself other than the execution flow control (conditional branching, repetition, etc). Individual web services that can be accessed via the BPEL are termed *partner links*. A request to execute a particular method associated with a partner link is termed an *invocation*. In most cases, an invocation has some parameters associated with it; therefore, immediately before any invocation in the BPEL workflow, there will be an `assign` step, in which the appropriate data is associated with the invocation's request parameter.

Since the RES only provides generic workflow execution, it is necessary to make additional services available in order to fulfil the semantic search tool evaluation requirements. Such services include timers for measuring how long particular stages of the processing take and a service for analysing the tool's query results in order to compute the precision, recall and f-measure metrics. Consequently, these services are implemented as custom web services (identified in Figure 4 as `timestamp:` and `analysis:`), which can be deployed on an ad hoc basis within the SEALS Platform at runtime.

On the SEALS Platform, a BPEL workflow is initiated with four parameters: two specifying the evaluation-specific configurations and two SEALS Platform-generated parameters. The latter are necessary for the platform-internal process tracking; the former specify the test data suite and its associated version to be used within the evaluation. As part of the platform-internal process tracking, the SEALS Platform parameters must be associated with all partner link invocation requests (this is achieved in the `setHeadersForInvocations` step).

The Result Composer Service provides a service for collating results when an evaluation is being executed. Such results are collated into a *bundle*; one further part of the bundle structure is the metadata for describing the result contents. This data is associated with the results bundle in the `rc:addMetadata` step.

The Test Data Repository service provides functionality for iterating over a test data suite's test cases. However, in order to do this, the Test Data Repository must first 'load' the test suite (the `tdrs:loadTestSuite` step). The `tdrs:hasNextTestCase` step ensures that the parameter for the while loop has been initiated.

Finally, before the core evaluation work starts (within `While more test cases`), the tool is instructed not to launch its GUI (since, if present, it is unnecessary).

Once the `While more test cases` loop has been entered, the next test case from the suite is loaded (`tdrs: nextTestCase`). Each test case has the same ontology associated with it. Therefore, it is only necessary for the tool to load it once. This is achieved by using a BPEL-local variable to determine whether the tool has loaded the ontology yet. If not, the URL of the ontology is retrieved from the Test Data Repository (`tdrs:getOntologyURL`) and the tool is asked to load it (`tool:loadOntology`). The amount of time taken to load the ontology is recorded.

Following this, the test question associated with this test case is retrieved (`tdrs:getQuestionURL`) and is passed on to the tool for the query execution phase (`tool:executeQuery`). It may take some time for the tool to complete the query, hence the BPEL workflow includes a while loop repeating

Figure 4: BPEL workflow for the evaluation of semantic search technologies. The right hand column is a continuation of the left hand column sequence.

until the results are ready (`tool:isResultSetReady`). Once this method returns true, the URL of the results file is obtained from the tool (`tool:getResults`) and associated with the results bundle (`rc:addResultToBundle`). Again, the time required for the query to be executed is recorded. In addition, these results are used to compute the precision, recall and f-measure associated with this query (`analysis:getPRF`). The final task of the while loop is to check once more whether there are any more test cases remaining (`tdrs: hasNextTestCase`).

Once the core evaluation work has been completed, there are some preprocessing steps required in order to finalise the result bundle before it's ready for being uploaded to the Results Repository. The metadata stored at the beginning of the BPEL workflow is retrieved (`rc:getMetadata`) and used to create the final result bundle ZIP file (`rc:createBundle`). This is then stored in the Results Repository (`rrs: storeRawResults` step).

The workflow is asynchronous, i.e., the client does not wait for the complete execution of the evaluation. For that reason a notification must be sent when the process finishes (`client:callbackClient`).

Note that this workflow makes use of two types of ser-

vice: ones common across all evaluations (e.g., services to access the repositories) and custom services which provide functionality specific to the evaluation of semantic search technologies (e.g., computation of precision, recall and f-measure).

4. CONCLUSIONS AND FUTURE WORK

This paper has presented an evaluation framework for semantic technologies (such as semantic search), implemented by the SEALS Platform. With the main goals of automation and reusability, all the evaluation entities used in the framework are described in machine-processable formats and based on standards. This has allowed us to define an automatic process for executing evaluation workflows in a service-oriented environment. The SEALS Platform has already been used to host a worldwide evaluation campaign targeting a range of semantic technologies [12] including semantic search tools [15].

We have also presented a BPEL workflow case study. Using BPEL to describe evaluation workflows significantly facilitates workflow reusability. Indeed, with the exception of the tool and interpreter services, the rest of the services are reused across all the evaluations. Furthermore, the use of a standardised approach such as BPEL facilitates the creation of new evaluation workflows by researchers who may be unfamiliar with the rest of the platform infrastructure.

Future work will include implementing validation functionalities that allow the correctness of the evaluation entities (evaluation workflows, tools, test data, and results) to be ensured before evaluation workflow execution. This would allow the detection of potential problems in advance, e.g., when test data are uploaded to the Test Data repository or during the definition of an evaluation workflow.

5. ACKNOWLEDGMENTS

The authors would like to thank Jérôme Euzenat, Liliana Cabral and Mikalai Yatskevich for their valuable comments on a previous version of this paper. The authors would also like to thank the other members of the SEALS consortium. The authors are partially supported by the SEALS EU FP7 project (IST-2009-238975).

6. REFERENCES

[1] A. Alves, A. Arkin, S. Askary, B. Bloch, F. Curbera, Y. Goland, N. Kartha, Sterling, D. König, V. Mehta, S. Thatte, D. van der Rijn, P. Yendluri, and A. Yiu. Web services business process execution language version 2.0. OASIS Standard Committee, April 2007.

[2] M. Esteban-Gutiérrez, R. García-Castro, and A. Gómez-Pérez. Executing evaluations over semantic technologies using the SEALS Platform. In *Proceedings of the International Workshop on Evaluation of Semantic Technologies (IWEST 2010)*.

[3] J. Euzenat *et al.* Results of the ontology alignment evaluation initiative 2010. In *Proc. of the 5th Workshop on Ontology Matching (OM-2010), collocated with ISWC-2010*, pages 85–117, Shanghai (CN), 2010.

[4] R. García-Castro, M. Esteban-Gutiérrez, and A. Gómez-Pérez. Towards an infrastructure for the evaluation of semantic technologies. In *Proceedings of the eChallenges 2010 Conference*, Warsaw, Poland, October 27-29 2010.

[5] R. García-Castro and A. Gómez-Pérez. RDF(S) interoperability results for semantic web technologies. *International Journal of Software Engineering and Knowledge Engineering*, 19(8):1083–1108, December 2009.

[6] R. García-Castro and A. Gómez-Pérez. Interoperability results for Semantic Web technologies using OWL as the interchange language. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8:278–291, November 2010.

[7] I. Horrocks and P. F. Patel-Schneider. DL systems comparison. In *Proc. of the 1998 Description Logic Workshop (DL'98)*, volume 11, pages 55–57, 1998.

[8] IEEE. *IEEE 1061-1998. IEEE Standard for a Software Quality Metrics Methodology*. IEEE, December 1998.

[9] E. Kaufmann. *Talking to the Semantic Web — Natural Language Query Interfaces for Casual End-Users*. PhD thesis, Faculty of Economics, Business Administration and Information Technology of the University of Zurich, September 2007.

[10] M. Klusch, A. Leger, D. Martin, M. Paolucci, A. Bernstein, and U. Kuester. Annual International Contest S3 on Semantic Service Selection. `http://www-ags.dfki.uni-sb.de/~klusch/s3/`.

[11] F. Massacci and F. M. Donini. Design and results of TANCS-2000 non-classical (modal) systems comparison. In *TABLEAUX '00: Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 52–56, London, UK, 2000. Springer-Verlag.

[12] L. Nixon, R. García-Castro, S. N. Wrigley, M. Yatskevich, C. T. D. Santos, and L. Cabral. The state of semantic technology today – overview of the first SEALS evaluation campaigns. In *I-SEMANTICS 2011 – 7th International Conference on Semantic Systems*, 2011.

[13] C. Petrie, T. Margaria, H. Lausen, and M. Zaremba. *Semantic Web Services Challenge: Results from the First Year*. Springer, 2009.

[14] V. Uren, Y. Lei, V. Lopez, H. Liu, E. Motta, and M. Giordanino. The usability of semantic search tools: a review. *The Knowledge Engineering Review*, 22(4):361–377, 2007.

[15] S. N. Wrigley, K. Elbedweihy, D. Reinhard, A. Bernstein, and F. Ciravegna. Evaluating semantic search tools using the SEALS Platform. In *Proceedings of the International Workshop on Evaluation of Semantic Technologies (IWEST 2010) Workshop, collocated at ISWC 2010*, 2010.

[16] S. N. Wrigley, D. Reinhard, K. Elbedweihy, A. Bernstein, and F. Ciravegna. Methodology and campaign design for the evaluation of semantic search tools. In *Proceedings of the International Workshop on Semantic Search, collocated at WWW 2010*, 2010.