Using WS-BPEL for Automation of Semantic Web Service Tools Evaluation

Serge Tymaniuk[1], Ioan Toma[1], Liliana Cabral[2]

[1] Semantic Technology Institute, Universität Innsbruck, Austria.
{serge.tymaniuk,ioan.toma}@sti2.at
[2] Knowledge Media Insitute, The Open University, Manchester, UK
l.s.cabral@open.ac.uk

Abstract. Although a significant number of semantic tools have been produced, there exists a shortage of approaches that allow tools benchmarking in an automatic way. This paper presents an approach for automation of Semantic Web Service (SWS) tools evaluation by means of WS-BPEL based Web services using the infrastructure provided by the SEALS project. One of the objectives of the SEALS project is to promote advancement of semantic technologies state of the art by enabling continuous mechanized evaluation. We describe the methodology, used for automating SWS discovery tools evaluation, present an orchestration solution for enabling automation of evaluation workflow and discuss implementation results and lessons learnt.

Keywords: Semantic Web service evaluation, automation, BPEL

1 Introduction

Although many initiatives that created Semantic Web Service (SWS) approaches and semantic service descriptions have been completed recently, there have been insufficient efforts in ensuring automatic evaluation of functional and non functional aspects of respective systems and tools. Indeed, the significance of software benchmarking should not be underestimated since the evaluation of efficiency, effectiveness and performance costs of SWS tools allows discovering bottlenecks, elaborating improvements and best-practices, which could provide feedback and be advantageous not only for the tool providers but also for the whole SWS community, and, in general, would facilitate further development of SWS field.

Furthermore, the evaluation process itself has to meet specific requirements in order to provide quality, time and performance assessment in an inexpensive, timely and effective way. Ideally, an automatic evaluation approach, in most of the cases, is preferred over the manual testing while doing tools benchmarking. However, the construction of complex automated evaluation systems often involves different process management concepts that can support underlying IT infrastructure and ease the design and implementation processes. In this case, the Business Process Management (BPM) concept can act as a linking approach

Proceedings of the Second International Workshop on
Evaluation of Semantic Technologies (IWEST 2012)
May 28th, Heraklion, Greece
CEUR Workshop Proceedings Vol. 843

that enables the modelling of an evaluation scenario at a higher level perspective and thus defining the means for its implementation. The BPM stack [1], introduced by the Business Process Management Initiative group[1], represents a hierarchical systematization of a set of technologies for supporting business process management, consisting of basic OASIS and W3C web services infrastructure technologies at the bottom such as *SOAP* (Simple Object Access Protocol)[2], *WSDL* (Web Service Definition Language)[3] and *UDDI* (Universal Description Discovery Integration)[4]; choreography and execution standards such as *WS-CDL* (Web Services Choreography Description Language)[5] and *WS-BPEL* (Web Services Business Process Execution Language)[6] at the middle layer; and business process extension layers (*BPEL4People*)[7] and graphical notation standards (*BPMN*)[8] at the top.

The goal of this paper is to describe an approach for the automation of SWS tools evaluation based on the BPM view above, using WS-BPEL based Web services and to discuss the implementation results and lessons learnt. The research, provided in this paper builds on our initial work described in [2], in which we have presented the data sets and tools that are candidates for the evaluation, the evaluation goals and criteria against which tools benchmarking should be conducted, and the tool wrapper, required for tool integration with the evaluation platform.

Our work on automation of SWS tools evaluation is part of the SEALS project[9], which aims at creating a lasting reference infrastructure for semantic technology evaluation. In SEALS we use WS-BPEL to represent an evaluation scenario process. In particular, in this paper we discuss the representation and execution of a WS-BPEL based process for evaluating SWS based discovery tools. The SWS Discovery evaluation BPEL process contains several Web Services, which enable access to testdata, execution of provider's tools and execution of measurements over tool's results, among other evaluation tasks.

This paper is organized as follows. In Section 2, an overview of the related work is provided. The automatic evaluation is implemented as a part of the SEALS platform. An overview of this platform, including basic terms used in the evaluation, and infrastructure components are presented in Section 3. The evaluation methodology and the implementation of the SWS discovery workflow in BPEL is described in Section 4. Implementation results and lessons learnt are discussed in Section 5. Finally, the conclusion and future work is described in Section 6.

[1] http://www.bpmi.org/
[2] http://www.w3.org/TR/soap/
[3] http://www.w3.org/TR/wsdl/
[4] http://www.oasis-open.org/committees/uddi-spec/
[5] http://www.w3.org/TR/ws-cdl-10/
[6] http://www.oasis-open.org/committees/wsbpel/
[7] http://www.oasis-open.org/committees/bpel4people/
[8] http://www.omg.org/spec/BPMN/
[9] http://www.seals-project.eu/

2 Related Work

The section provides an overview of the related research work, which can be grouped according to the following areas: SWS tools evaluation and Workflow automation.

With respect to *SWS tools evaluation* area, there have been several evaluation initiatives with the objective of comparatively accessing performance, scalability and effectiveness of multiple SWS tools such as S3 (Semantic Service Selection) [3], the IEEE Web Service Challenge (WSC) [4], SWS Challenge [5] and the JGD Benchmark [6]. These initiatives produced a considerable amount of SWS test data sets for SWS tools benchmarking such as OWLS Test Collection[10], SAWSDL Test Collection[11] (counterpart of OWLS-TC that has been semi-automatically derived from OWLS-TC), Jena Geography Dataset[12] and others. The WS Challenge describes a platform that allows evaluating Semantic Web service composition based on the OWLS, WSDL, WS-BPEL schemas data and contest data formats, by different tools [7]. The authors specify WS-BPEL based orchestration format as the output format for the composition solutions.

In the S3 contest, the Semantic Web Service Matchmaker Evaluation Environment (SME2)[13] has been developed, which is a Java framework for evaluating SWS matchmakers over specified test collections. The environment is managed from a Graphical User Interface (GUI), where an end user can select a test collection, a matchmaker, specify evaluation options and use the control panel to manage the execution of the evaluation process itself [8]. Additionally, there exist particular requirements for the evaluation of a matchmaker into the SME2, which include implementation of the provided interface, creation of an XML plugin description file and deployment of created files into the plugin directory of SME2. After the evaluation process is finished, the user can navigate through the results tab in order to manage conducted experiments (i.e. *merge*, *split*, *load*, *save*) and select the visualization type.

In addition, we investigated solutions that focus on automation of workflows for software evaluation (i.e. graphical and execution standards and approaches). In [9] the authors present a framework for comparing scientific workflow systems based on their data/control flows properties and discusses the benefits and limitations of the following major workflow systems: Discovery Net[14], Taverna[15], Triana[16], Kepler[17], YAWL[18] and WS-BPEL. Although all the workflow systems offer the variety of data and control elements, only BPEL has been standardized for the business process domain [9].

[10] http://projects.semwebcentral.org/projects/owls-tc/
[11] http://projects.semwebcentral.org/projects/sawsdl-tc/
[12] http://fusion.cs.uni-jena.de/professur/jgd/
[13] http://projects.semwebcentral.org/projects/sme2/
[14] http://www3.imperial.ac.uk/lesc/projects/archived/discoverynet
[15] http://www.taverna.org.uk/
[16] http://www.trianacode.org/
[17] https://kepler-project.org/
[18] http://www.yawl-system.com/

3 Evaluation Platform Overview

In this section we present the basic terms used in SEALS and provide an overview of the evaluation components of the SEALS platform.

3.1 Terminology

In terms of SEALS by *evaluation* we imply behaviour examination of a particular tool under certain conditions and with particular input test data [10], [11]. *Evaluation workflow* defines how the evaluation use case is addressed and conducted; it specifies the way the input data is consumed and the output is presented [11]. *Evaluation description* refers to the test data and tools participating in a specific evaluation scenario and provides the evaluation workflow for this scenario. By *evaluation campaign* we imply an activity, where participant's tools are evaluated by executing evaluation campaign scenarios [11]. The SEALS evaluation campaigns[19] are organized for different types of tools, including SWS tools evaluation, where tools can be benchmarked and compared by tool providers.

3.2 Platform Evaluation components

The SEALS platform consists of four major components: *Runtime Evaluation Service (RES)*, which runs evaluation according to a specific evaluation description using a certain tool against particular datasets, *SEALS Service Manager (SSM)*, responsible for coordinating platform modules and ensuring consistency, *SEALS Repositories* and *SEALS Portal*, represented by a web user interface, which allows end user to interact with the SEALS platform [12]. 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 tools evaluation. *SEALS Test Data Repository Service (TDRS)* provides access to the test data repository and allows retrieving the whole test data in a ZIP file as well as specific test items directly from the repository and iterate over the suite. *SEALS Results Repository Service (RRS)* provides access to the results repository in order to add, retrieve raw results and interpretation. In *SEALS Tools Repository Service (TRS)* tools, which are used in the evaluation, are stored. Furthermore, several additional services are used in the workflow such as *SEALS Results Composer Service*, which provides functionality for building results and making a ZIP bundle out of them and *SEALS Tool service*, which provides access to the deployed tool.

[19] http://www.seals-project.eu/seals-evaluation-campaigns/

4 Approach for SWS Tools Automatic Evaluation

In this section we provide an overview of the methodology used to implement the SWS Discovery evaluation workflow and describe the workflow in details.

4.1 Methodology

In SEALS, a campaign organizer initiates the evaluation execution and the evaluation request is sent to internal platform components. At a higher level, orchestrations of the activities that enable the evaluation, originate in the SSM component [13]. Afterwards, computing resources, required for executing an evaluation, are requested and scheduled. When the resources are ready for evaluation the SSM notifies the RES component, which starts the evaluation workflow. At the level of a specific evaluation scenario, the modelling and automatic execution is enabled by using WS-BPEL, which was chosen as the language for evaluation workflow specification.

We define and implement the following methodology for SWS tools automatic evaluation using the SEALS platform. At the initial stage we create a set of Web services that can be generically used to access testdata for SWS discovery. For this purpose we define appropriate metadata according to the SEALS uniform ontologies[20].

Secondly, in order to ensure automatic tools benchmarking we specify a tool plug-in, and a set of Web services, which can initiate discovery tasks for the evaluation workflow.

Thirdly, the orchestration and automatic execution of the Web Services used in a specific evaluation scenario are accomplished by using WS-BPEL technology as explained in the next section.

Finally, the results of the SWS tools evaluation are stored in the SEALS Repositories, where raw results and measurements are described using RDF/OWL. The results in the repository can be mapped for different types of visualisation and presented to the end user.

4.2 Evaluation Workflow Description

We describe the SWS discovery evaluation workflow as implemented in WS-BPEL. The workflow, as shown in Figure 1, invokes a number of SOAP based Web Services, which access the SEALS repositories for metadata and datasets as well as invoke tool's specific methods and apply evaluation measures (interpretations).

As presented in Figure 1, the WS-BPEL activities represent services associated with specific evaluation artifacts, to which different prefixes have been assigned, as follows:

– **tdrs**: SEALS Test Data Repository Service.

[20] http://www.seals-project.eu/ontologies/

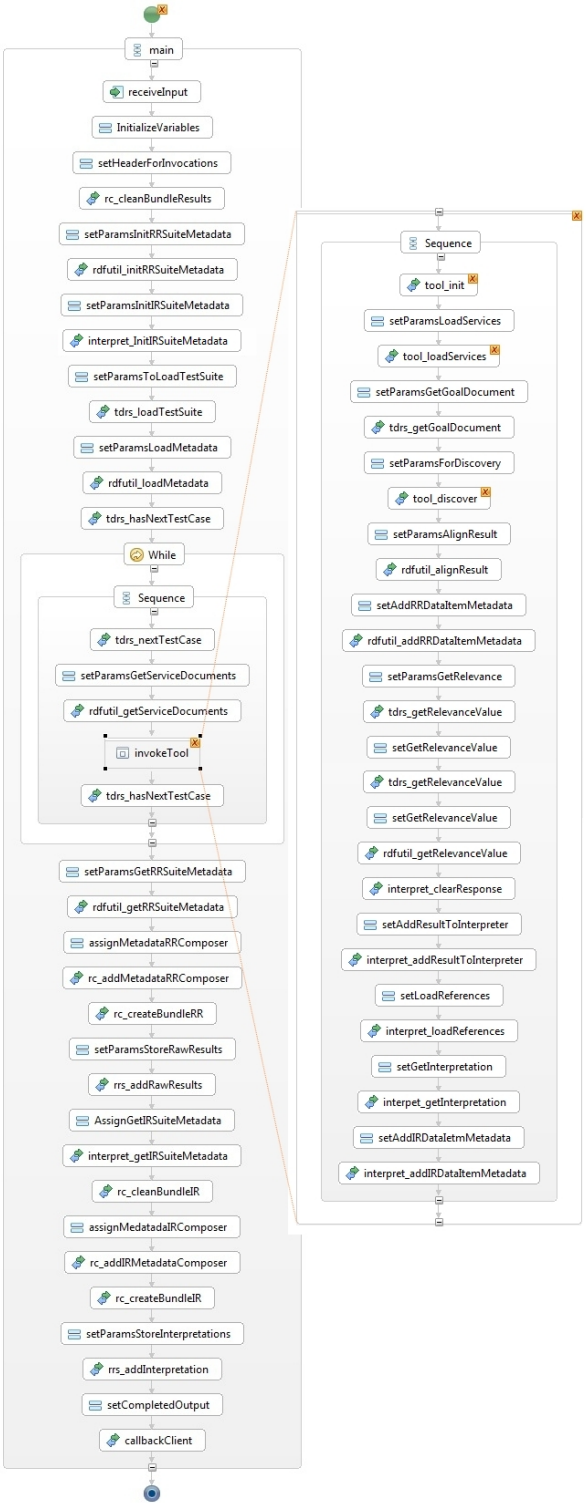

Fig. 1. SWS Discovery tools evaluation workflow. Eclipse WS-BPEL Designer was used to construct the WS-BPEL workflow.

– **rrs**: SEALS Results Repository Service.
– **tool**: SEALS Tool service.
– **rc**: SEALS Result Composer Service.
– **rdfutil**: Customized services specific to SWS evaluation, which extends basic functionality provided by the SEALS platform.
– **interpret**: Customized services used for handling interpretation results (measurements) within the evaluation.

Below in a sequential way we describe each WS-BPEL activity from the evaluation workflow.

– **RecieveInput**: receives initial input parameters such as tool identifier and tool version, test data identifier and results identifiers specified by user.
– **CleanBundleResults**: initializes the results' bundle.
– **InitResultSuiteMetadata**: creates metadata header and repository metadata consuming as an input test suite, tool and results identifiers. This method creates constant metadata, which is invoked from the WS-BPEL workflow only once before loading a test collection. The repository metadata is needed when the ZIP files with raw results and interpretations are uploaded to the Result Repository from a local temporary directory.
– **LoadTestSuite**: loads a test suite specified by end user in an evaluation request.
– **LoadMetadata**: loads the information that is contained within the test data metadata and will be used by the following methods to provide their functionality.
– **HasNextTestCase**: searches next test case in a test data and returns a boolean value.

After the initialization phase the evaluation is carried out in a while loop, where an iteration over all discovery tests cases is performed. The tests are loaded sequentially by executing the *nextTestCase* operation. In each iteration we extract all necessary data for a discovery test.

– **NextTestCase**: locates next test case in the test data.
– **GetServiceDocuments**: analyzes the metadata and for a given goal extracts identifiers (*dc:identifier*) for all relevant service documents' URLs. This list of URLs is passed to the workflow engine and then forwarded to the tool, which uses the URLs to perform the SWS discovery.
– **Initialize**: initializes SWS tool.
– **LoadServices**: invokes a tool and loads service documents.
– **GetGoalDocument**: retrieves the goal document location for a relevant test case from the TDRS.
– **Discover**: invokes tool's discovery method.
– **AlignResult**: obtains the result of discovery tools evaluation as input and exchanges the URLs that are used by the URIs, which consecutively are applied to reference goal and service documents.
– **AddDataItemMetadata**: serializes results for a particular test case.

- **Tdrs:GetRelevanceValue**: extracts relevance value document location from the Testdata Repository.
- **Rdfutil:GetRelevanceValue**: XML document, which contains relevance values is parsed by Rdfutil custom service.
- **ClearResponse**: initializes all components that are necessary to calculate discovery interpretation.
- **AddResultToInterpreter**: adds the result to interpretation service.
- **LoadReferences**: adds reference values for a given goal document to the set of results, which will be considered when calculating overall interpretation.
- **GetInterpretation**: retrieves the latest raw result and set of reference values that have been provided to the Web service and calculate interpretation from these values. The interpretation object is returned to the workflow engine to be serialized to RDF format.
- **GetResultSuiteMetadata**: the whole raw results and interpretation metadata models are built up.
- **AddMetadataResultComposer**: results are passed to the results composer, which structure them for the following operation.
- **CreateBundleResult**: creates a ZIP bundle with the raw result and interpretation.
- **AddResults**: adds the results′ bundles to the Result Repository Service.
- **CleanBundleInterpretation**: cleans bundle's generation functionality in order to be reused for bundling interpretation.
- **CallbackClient**: asynchronous WS-BPEL callback is used in order to handle long-lived invocations.

SWS discovery results can be queried from the SEALS portal using results identifier or name. Furthermore, in our discovery workflow error handling is used. By an *evaluation error* or *fault* we imply any kind of faults, which prevent WS-BPEL from completing its processes successfully [14]. Multiple scopes and faults types were introduced in WS-BPEL to tackle multiple errors, which could occur at runtime such as tool scope that handle tool bridge and tool wrapper errors, unexpected platform faults or custom service faults. Depending on the faults criticality the evaluation workflow is either forced to stop and throw a platform fault or if an error does not have a serious effect on the normal flow, the workflow resumes its execution with the next test case.

5 Discussion

According to the general objective of the SEALS project, in order to ensure automation of semantic technologies evaluation we aimed at providing a flexible, scalable, reusable and robust evaluation service. The section reports on the implementation results and discusses lessons learnt with respect to the evaluation workflow implementation at the current stage.

In [15] the authors introduce requirements for scientific workflow assessment such as *modular design, exception handling, compensation handler, adaptivity*

and flexibility, and, finally, *management of workflow*. These criteria will be used as a basis for examination of the SWS tools evaluation workflow.

With respect to **modular design**, the usage of WS-BPEL allowed decoupling business logic and treating each service separately, which was important for the platform and increased the reusability of components within different semantic evaluation workflows.

As discussed in Section 4 the SWS workflow implements **exception handling and compensation mechanisms**, which are critical issues in order to debug problems during the evalution.

The authors in [15] associate **adaptivity** with an ability of swift and effective adjustment (redesign) to meet the changing requirements. In SEALS SWS tools evaluation adaptability is supported at WS-BPEL side since the SEALS platform supports integration of custom services in SWS tools evaluation description by means of creating a custom service definition, implementing business logic, specifying how the service is connected to the platform's components and, finally, integrating it within the WS-BPEL. It can be stated that adaptability supports **flexibility** in this case since custom procedures allow redefining or replacing workflow activities; moreover, the SEALS platform supports generic tool interface, which provides a possibility for any tool to be integrated in the evaluation workflow. Depending on the requirements for workflow flexibility in order to model and orchestrate business processes of a system, the suitable trade-off orchestration standard should be considered since tight restrictions, imposed on Web services composition, allow ensuring control and monitoring while less restricted environment may threaten system's stability at runtime [16].

A **workflow management** perspective involves breakpoints mechanisms that provide a possibility to split the workflow into the following sub-components to facilitate persistence and fault tolerance: *steering*, used for established breakpoints, *monitoring, tasks rescheduling and reordering* [15]. Regarding SWS tools evaluation workflow, its management is handled by the RES and SSM components of the SEALS platform to ensure robustness and scalability of the evaluation.

Moreover, from **usability** view point we recommend to use proprietary engines in large scale projects, which has considerable amount of built-in functions that ease implementation process and aids in debugging the code. Besides, it is important to verify in advance if the version of executing engine, which was Apache ODE[21] in our case, supports all the required elements and components of WS-BPEL specification. Additionally, in our evaluation workflow XSL transformation[22] (*doXslTransform* function) was used to dynamically query web services response and present it in a format to be suitable for the input of the next operation, as defined in WSDL data types. In our case the usage of XSL transformation replaced the need of creating additional custom service that could also parse the response and align the output message according to XML schema.

[21] http://ode.apache.org/
[22] http://www.w3.org/TR/xslt

However, it should be also considered that the version of execution engine, used in the evaluation, supports XSLT.

6 Conclusions and Future Work

To the best of our knowledge, the automatic evaluation workflow of SWS tools is a unique approach in tool evaluation field for SWS discovery. In the project we sought constructing a workflow with the ultimate focus on ensuring seemless interaction with the platform, and minimizing the need for user interaction.

In contrast to the S3 contest described in section 2, in SEALS we aimed at ensuring automatic evaluation for large spectrum of semantic technologies. Hence, enhancing the current state of the art, the SEALS SWS tools evaluation was elaborated using a top-down structured approach, with a particular focus on modularity required for establishing efficient reuse among different types of evaluations of semantic technologies.

Similarly to the S3 contest, the wrapped tool name, its version, and the test data are specified in the evaluation execution request, and, thereafter, the evaluation is run. In contrast to the SME2, in SEALS the end user does not interact with the evaluation platform directly, rather manages it from the SEALS web portal. Furthermore, alternatively to the S3 contest, where an end user can specify which metrics to use from the platforms GUI, the SEALS evaluation provides a Web Service interface for measurements, which can be extended for applicable evaluation criteria for all SWS discovery tools.

In the SEALS project the automation of workflow evaluation is implemented using WS-BPEL orchestration language. The WS-BPEL technology is chosen since it represents a standardized solution for managing Web services based on clear XML standards, allows separating business logic and ensures reusability.

The current version of the SWS tools evaluation workflow at the time of writing has been integrated in the SEALS platform in the virtual environment and allowed the tools, deployed in advance, to be executed at runtime. However, while elaboration of services is done iteratively, the work towards their implementation for the automatic evaluation of SWS tools will be continued.

As future work we are planning to use our approach for evaluation of other types of SWS activities, not only discovery as presented in this paper.

Acknowledgements

The authors would like to thank the members of the SEALS consortium. This work has been partially supported by the SEALS EU FP7 project (IST-2009-238975).

References

1. BPMI: BPMI.org: Phase 2 Insight, Innovation, Interoperability (2004) Available at: `http://www.bpmi.org/`, Accessed: 09/02/2012.

2. Cabral, L., Toma, I.: Evaluating semantic web service tools using the seals platform. In Gómez-Pérez, A., Ciravegna, F., Harmelen, F., Hefflin, J., eds.: Proceedings of the International Workshop on Evaluation of Semantic Technologies (IWEST 2010). Volume 666., http://CEUR-WS.org (Nov 2010)
3. S3: Annual International Contest S3 on Semantic Service Selection - Retrieval Performance Evaluation of Matchmakers for Semantic Web Services (2011) Available at: `http://www-ags.dfki.uni-sb.de/~klusch/s3/index.html`, Accessed: 09/02/2012.
4. WSC: Introducing the Web Service Challenge (2008) Available at: `http://ws-challenge.georgetown.edu/ws-challenge/`, Accessed: 09/02/2012.
5. SWSC: Semantic Web Service Challenge: Evaluating Semantic Web Services Mediation, Choreography and Discovery Wiki (2011) Available at: `http://sws-challenge.org/`, Accessed: 09/02/2012.
6. Küster, U.: An Evaluation Methodology and Framework for Semantic Web Services Technology. PhD thesis, Friedrich-Schiller-University Jena (Feb 2010)
7. Bleul, S., Weise, T., Geihs, K.: The web service challenge - a review on semantic web service composition. In Wagner, M., Hogrefe, D., Geihs, K., David, K., eds.: Service-Oriented Computing (SOC'2009), The European Association of Software Science and Technology (2009)
8. Klusch, M., Dudev, M., Misutka, J., Kapahnke, P.: Semantic Web Service Matchmaker Evaluation Environment, SME2, Version 2.1. Technical report, S3 Contest: Performance of Semantic Web Service Matchmaking Tools (2010) Available at: `http://projects.semwebcentral.org/projects/sme2/`, Accessed: 20/02/2012.
9. Curcin, V., Ghanem, M.: Scientific workflow systems - can one size fit all? In: Biomedical Engineering Conference, 2008. CIBEC 2008. Cairo International. (Dec 2008) 1–9
10. Esteban-Gutiérrez, M., García-Castro, R., Gómez-Pérez, A.: Executing Evaluations over Semantic Technologies using the SEALS Platform. In: Proceedings of the International Workshop on Evaluation of Semantic Technologies (IWEST 2010). Volume 666., http://CEUR-WS.org (2010)
11. Schneider, M., Grimm, S., Abecker, A., Titov, A.: SEALS D8.1: Design of the Architecture and Interfaces of the Evaluation Descriptions Repository Service. Technical report, SEALS Project (2009)
12. Wrigley, S., García-Castro, R., Trojahn, C.: Infrastructure And Workflow for the Formal Evaluation of Semantic Search Technologies. In: In Proceedings of the 2011 workshop on Data infrastructures for supporting information retrieval evaluation. DESIRE '11, ACM Press (2011) 29–34
13. Esteban-Gutiérrez, M.: SEALS D4.4: Iterative Evaluation and Implementation of the SEALS Service Manager v.1.0-FR. Technical report, SEALS Project (2011)
14. Esteban-Gutiérrez, M.: SEALS D9.1: Iterative Avaluation and Implementation of the Runtime Evaluation Service v.1.1. Technical report, SEALS Project (2011)
15. Akram, A., Meredith, D., Allan, R.: Evaluation of bpel to scientific workflows. In: Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on. Volume 1. (may 2006) 269–274
16. Regev, G., Bider, I., Wegmann, A.: Defining business process flexibility with the help of invariants. Software Process: Improvement and Practice **12**(1) (2007) 65–79