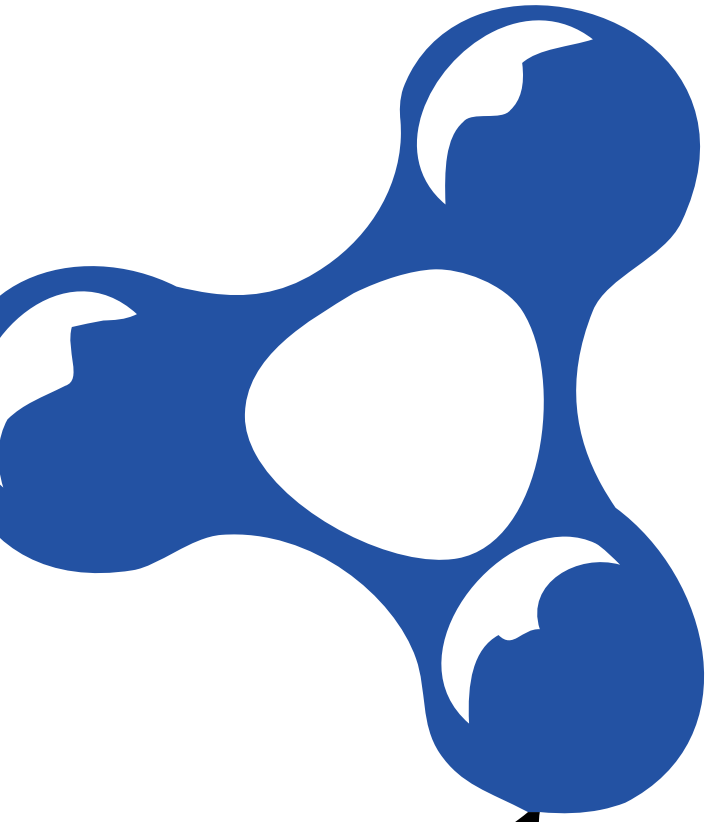

SEVENTH FRAMEWORK PROGRAMME

The state of semantic technology today – Overview of the First SEALS Evaluation Campaigns

Raúl García-Castro, Mikalai Yatskevich, Cássia Trojahn dos Santos, Stuart N. Wrigley, Liliana Cabral, Lyndon Nixon and Ondřej Šváb-Zamazal

April 2011

Document Information

IST Project Number	FP7 – 238975	**Acronym**	SEALS
Full Title	Semantic Evaluation at Large Scale		
Project URL	`http://www.seals-project.eu/`		

Authors	Raúl García-Castro (Universidad Politécnica de Madrid), Mikalai Yatskevich (University of Oxford), Cássia Trojahn dos Santos (INRIA), Stuart N. Wrigley (University of Sheffield), Liliana Cabral (Open University), Lyndon Nixon (STI) and Ondřej Šváb-Zamazal (University of Economics, Prague, Czech Republic)			
Contact Author	**Name**	Raúl García-Castro	**E-mail**	rgarcia@fi.upm.es
	Inst.	Universidad Politécnica de Madrid	**Phone**	+34 91 336 3670

Abstract	This report describes the first five SEALS Evaluation Campaigns over the semantic technologies covered by the SEALS project (ontology engineering tools, ontology reasoning systems, ontology matching tools, semantic search tools, and semantic web service tools). It presents the evaluations and test data used in these campaigns and the tools that participated in them along with a comparative analysis of their results. It also presents some lessons learnt after the execution of the evaluation campaigns and draws some final conclusions.
Keywords	semantic technology, evaluation, evaluation campaign, ontology engineering, reasoning, ontology matching, semantic search, semantic web service

Table of Contents

List of Figures

List of Tables

1. Introduction

The role of the SEALS initiative is two-fold: to create a lasting infrastructure for evaluating semantic technologies and to organise and execute two series of international evaluation campaigns over the different types of semantic technologies covered in the project.

Over the past 18 months, the SEALS consortium has designed and implemented a general methodology for carrying out evaluation campaigns; within this framework, the consortium has created the infrastructure for organising five international evaluation campaigns focussed on ontology engineering tools, ontology reasoning systems, ontology matching tools, semantic search tools and, finally, semantic web services. Each of these five evaluation campaigns was conducted during the Summer of 2010.

This report provides a summary of these first five SEALS Evaluation Campaigns; further details about the evaluation campaigns and its results can be found in the SEALS public deliverables[1] devoted to each of the campaigns.

The chapters about the different evaluation campaigns are similarly structured, covering previous evaluations related to the evaluation campaign, an overview of the evaluation scenarios defined for the evaluation campaign and of the test data used in them, the tools that have participated in the evaluation campaign, the results for these tools in the different evaluation scenarios, and the main lessons learnt during the evaluation campaign.

The report is structured as follows. Chapter 2 gives an overview of the common process followed in the SEALS Evaluation Campaigns and of how these campaigns were organised. Chapters 3, 4, 5, 6, and 7 each describes one of the evaluation campaigns over ontology engineering tools, ontology reasoning systems, ontology matching tools, semantic search tools, and semantic web service tools, respectively. Finally, chapter 8 draws some final conclusions for this report.

[1] http://about.seals-project.eu/deliverables

2. Overview of the SEALS Evaluation Campaigns

2.1 Technologies covered in the evaluation campaigns

As mentioned in the introduction, in the SEALS Evaluation Campaigns we dealt with five different types of semantic technologies: ontology engineering tools, ontology reasoning systems, ontology matching tools, semantic search tools, and semantic web services.

Figure 2.1 shows how these evaluation campaigns map the functionality-based categorization of the Semantic Web Framework [29]. We can see that in this first set of evaluation campaigns we fully cover the "Querying and Reasoning" dimension and partially cover the "Ontology Engineering" and "Semantic Web Services" ones.

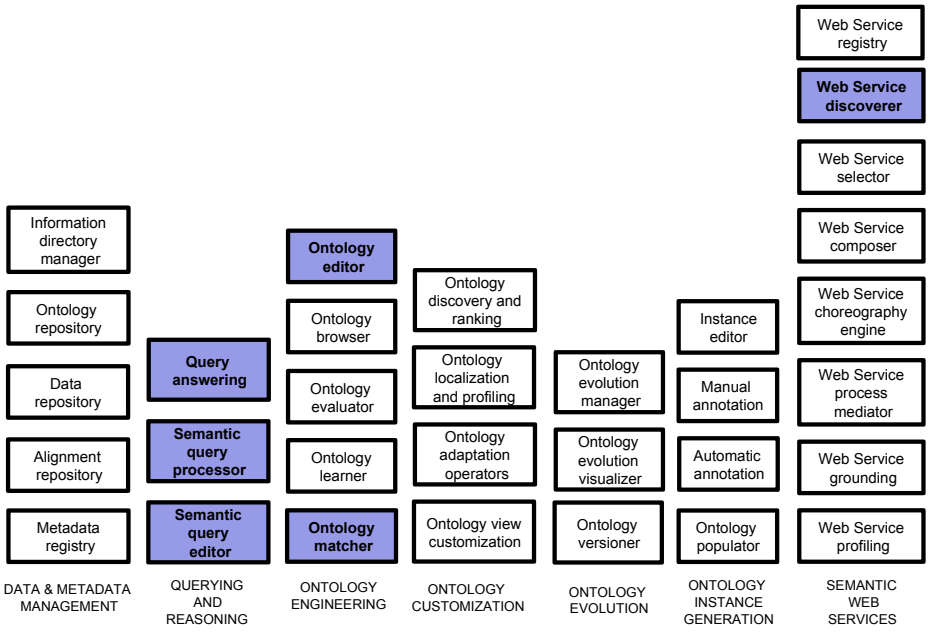

Figure 2.1: Categories covered in the Semantic Web Framework.

2.2 The SEALS evaluation campaign process

In the SEALS project, a common methodology and process for organizing an executing evaluation campaigns was defined, based in an analysis of previous evaluation campaigns in different domains [27].

The tasks of the evaluation campaign process are carried out by different actors: one Organizing Committee is in charge of the general organization and monitoring of all the evaluation campaigns, each evaluation campaign will have an Executing Committee in charge of organizing the evaluation scenarios that are performed in the evaluation campaign and of taking them to a successful end, and each evaluation campaign will have a number of participants, tool providers or people with the permission of tool providers, that participate with a tool in the evaluation campaign.

Since the evaluation campaign process must be general to accommodate different types of evaluation campaigns, it only suggests a set of general tasks to follow, not imposing any restrictions or specific details in purpose. Some tasks have alternative paths that can be followed; in these cases, all the possibilities are presented so the people carrying out the task can decide which path to follow. Moreover, the description of the tasks is completed with a set of recommendations extracted from the analysis of other evaluation campaigns.

Figure 2.2: The evaluation campaign process.

The evaluation campaign process is composed of four main phases (shown in Figure 2.2). The main goals of these phases are the following:

- **Initiation phase**. It comprises the set of tasks where the different people involved in the organization of the evaluation campaign and the evaluation scenarios are identified and where the different evaluation scenarios are defined.
- **Involvement phase**. It comprises the set of tasks in which the evaluation campaign is announced and participants show their interest in participating by registering for the evaluation campaign.
- **Preparation and Execution phase**. It comprises the set of tasks that must be performed to insert the participating tools into the evaluation infrastructure and to execute each of the evaluation scenarios and analyse their results.
- **Dissemination phase**. It comprises the set of tasks that must be performed to disseminate the evaluation campaign results by publicly presenting them and to make all the evaluation campaign result and resources available.

The activities performed in the SEALS evaluation campaigns in each of these four phases (and in the tasks that constitute them) are next described.

Initiation. During this phase, an initial effort was performed to initiate and coordinate all the evaluation campaigns.

To this end, first, the organizers of the evaluation campaigns were identified. In SEALS there is one committee in charge of the general organization and monitoring of all the evaluation campaigns and there have been different committees in charge of organizing the evaluation scenarios of each evaluation campaign and of taking them to a successful end.

Then, the different evaluation scenarios to be executed in each evaluation campaign were discussed and defined. This involved describing the evaluation to be performed over the tools and the test data to be used in it.

Involvement. In order to involve participants in the evaluation campaigns, the campaigns were announced using different mechanisms: the project dissemination mechanisms (e.g., portal, blog), relevant mailing lists in the community, leaflets and presentations in conferences and workshops, etc.

Participant registration mechanisms were prepared in the SEALS Community portal to allow potential participants to indicate their interest in the evaluation campaigns. Even if not every material to be used in the evaluation scenarios was ready by that time, this allowed involving participants early in the campaign.

Preparation and execution. In this phase, the organizers of each evaluation campaign provided to the registered participants with all the evaluation materials needed in the evaluation (e.g., descriptions of the evaluation scenarios and test data, instructions on how to participate, etc.). These materials were made available through the SEALS Community Portal.

In the SEALS project we have developed the SEALS Platform to support the execution of evaluations by providing different services to manage test data, execute evaluations, manage evaluation results, and so on.

Participants connected their tools with the SEALS Platform and, in the case of some relevant tools, members of the SEALS project connected these relevant tools.

Once all the participating tools were connected to the SEALS Platform, the different evaluation scenarios were executed with the corresponding test data and tools. The results obtained were stored in the platform and later analysed; in most of the cases, result visualisation services were developed to facilitate this analysis.

Dissemination. The results of all the evaluation campaigns were published in public SEALS deliverables and disseminated jointly in the International Workshop on Evaluation of Semantic Technologies[1] and separately in other events. Also, this report has been produced to provide an overview of the five evaluation campaigns and their results.

Finally, all the evaluation resources used in the evaluations have been made publicly available through the SEALS Platform.

2.3 Overview of the SEALS Platform

The SEALS Platform offers independent computational and data resources for the evaluation of semantic technologies and, as mentioned in the previous section, we used the first versions of the evaluation services developed for the platform to execute the evaluation scenarios of each evaluation campaign.

The SEALS Platform follows a service-oriented approach to store and process semantic technology evaluation resources. Its architecture comprises a number of components, shown in Figure 2.3, each of which are described below.

- **SEALS Portal.** The SEALS Portal provides a web user interface for interacting with the SEALS Platform. Thus, the portal will be used by the users for the

[1] http://oeg-lia3.dia.fi.upm.es/iwest2010/

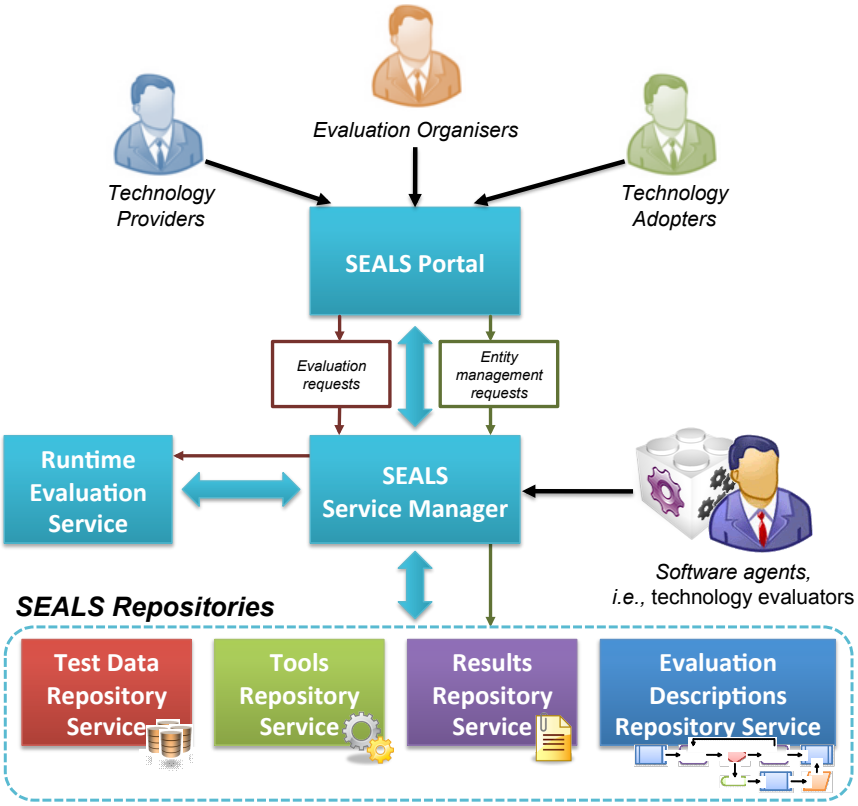

Figure 2.3: Architecture of the SEALS Platform.

management of the entities in the SEALS Platform , as well as for requesting the execution of evaluations. The portal will leverage the SEALS Service Manager for carrying out the users' requests.

- **SEALS Service Manager.** The SEALS Service Manager is the core module of the platform and is responsible for coordinating the other platform components and for maintaining consistency within the platform. This component exposes a series of services that provide programmatic interfaces for the SEALS Platform. Thus, apart from the SEALS Portal, the services offered may be also used by third party software agents.

- **SEALS Repositories.** These repositories manage the entities used in the platform (i.e., test data, tools, evaluation descriptions, and results).

- **Runtime Evaluation Service.** The Runtime Evaluation Service is used to automatically evaluate a certain tool according to a particular evaluation description and using some specific test data.

3. Ontology engineering tools evaluation campaign

The SEALS Evaluation Campaign for Ontology Engineering Tools included three scenarios to evaluate the conformance, interoperability and scalability of these tools. In the conformance and interoperability scenarios we aimed to fully cover the RDF(S) and OWL specifications; in the scalability scenario we evaluated tools using both real-world ontologies and synthetic test data.

3.1 Previous Evaluations

The first characteristic that we have covered in the evaluation campaign is **conformance**. Previously, conformance has only been measured in qualitative evaluations that were based on tool specifications or documentation, but not on running the tools and obtaining results about their real behaviour (e.g., the evaluation performed in the OntoWeb project [52] or the one performed by Lambrix and colleagues [41]).

Besides, some previous evaluations provided some information about the conformance of the tools since such conformance affected the evaluation results. This is the case of the EON 2002 ontology modelling experiment [1], the EON 2003 interoperability experiment [59], or the evaluations performed in the RDF(S) [24] and OWL [25] Interoperability Benchmarking activities.

However, currently the real conformance of existing tools is unknown since such conformance has not been evaluated. Therefore, we will evaluate the conformance of ontology engineering tools and we will cover the RDF(S) and OWL recommendations.

A second characteristic that we have covered, highly related to conformance, is **interoperability**. Previously, an interoperability experiment was proposed in the EON 2003 workshop [59] where participants were asked to export and import to an intermediate language to assess the amount of knowledge lost during these transformations.

Later, the RDF(S) [24] and OWL [25] Interoperability Benchmarking activities involved the evaluation of the interoperability of different types of semantic technologies using RDF(S) and OWL as interchange languages and provided a set of common test data, evaluation procedures and software to support these evaluations.

In this evaluation campaign we have extended these evaluations with test data for OWL DL and OWL Full to fully cover the RDF(S) and OWL specifications.

Scalability is a main concern for any semantic technology, including ontology engineering tools. Nevertheless, only one effort was previously performed for evaluating the scalability of this kind of tools (i.e., the WebODE Performance Benchmark Suite[1] [23]) and it was specific to a single tool.

In scalability evaluations, the generation of test data is a key issue. The WebODE Performance Benchmark Suite includes a test data generator that generates synthetic ontologies in the WebODE knowledge model according to a set of load factors and these ontologies can be later exported to several languages (RDF(S), OIL, DAML+OIL,

[1] http://knowledgeweb.semanticweb.org/wpbs/

OWL, etc.). Also, one of the most common test data generators used when evaluating ontology management frameworks is the Lehigh University Benchmark (LUBM)[32].

On the other hand, other evaluations use real ontologies as test data (e.g., subsets of ARTstor art metadata and the MIT OpenCourseWare metadata were used in the scalability evaluation performed in the SIMILE project [43].

In this first evaluation campaign we have established the grounds for the automatic evaluation of the scalability of ontology engineering tools, using both real ontologies and generated data, with the aim of proposing an extensible approach to be further extended in the future.

3.2 Evaluation Scenarios

The next sections describe three scenarios to evaluate the conformance, interoperability and scalability of ontology engineering tools.

The approach followed in them has been motivated by the need of having an automatic and uniform way of accessing most tools and, therefore, the way chosen to automatically access the tools is through the following two operations commonly supported by most of the tools: to import an ontology from a file, and to export an ontology to a file.

These two operations are viewed as an atomic operation. Therefore, there is not a common way of checking how good importers and exporters are; we just have the results of combining the import and export operation (the file exported by the tools). This causes that if a problem arises in one of these steps, we cannot know whether it was originated when the ontology was being imported or exported because we do not know the state of the ontology inside each tool.

3.2.1 Evaluating Conformance

The conformance evaluation has the goal of evaluating the conformance of semantic technologies with regards to ontology representation languages, that is, to evaluate up to what extent semantic technologies adhere to the specification of ontology representation languages.

During the evaluation, a common group of tests is executed in two steps. Starting with a file containing an ontology, the execution consists in importing the file with the ontology into the origin tool and then exporting the ontology to another file.

After a test execution, we have two ontologies in the ontology representation language, namely, the original ontology and the final ontology exported by the tool. By comparing these ontologies we can know up to what extent the tool conforms to the ontology language.

From the evaluation results, the following three metrics for a test execution can be defined:

- **Execution** (*OK/FAIL/P.E.*) informs of the correct test execution. Its value is *OK* if the test is carried out with no execution problem; *FAIL* if the test is carried out with some execution problem; and *P.E.* (Platform Error) if the evaluation infrastructure launches an exception when executing the test.

- **Information added or lost** shows the information added to or lost from the ontology in terms of triples. We can know the triples added or lost by comparing the original ontology with the final one; then we can store these triples in some human-friendly syntax (e.g., N3[2]).
- **Conformance** (*SAME*/*DIFFERENT*/*NO*) explains whether the ontology has been processed correctly with no addition or loss of information. From the previous basic metrics, we can define *Conformance* as a derived metric that is *SAME* if *Execution* is *OK* and *Information added* and *Information lost* are void; *DIFFERENT* if *Execution* is *OK* but *Information added* or *Information lost* are not void; and *NO* if *Execution* is *FAIL* or *P.E.*.

3.2.2 Evaluating Interoperability

The interoperability evaluation has the goal of evaluating the interoperability of semantic technologies in terms of the ability that such technologies have to interchange ontologies and use them.

In concrete terms, the evaluation takes into account the case of interoperability using an interchange language, that is, when an ontology is interchanged by storing it in a shared resource (e.g., a fileserver, a web server, or an ontology repository) and is formalised using a certain ontology language.

During the evaluation, a common group of tests is executed in two sequential steps. Let start with a file containing an ontology. The first step consists in importing the file with the ontology into the origin tool and then exporting the ontology to a file. The second step consists in importing the file with the ontology exported by the origin tool into the destination tool and then exporting the ontology to another file.

After a test execution, we have three ontologies in the ontology representation language, namely, the original ontology, the intermediate ontology exported by the first tool, and the final ontology exported by the second tool. By comparing these ontologies we can know up to what extent the tools are interoperable. For each of the two steps and for the whole interchange we have metrics similar to those presented above for evaluating conformance. Therefore, we can use the *Execution* and *Information added and lost* metrics as well as an *Interoperability* one, which explains whether the ontology has been interchanged correctly with no addition or loss of information.

3.2.3 Evaluating Scalability

The scalability evaluation has the goal of evaluating the scalability of ontology engineering tools in terms of time efficiency. More concretely to our case, the scalability evaluation is concerned with evaluating the ability of ontology engineering tools to handle large ontologies.

The scalability evaluation scenario is similar to the conformance one. During the evaluation, a common group of tests is executed and each test describes one input ontology that has to be imported by the tool and then exported to another file. In the

[2] http://www.w3.org/TeamSubmission/n3/

evaluation we measure the time (in milliseconds) before the import/export operation is executed and the time after the import/export operation is executed.

After a test execution, we have as result:

- **Execution** informs of the correct test execution, as in the previous scenarios.
- **Import/export duration** contains the duration of the import/export operation in milliseconds.

3.3 Test Data

3.3.1 Conformance and Interoperability

This section presents the test data that has been used for the conformance and interoperability evaluations.

In the first evaluation campaign, the conformance and interoperability evaluations have covered the RDF(S) and OWL specifications. To this end, we will use four different test suites that contain synthetic ontologies with simple combinations of components of the RDF(S), OWL Lite, OWL DL, and OWL Full knowledge models.

These test suites are based in the following assumptions: (1) To have a small number of tests since, even if the execution of the tests is automatic, a large number of tests leads to an increment in the time required for analysing the results. (2) To use the RDF/XML syntax[3] for serialising ontologies since this syntax is the most used by Semantic Web tools for importing and exporting ontologies. (3) To define correct ontologies only. The ontologies defined in the tests do not contain syntactic or semantic errors. (4) To define simple ontologies only. This will allow detecting problems easily in the tools.

Any group of ontologies could have been used as input for the evaluations. For example, we could have employed a group of real ontologies in a certain domain, ontologies synthetically generated such as the Leigh University Benchmark (LUBM) [32] or the University Ontology Benchmark (UOBM) [46], or the OWL Test Cases[4] (developed by the W3C Web Ontology Working Group). Although these ontologies could complement our experiments, we aim at using simple ontologies that, even though they do not cover exhaustively the language specifications, are simple and allow isolating problem causes and highlighting problems in the tools. Besides, using real, large, or complex ontologies can be useless if we do not know whether the tools can deal with simple ontologies correctly.

The RDF(S) and OWL Lite Import Test Suites already exist (they were named the RDF(S) Import Benchmark Suite[5] and the OWL Lite Import Benchmark Suite[6], respectively) and detailed descriptions of them can be found in [22]. The OWL DL

[3] http://www.w3.org/TR/rdf-syntax-grammar/
[4] http://www.w3.org/TR/owl-test/
[5] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/rdfs/rdfs_import_benchmark_suite.html
[6] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/owl/import.html

and OWL Full Import Test Suites have been developed in the context of the SEALS project and are described in [28]. Next, we provide a brief description of them.

The **OWL DL Import Test Suite** contains OWL DL ontologies that have been generated following a keyword-driven test suite generation process implemented by the OWLDLGenerator tool[7].

The OWLDLGenerator tool uses a Keyword Library that contains the definition of all the keywords that can be used to define conformance tests. In our case, these keywords define combinations of ontology components that can be composed to build the ontology that will be used in the test and have been extracted from the OWL abstract syntax grammar [53], which defines the production rules that can be used to generate OWL DL ontologies.

We defined 561 tests to cover all the simple combinations of components of the OWL DL knowledge model that can be extracted from the abstract syntax grammar and classified them in groups according to the combinations of components they cover (e.g., class descriptions, property descriptions, etc.). The script used to generate the OWL DL Import Test Suite can be found online[8].

The **OWL Full Import Test Suite** is complementary to both the RDF(S) Import Test Suite and the OWL DL Import Test Suite. On the one hand, the test suite provides ontologies that are syntactically valid in OWL Full but generally invalid in OWL DL. On the other hand, the test suite makes specific use of OWL vocabulary terms and therefore goes beyond the typical content of RDF(S) ontologies.

A basic design principle when defining this test suite was to focus on *distinctive syntactic features* of OWL Full, that is, features that are outside the syntactic specification of OWL DL and that will typically not be found in RDF(S) ontologies.

For each of the defined features, one ontology has been developed and can be seen as an "representative" for the feature. Finally, the set of these 90 representative ontologies builds the OWL Full Import Test Suite.

3.3.2 Scalability

For the scalability evaluations, two test suites were defined. The first one is based on existing real-world ontologies and the second one has been generated using the LUBM ontology generator.

The Real-World Ontologies Scalability Test Suite includes real-world ontologies in OWL DL that have been identified in [26] as being relevant for scalability evaluation. We have selected 20 ontologies of various sizes (up to 37.7 Mb): AEO[9], the NCI

[7] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/OWLDLGenerator/

[8] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/OWLDLGenerator/OWLDLTSScript.csv

[9] http:/www.boemie.org/ontologies

Thesaurus[10], GALEN[11], the Foundational Model of Anatomy Ontology[12] (FMA), the OBO Foundry[13], Robert's family ontology, and the wine and food ontologies[14].

The LUBM Generator Scalability Test Suite contains ontologies generated using the Lehigh University Benchmark (LUBM) data generator [32]. Five ontologies with the correspondence instances based on UBA 1.7 were generated. The number of universities, which is the minimum unit of data generation, is increasing with each test in order to find out how the tool scales with the linearly growing data. Therefore the size of the generated ontologies varies from 8 Mb up to 50 Mb.

3.4 Tools Evaluated

In the first evaluation campaign over ontology engineering tools we have evaluated six different tools: three ontology management frameworks (Jena[15], the OWL API[16], and Sesame[17]) and three ontology editors (the NeOn Toolkit[18], Protégé OWL[19], and Protégé version 4[19]).

Sometimes tools use ontology management frameworks for processing ontologies, which has an effect in their conformance and interoperability. Table 3.1 shows the tools evaluated and the ontology management frameworks (i.e., APIs) that they use, including in both cases their version numbers.

Table 3.1: List of tools evaluated.

Ontology management frameworks			
Tool	**Version**		
Jena	2.6.3		
OWL API	3.1.0 1592		
Sesame	2.3.1		
Ontology editors			
Tool	**Version**	**API**	**API version**
NeOn Toolkit	2.3.2	OWL API	3.0.0 1310
Protégé 4	4.1 beta 209	OWL API	3.1.0 1602
Protégé OWL	3.4.4 build 579	Protégé OWL API	3.4.4 build 579

[10] http://www.mindswap.org/papers/WebSemantics-NCI.pdf
[11] http://www.opengalen.org/index.html
[12] http://sig.biostr.washington.edu/projects/fm/AboutFM.html
[13] http://obofoundry.org
[14] http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine
[15] http://jena.sourceforge.net/
[16] http://owlapi.sourceforge.net/
[17] http://www.openrdf.org/
[18] http://neon-toolkit.org/
[19] http://protege.stanford.edu/

3.5 Evaluation Results

3.5.1 Conformance

The conformance results show that Jena and Sesame have no problems when processing the ontologies included in the test suites for the different languages. Therefore, no further comments will be made about these tools.

Moreover, as shown in table 3.1, the NeOn Toolkit and Protégé 4 use the OWL API for ontology management.

The version of Protégé 4 that has been evaluated uses a version of the OWL API that is quite contemporary to the one evaluated. Therefore, after analysing the results of Protégé 4 we obtained the same conclusions as for the OWL API.

However, the version of the NeOn Toolkit that has been evaluated uses a version of the OWL API that differs in some months to the one evaluated. In general, from the results of the NeOn Toolkit we obtained the same conclusions as those obtained for the OWL API. In the next sections we will only note those cases where the behaviour of the NeOn Toolkit and the OWL API differ.

RDF(S) Conformance

When the **OWL API** processes RDF(S) ontologies, it always produces different ontologies because it converts the ontologies into OWL 2.

For this conversion, the OWL API applies a set of rules to transform RDF(S) ontology components into OWL 2. Some of these rules apply direct transformations on the components (e.g., instances are converted into OWL 2 named individuals) and other rules transform components according to their use in the ontology (e.g., undefined resources that have instances are defined as classes).

However, in some cases the application of a rule (or a combination of them) causes unexpected effects; for example, transforming an RDF property into both an object property and a datatype property or transforming classes into individuals.

When **Protégé OWL** processes an RDF(S) ontology, the ontology is always created as an OWL ontology with a randomly generated name[20]. Regardless of this, it always produces the same ontology except when the origin ontology contains a property with an undefined resource as range (the undefined resource is created as a class) or a literal value (which is created with a datatype of *xsd:string* and, therefore, it is a different literal. According to the RDF specification, one requirement for literals to be equal is that either both or neither have datatype URIs[21].

OWL Lite Conformance

When the **OWL API** processes OWL Lite ontologies, it converts the ontologies into OWL 2. Since OWL 2 covers the OWL Lite specification, most of the times the OWL API produces the same ontologies. However, one effect of this conversion is that individuals are converted into OWL 2 named individuals.

[20] E.g., http://www.owl-ontologies.com/Ontology1286286598.owl
[21] http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#dfn-typed-literal

The cases when the ontologies are different are when the ontology contains a named individual related through an object property with an anonymous individual, and this anonymous individual is related through a datatype property with a literal value. In this case, the named individual is related through the object property to an anonymous resource, another anonymous resource is related through a datatype property with a literal value, and the anonymous individual is not related to anything.

After analysing the results of the **NeOn Toolkit** we obtained the same conclusions that were previously presented for the OWL API with one exception. When the ontology contains an anonymous individual related to a named individual through an object property, the execution of the NeOn Toolkit fails.

When **Protégé OWL** processes an OWL Lite ontology, most of the times it produces the same ontology. The only exception is when the ontology contains a literal value, which is created with a datatype of *xsd:string* and, as mentioned above, it is a different literal.

OWL DL Conformance

When the **OWL API** processes OWL DL ontologies, it converts the ontologies into OWL 2. Since OWL 2 covers the OWL DL specification, most of the times the OWL API produces the same ontologies. However, one effect of this conversion is that individuals are converted into OWL 2 named individuals.

The cases when the ontologies are different are those when the ontology contains an anonymous individual related through an object property with some resource or through a datatype property with a literal value (in this case, an anonymous resource is related through the property with the resource or literal, and the anonymous individual is not related to anything) or a datatype property that has as range an enumerated datatype (in this case, an *owl:Datatype* class is created as well as an anonymous individual of type *owl:Datatype*. However, the *owl:Datatype* class does not exist in the RDF(S), OWL or OWL 2 specifications, only *rdfs:Datatype* exists).

Also, there is one case when the test execution fails; in the cases when the ontology imports another ontology, the OWL API does not produce any ontology. This happens because, since the OWL API cannot find the ontology in the *owl:imports* property, it does not import anything. However, the tool should not rely on having full access to the imported ontology for just importing one ontology.

After analysing the results of the **NeOn Toolkit** we obtain the same conclusions that were previously presented for the OWL API with one exception. When the ontology contains an anonymous individual related to another anonymous individual through an object property, the execution of the NeOn Toolkit fails.

When **Protégé OWL** processes OWL DL ontologies, it usually produces the same ontology. The cases when the ontologies are different are those when the ontology contains a literal value (the literal value is created with a datatype of *xsd:string* and, therefore, it is a different literal) or class descriptions that are the subject or the object of a *rdfs:subClassOf* property (in these cases, the class description is defined to be equivalent to a new class named "Axiom0", and this new class is the subject or the object of the *rdfs:subClassOf* property).

OWL Full Conformance

When the **OWL API** processes OWL Full ontologies, it produces different ontologies in the majority of cases. OWL API typically adds declaration triples for classes, object and data properties, and it also declares individuals as OWL 2 named individuals. In some cases it has been observed that an object property is additionally declared as a data or annotation property, or that a built-in vocabulary term is explicitly declared as a class or an object property. Even some built-in datatypes are occasionally declared as classes. For the encoding of property restriction class expressions, an *owl:Restriction* triple is being created if it is missing, an *owl:differentFrom* triple is replaced by the RDF encoding of an *owl:AllDifferent* axiom, and the order of elements in the argument list of a construct, such as *owl:intersectionOf*, is often being changed. There have also been heavily broken conversions, which may in some or all cases be due to errors in the OWL API. For example, under some circumstances, statements containing blank nodes are being completely removed.

Mainly the same conclusions as for the OWL API have been obtained for **Protégé 4** and the **NeOn Toolkit**. However, unlike the OWL API, the execution of the NeOn Toolkit fails on five of the eight test cases that probe the use of blank nodes in ways that are outside the scope of OWL DL.

The results for **Protégé OWL** differ from those of the rest of the evaluated tools in that it succeeds to reproduce the original ontology for the majority of test cases, but still fails or produces different results for 16 test cases. Execution fails, for example, when a blank node is associated to a URI via *owl:sameAs*. If an ontology header is missing, Protégé OWL adds one with a random ontology URI. On the other hand, some kinds of statements are being removed, such as *owl:sameAs* statements associating a blank node with a typed literal.

3.5.2 Interoperability

The conclusions about the behaviour of the tools that can be obtained from the interoperability results are the same as those already presented when analysing their conformance. The only new facts obtained while analysing the interoperability results are explained next.

In interchanges of OWL DL ontologies from the OWL API (or from those tools that use the OWL API, i.e., Protégé 4 and the NeOn Toolkit) to Protégé OWL, when the ontology contains an anonymous individual related through a datatype property with a literal value, Protégé OWL has execution problems.

In addition, when using the OWL Full test suite as test data, when the origin tools are the OWL API, Protégé 4 or the NeOn Toolkit and the destination tools are either Jena or Protégé OWL, in most of the cases, no ontology was produced by the destination tool. An analysis of the situation has revealed that the OWL API (which also underlies Protégé 4 or the NeOn Toolkit) produces broken URIs under certain circumstances. Most test cases in the current version of the OWL Full test suite use the ontology URI `http://example.org#h`. That URI can be found in the output ontologies produced by OWL API, but it is not a valid URI. Both Jena and Protégé

OWL stop by signalling an error for ontologies with this URI. All the other evaluated tools seem to be tolerant against this error.

In summary, in terms of interoperability:

- Regardless of the ontology language used in the interchange, while Jena and Sesame have no interoperability problems the rest of the tools have some issues that prevent their full interoperability.
- Tools based on the OWL API convert RDF(S) ontologies into OWL 2 and, therefore, they cannot interoperate using RDF(S) as the interchange language.

3.5.3 Scalability

Scalability evaluation was performed on the local machine with the following configuration: Intel Core 2 Duo CPU 2,40 GHz, 3,00 GB RAM, 64-bit OS. When running scalability evaluation with real-world ontologies some tools failed to load the ontologies. Sesame showed the best performance results in the scalability evaluation and could process large ontologies efficiently. Jena performed fast on processing small and medium sized ontologies (less than 10MB), while the large tests led to a significant execution time increase. As the OWL API is used in the NeOn Toolkit and Protégé version 4, the performance is practically the same for these tools and framework. There is a direct dependence between execution time and the size of ontologies for all tools and frameworks.

In the scalability results using the LUBM-generated ontologies, it is seen that the size of the ontology leads to a time increase, specifically there is a linear dependence between ontology size and execution time. The best performance was provided by Sesame for large ontologies, while Protégé OWL failed to execute the two scalability tests, which contained the largest ontologies, due to GC overhead limit excess. The results of OWL API, NeOnToolkit, Protégé 4 are relatively the same. In addition, those tools, which showed good results in exporting/importing relatively small ontologies, performed well with the largest ontologies.

3.6 Lessons Learnt

In the conformance and interoperability results, we can see that all those tools that manage ontologies at the RDF level (Jena and Sesame) have no problems in processing ontologies regardless of the ontology language. Since the rest of the tools evaluated are based in OWL or in OWL 2, their conformance and interoperability is clearly better when dealing with OWL ontologies.

From the results presented in chapter 3.5 we can see that conformance and interoperability are highly influenced by development decisions. For example, the decision of the OWL API developers (propagated to all the tools that use it for ontology management) of converting all the ontologies into OWL 2 makes the RDF(S) conformance and interoperability of these tools quite low.

Since the OWL Lite language is a subset of the OWL DL one, there is a dependency between the results obtained using the test suites for OWL Lite and OWL DL. In the

results we can also see that, since the OWL DL test suite is more exhaustive than the OWL Lite one, the OWL DL evaluation unveiled more problems in tools than the OWL Lite evaluation. These included issues not only related to the OWL DL language, but also related to OWL Lite ontologies included in the OWL DL test suite.

The results also show the dependency between the results of a tool and those of the ontology management framework that it uses; using a framework does not isolate a tool from having conformance or interoperability problems. Besides inheriting existing problems in the framework (if any), a tool may have more problems if it requires further ontology processing (e.g., its representation formalism is different from that of the framework or an extension of it) or if it affects the correct working of the framework.

However, using ontology management frameworks may help increasing the conformance and interoperability of the tools, since developers do not have to deal with the problems of low-level ontology management. Nevertheless, as observed in the results, this also requires being aware of existing defects in these frameworks and regularly updating the tools to use their latest versions.

The results of the scalability evaluation showed the linear dependence between the ontology size and export/import operations execution. However, the performance between the tools varies to a considerable extent, namely between Sesame, Jena and Protégé OWL. As the OWL API is used in the NeOn Toolkit and Protégé version 4, the performance is practically the same for them. Therefore, based on the obtained evaluation results we can conclude that Sesame is one of the most suitable tools for handling large ontologies.

While managing ontologies at an RDF level prevents conformance and interoperability problems, not every tool is able to do so. However, OWL (or OWL 2) tools include richer knowledge representation and reasoning capabilities that provide added value as long as users are aware of their potential problems (including those of the underlying ontology management frameworks that they use). In terms of scalability tool behaviour considerably varies, which requires analysing tools' efficiency in deep before opting for one.

4. Storage and reasoning systems evaluation campaign

The SEALS Storage and Reasoning Systems evaluation campaign focused on interoperability and performance evaluation of advanced reasoning systems. Class satisfiability, classification, ontology satisfiability and entailment evaluations have been performed on a wide variety of real world tasks.

4.1 Previous evaluations

Definition, execution, and analysis of evaluations for testing description logic based systems (DLBS) has been extensively considered in the past to compare the performances of these kind of systems and to prove their suitability for real case scenarios. The implementation of new optimisations for existing DLBS or the development of new DLBS has been disseminated together with specific evaluations to show how they improve the state of the art of DLBS.

Several attempts to systematise the evaluation of DLBS and to provide a lasting reference framework for automatisation of this kind of evaluations have failed in the past. The community of developers and researchers of DLBS still do not have a common open platform to execute evaluations and to study the results of these executions. The test data, DLBS and evaluation results were temporally available in dispersed Web sites that after some years are not longer available. Even with recent papers it is nearly impossible to reproduce and verify the evaluation results that their authors claimed.

However, all previous work on evaluation of DLBS provides a solid foundation to accomplish our objectives towards the correct design of evaluations and the implementation of specific software components for the execution and analysis of these evaluations using the SEALS platform.

For the sake of conciseness, we will discuss only some relevant previous contributions starting with the first notorious attempt of building a framework for testing ABox reasoning. Largely inspired by the Wisconsin benchmark [15] for testing database management systems, the Lehigh University Benchmark (LUBM) is still the facto standard for testing ABox reasoning. LUBM provides a simple TBox with 43 classes and 32 properties encoded in OWL-Lite. This TBox describes Universities, their departments and some related activities. LUBM also includes a synthetic data generator for producing ABoxes of different sizes. A set of 14 predefined SPARQL queries has been specifically designed for measuring five different factors related to ABox reasoning capabilities. LUBM was extended to provide some TBox reasoning evaluation support and to increase the complexity of the ABoxes generated. The UOBM [45] enriched the original TBox with new axioms that use most of OWL-DL constructors. In fact, UOBM provides two TBoxes, one in OWL-DL and one in OWL-Lite. The OWL-DL TBox has 69 classes and 43 properties, and the OWL-Lite TBox includes 51 classes and 43 properties. The ABox generator was also improved to provide higher connected ABoxes.

Gardiner et al. [30], developed a Java application for testing and comparing DLBS. This Java application used the DIG interface for accessing the system to be tested and MySQL for storing evaluation results. On the contrary to LUBM, the test data used was a selection of nearly 300 real ontologies of different size and expressivity. To warm-up DLBS prior the execution of an evaluation (classification of a selected ontology), the DLBS was forced to complete 4 tasks. Gardiner et al. were measuring the time needed to complete the classification reasoning task and they were comparing the classification results of several DLBS to check correctness.

Recent efforts of Luther et al. [44] have been focused on the study of the insights of testing environments and their influence in the evaluation results. They use the UOBM for evaluation of scalability for ABox reasoning, and they use a real TBox for testing classification. Luther et al. have shown the remarkable influence of native and standard (like the OWL API) interfaces or the impact of the version of the Java virtual machine used during the execution of the evaluations. Even, they compare the evaluation results of several versions of the same DLBS with interesting outcomes.

4.2 Evaluation scenarios

4.2.1 Evaluation Criteria

According to the ISO-IEC 9126-1 standard [37], interoperability is a software functionality sub-characteristic defined as "the capability of the software product to interact with one or more specified systems". In order to interact with other systems a DLBS must conform to the standard input formats and must be able to perform standard inference services. In our setting, the standard input format is the OWL 2 language. We evaluate the standard inference services:

- Class satisfiability;
- Ontology satisfiability;
- Classification;
- Logical entailment.

The last two are defined in the OWL 2 Conformance document[1], while the first two are extremely common tasks during ontology development, and are de facto standard tasks for DLBSs.

The performance criterion relates to the efficiency software characteristic from ISO-IEC 9126-1. Efficiency is defined as "the capability of the software to provide appropriate performance, relative to the amount of resources used, under stated conditions". We take a DLBS's performance as its ability to efficiently perform the standard inference services. We will not consider the scalability criterion for the Storage and Reasoning Systems Evaluation Campaign 2010 because suitable test data is not currently available. The reason for this is the fact that while hand crafted ontologies can be tailored to provide interesting tests, at least for particular systems it is very

[1] http://www.w3.org/TR/2009/PR-owl2-conformance-20090922/

difficult to create hand crafted ontologies that are resistant to various optimisations used in modern systems. Furthermore, hand crafted ontologies are rather artificial since their potential models often are restricted to those having a very regular structure. Synthetic DL formulas may be randomly generated [35, 48, 54]. Thus, no correct answer is known for them in advance. There have been extensive research on random ABox generation in recent years [31, 45]. These works are tailored to query answering scalability evaluation. Real-world ontologies provide a way to assess the kind of performance that DLBSs are likely to exhibit in end-user applications, and this is by far the most common kind of evaluation found in recent literature. However, it is not a trivial task to create a good scalability test involving real-world ontologies. To the best of our knowledge, no such tests are available at the moment. The particular problems are parameterization and uniformity of the input.

4.2.2 Evaluation Metrics

The evaluation must provide informative data with respect to DLBS interoperability. We use the number of tests passed by a DLBS without parsing errors is a metric of a system's conformance to the relevant syntax standard. The number of inference tests passed by a DLBS is a metric of a system's ability to perform the standard inference services. An inference test is counted as passed if the system result coincides with a "gold standard". In practice, the "gold standard" is either produced by a human expert or computed. In the latter case, the results of several systems should coincide in order to minimise the influence of implementation errors. Moreover, systems used to generate the "gold standard" should be believed to be sound and complete, and should be known to produce correct results on a wide range of inputs.

The evaluation must also provide informative data with respect to DLBS performance. The performance of a system is measured as the time the system needs to perform a given inference task. We also record task loading time to assess the amount of preprocessing used in a given system. It is difficult to separate the inference time from loading time given that some systems perform a great deal of reasoning and caching at load time, while others only perform reasoning in response to inference tasks. Thus, we account for both times reflecting the diversity in DLBS behaviour.

4.2.3 Evaluation Process

We evaluate both interoperability and performance for the standard inference services. Our evaluation infrastructure requires that systems either conform to the OWL API 3 [33] or implement API methods depicted on Table 4.1.

The output of an evaluation is the evaluation status. The evaluation status is one of the following TRUE, FALSE, ERROR, UNKNOWN. TRUE is returned if ontology and ontology class are satisfiable and in the case the entailment holds. FALSE is returned if ontology and ontology class are unsatisfiable and in the case the entailment does not hold. ERROR indicates a parsing error. UNKNOWN is returned if a system is unable to determine an evaluation results.

Table 4.1: DLBS interface methods

OWLOntology classifyOntology(OWLOntology ontology)	Classifies an ontology
boolean entails(OWLOntology premise, OWLOntology consequent)	Checks whether premise entails consequent
boolean isSatisfiable(OWLOntology ontology)	Checks ontology satisfiability
boolean isSatisfiable(OWLOntology ontology, OWLClass class)	Checks class satisfiability
OWLOntology loadOntology(IRI iri)	Loads ontology from IRI
IRI saveOntology(OWLOntology ontology, IRI iri)	Saves ontology to IRI

Class satisfiability evaluation

Class satisfiability is a standard inference service that is widely used in ontology engineering. The goal of class satisfiability evaluation is to assess both the interoperability and the performance of DLBSs in the standard class satisfiability tasks. Interoperability is assessed w.r.t. to both syntactic conformance and semantic correctness given that the result for each class satisfiability test is known. The ontology loading time and the satisfiability time are measured to assess performance. A class satisfiability evaluation input is an OWL 2 ontology and one or more class URIs. The result of a class satisfiability evaluation is the class satisfiability status, the ontology loading time, and the satisfiability tests times.

Ontology satisfiability evaluation

Ontology satisfiability is a standard inference service typically carried out before preforming any other reasoning task—if the ontology is unsatisfiable, most other reasoning tasks are meaningless/trivial. The ontology satisfiability evaluation goal is to assess both the interoperability and performance of DLBSs in the standard ontology satisfiability tasks. Interoperability is assessed w.r.t. to both syntactic conformance and semantic correctness given that the result for each ontology satisfiability test is known. The ontology loading time and the ontology satisfiability time is measured to assess performance. An ontology satisfiability evaluation input is an OWL 2 ontology. The result of an ontology satisfiability evaluation is the ontology's satisfiability status, the test ontology loading time, and the satisfiability test time.

Classification evaluation

Ontology classification is an inference service that is typically carried out after testing ontology satisfiability and prior to performing any other reasoning task. The goal of classification evaluation is to assess both the interoperability and the performance of DLBSs in the typical ontology classification tasks. Interoperability is assessed w.r.t. to both syntactic conformance and semantic correctness, i.e., if the class hierarchy produced by a system coincide with "gold standard" class hierarchy the classification process is counted as being correct. The ontology loading time and the classification

time is measured to assess performance. A classification evaluation input is an OWL 2 ontology. The result of a classification evaluation is the OWL 2 ontology, the ontology loading time and the ontology classification time. Classification quality metric is calculated using comparison with "gold standard". Classification hierarchy are extracted from resulting ontology and compared with "gold standard" classification hierarchy. If hierarchies are the same classification result is correct.

Logical entailment evaluation

Logical entailment is a standard inference service that is the basis for query answering. The goal of logical entailment evaluation is to assess both the interoperability and the performance of DLBSs in the typical logical entailment tasks. Interoperability is assessed w.r.t. to both syntactic conformance and semantic correctness given that for each logical entailment test the result is known. The ontologies loading times and the entailment time is measured to assess performance. A logical entailment evaluation input is two OWL 2 ontologies. The logical entailment evaluation status is produced as an output.

4.3 Testing data

Our collected data set contains most of the ontologies that are well established and widely used for testing DBLS's inference services. More precisely, it contains:

- The ontologies from the Gardiner evaluation suite. This suite now contains over 300 ontologies of varying expressivity and size. The test suite was originally created specifically for the purpose of evaluating the performance of ontology satisfiability and classification of DLBSs [30]. It has since been maintained and extended by the Oxford University Knowledge Representation and Reasoning group[2], and has been used in various other evaluations (e.g., [50]).
- The ontology repository described in [65] with more than 600 ontologies of diverse expressivity and size
- Various versions of the GALEN ontology [56]. The GALEN ontology is a large and complex biomedical ontology which has proven to be notoriously difficult for DL systems to classify, even for modern highly optimized ones. For this reason several "weakened" versions of GALEN have been produced by system developers in order to provide a subset of GALEN which some reasoners are able to classify.
- Various ontologies that have been created in EU funded projects, such as SEMINTEC, VICODI and AEO.

Table 4.2 presents some meta-data about the most important ontologies that we have managed to collect and include in our testing dataset. Since domain and range restrictions as well as functional number restrictions are usually translated into GCI

[2] http://web.comlab.ox.ac.uk/activities/knowledge/index.html

axioms this number is an aggregation of the number of explicitly defined GCIs in the ontology with the number of such axioms. Nevertheless, there are proposed optimizations in the literature that do not treat domain and range restrictions as GCIs but they perform a form of lazy unfolding [63].

Ontology	Classes	Roles	Indvs	SubClass	EqvCl	GCIs	Disj	RBox	ABox	DL
AEO	760	63	16	1,441	0	15	1,957	18	16	$\mathcal{SHIN}(\mathcal{D})$
NCI	66,725	190	0	85,335	10,280	261	171	278	0	$\mathcal{ALCH}(\mathcal{D})$
GALEN-full	23,141	950	0	25,563	9,968	674	0	1,436	0	$\mathcal{ALEHIF}_{R^+}$
GALEN-module1	6,362	162	0	10,901	3,614	59	0	160	0	$\mathcal{ALEHIF}_{R^+}$
GALEN-undoctored	2,748	413	0	3,480	699	514	0	649	0	$\mathcal{ALEHIF}_{R^+}$
GALEN-doctored	2,748	413	0	3,238	699	507	0	649	0	$\mathcal{ALEHIF}_{R^+}$
GALEN-horrocks	2,748	413	0	3,480	699	363	0	442	0	$\mathcal{ALEH}_{R^+}$
FMA	41,651	168	93	122,136	559	366	0	28	85	$\mathcal{ALCOIF}$
FMA-lite	75,145	3	46,225	119,558	0	0	0	3	46,225	$\mathcal{ALEI}_{R^+}$
GeneOntology	20,526	1	0	28,997	0	0	0	1	0	$\mathcal{ALE}_{R^+}$
Robert's Family	61	87	405	5	55	123	4	143	1,524	$\mathcal{SROIQ}(\mathcal{D})$
Food & Wine	138	17	207	228	88	31	39	9	494	$\mathcal{SHOIN}(\mathcal{D})$
SNOMED CT	304,802	62	0	514,061	59,182	0	0	12	0	$\mathcal{ALE}_{R^+}$

Table 4.2: Meta-data of the most important ontologies from the dataset

We use the OWL 2 conformance document as a guideline for conformance testing data. In particular, we aim at semantic entailment and non-entailment conformance tests. 148 entailment tests and 10 non-entailment tests from the OWL 2 test cases repository[3] are used for evaluating a DLBS's conformance.

4.4 Tools evaluated

The testing data was used for evaluation of three DLBSs HermiT 1.2.2 [4], FaCT++ 1.4.1 [5] and jcel 0.8.0 [6].

HermiT is a reasoner for ontologies written using the OWL [36]. HermiT is the first publicly-available OWL reasoner based on a novel hypertableau calculus [51] which provides efficient reasoning capabilities. HermiT can handle DL Safe rules and the rules can directly be added to the input ontology in functional style or other OWL syntaxes supported by the OWL API (see [6]).

FaCT++ [62] is the new generation of the well-known FaCT [34] OWL-DL reasoner. FaCT++ uses the established FaCT algorithms, but with a different internal architecture. Additionally, FaCT++ is implemented using C++ in order to create a more efficient software tool, and to maximise portability.

jcel is a reasoner for the description logic EL+. It is an OWL 2 EL [9] reasoner implemented in Java.

[3] http://owl.semanticweb.org/page/OWL_2_Test_Cases
[4] http://hermit-reasoner.com/
[5] http://owl.man.ac.uk/factplusplus/
[6] http://jcel.sourceforge.net/

4.5 Evaluation results

The evaluation has been run on two AMD Athlon(tm) 64 X2 Dual Core Processor 4600+ machines with 2GB of main memory. DLBSs were allowed to allocate up to 1 GB. To obtain the comparable results from both machines we executed a small subset of the class satisfiability tasks on them comparing the systems performance. The results have shown the influence of the software installed on the machines on the systems execution times. For example, average loading time (ALT) for class satisfiability tasks differ in 0.74 times for HermiT reasoner depending on the evaluation machine while average reasoning time (ART) differ in 1.29 times. Thus, we factored the results obtained to make them comparable. The systems had the 10 seconds evaluation time frame for single class satisfiability computation. More complex evaluations such as entailment and classification consists of many class satisfiability computations. Storage and reasoning systems evaluation component [68] has been used in the evaluation. All three systems support OWL API 3 [33]. The evaluation have been performed exploiting it as a common interface to DLBSs. Thus, the systems were run on the subset of the evaluation tasks that is OWL-API 3 parsable. OWL API 3.0 was used for evaluation. isSatisfiable(), isConsistent(), isEntailed() from OWLReasoner class were used for class satisfiability, ontology satisfiability and entailment evaluations. fillOntology method of InferredOntologyGenerator class initialized with InferredSubClassAxiomGenerator passed as list parameter was used for classification evaluation.

4.5.1 Classification

The evaluation results for classification evaluation are depicted in Table 4.3. It depicts ALT, ART, number of correct results (TRUE), number of incorrect results (FALSE), number of errors (ERROR), number of evaluation tasks that were not completed within the given time frame (UNKNOWN).

Table 4.3: Classification evaluation results

	FaCT++	HermiT	jcel
ALT, ms	68		856
ART, ms	15320		2144
TRUE	160		16
FALSE	0		0
ERROR	47		4
UNKNOWN	3		0

4.5.2 Class satisfiability

The evaluation results for class satisfiability evaluation are depicted in Table 4.4. It depicts ALT, ART, number of correct results (TRUE), number of incorrect results (FALSE), number of errors (ERROR), number of evaluation tasks that were not completed within the given time frame (UNKNOWN).

Table 4.4: Class satisfiability evaluation results

	FaCT++	HermiT	jcel
ALT, ms	1047	255	438
ART, ms	21376	517043	1113
TRUE	157	145	15
FALSE	1	0	0
ERROR	36	35	5
UNKNOWN	16	30	0

4.5.3 Ontology satisfiability

The evaluation results for ontology satisfiability evaluation are depicted in Table 4.5. It depicts ALT, ART, number of correct results (TRUE), number of incorrect results (FALSE), number of errors (ERROR), number of evaluation tasks that were not completed within the given time frame (UNKNOWN).

Table 4.5: Ontology satisfiability evaluation results

	FaCT++	HermiT	jcel
ALT, ms	1315		708
ART, ms	25175		1878
TRUE	134		16
FALSE	0		0
ERROR	45		4
UNKNOWN	31		0

4.5.4 Entailment

jcel system does not support entailment evaluation tasks. Therefore, we will provide the results for FaCT++ and HermiT. The evaluation results for entailment evaluation are depicted in Table 4.7. It depicts ALT, ART, number of correct results (TRUE), number of incorrect results (FALSE), number of errors (ERROR), number of evaluation tasks that were not completed within the given time frame (UNKNOWN).

Table 4.6: Entailment evaluation results

	FaCT++	HermiT
ALT, ms	21	81
ART, ms	3	34502
TRUE	6	55
FALSE	47	0
ERROR	11	9
UNKNOWN	0	0

The evaluation results for non entailment evaluation are depicted in Table 4.7. It depicts ALT, ART, number of correct results (TRUE), number of incorrect results

(FALSE), number of errors (ERROR), number of evaluation tasks that were not completed within the given time frame (UNKNOWN).

Table 4.7: Non entailment evaluation results

	FaCT++	HermiT
ALT, ms	53	92
ART, ms	7	127936
TRUE	5	7
FALSE	0	0
ERROR	3	1
UNKNOWN	0	0

4.6 Lessons learnt

4.6.1 Classification

The results for HermiT system was not obtained due to limited time allocated for Storage and Reasoning Systems Evaluation Campaign 2010. Most errors were related to the datatypes not supported in FaCT++ system. There were several description logic expressivity related errors such as NonSimpleRoleInNumberRestriction. There also were several syntax related errors where FaCT++ was unable to register a role or a concept.

4.6.2 Class satisfiability

FaCT++ clearly outperformed HermiT on the most of the reasoning tasks. Most errors for both FaCT++ and HermiT were related to the datatypes not supported in the systems. The evaluation tasks proved to be challenging enough for the systems. Thus, 16 and 30 evaluation tasks respectively were not solved in the given time frame. The relatively poor HermiT performance can be explained taking into account the small number of very hard tasks where FaCT++ was orders of magnitude more efficient.

4.6.3 Ontology satisfiability

The results for HermiT system was not obtained due to limited time allocated for Storage and Reasoning Systems Evaluation Campaign 2010. Most FaCT++ errors were related to not supported datatypes. There were several description logic expressivity related errors such as NonSimpleRoleInNumberRestriction. There also was several syntactic related errors where FaCT++ was unable to register a role or a concept.

4.6.4 Entailment

The HermiT time was influenced by small number of very hard tasks. FaCT++ demonstrated a big number of false and erroneous results.

The DL reasoners designed for less expressive subsets of the OWL 2 language not surprisingly demonstrated superior performance illustrating trade off between expressivity and performance. Most of the errors demonstrated by systems designed to work for more expressive language subsets were related to non supported language features. Further work includes the addition of the query answering evaluation and automatically generated datasets.

5. Ontology matching tools evaluation campaign

The SEALS Evaluation Campaign for Ontology Matching Tools has been coordinated with the Ontology Alignment Evaluation Initiative (OAEI) 2010 campaign. This chapter reports the evaluation results for this coordinated campaign, presenting the data sets and criteria used in the evaluation.

5.1 Previous evaluations

Since 2004, a group of researchers on ontology matching has organized annual evaluation campaigns for evaluating matching tools. This initiative is identified as Ontology Alignment Evaluation Initiative[1] (OAEI) campaigns. The main goal of the OAEI is to compare systems and algorithms on the same basis and to allow anyone for drawing conclusions about the best matching strategies.

OAEI had its first execution in 2004 and was realized together with two complementary events: (*i*) the Information Interpretation and Integration Conference (I3CON) held at the NIST Performance Metrics for Intelligent Systems (PerMIS) workshop and (*ii*) the Ontology Alignment Contest held at the Evaluation of Ontology-based Tools (EON) workshop of the annual International Semantic Web Conference (ISWC) [60]. In 2005, the campaign had its results discussed at the workshop on Integrating Ontologies held in conjunction with the International Conference on Knowledge Capture (K-Cap) [3]. From 2006, the OAEI campaigns are held at the Ontology Matching workshops collocated with ISWC [21, 20, 11, 18, 19].

In these campaigns, participants are invited to submit the results of their systems to organizers, who are responsible for running evaluation scripts and delivering the evaluation result interpretations. Since 2010, OAEI is being coordinated with the SEALS project and the plan is to integrate progressively the SEALS infrastructure within the OAEI campaigns. A subset of the OAEI tracks has been included in the new SEALS modality. Participants are invited to extend a web service interface[2] and deploy their matchers as web services, which are accessed in an evaluation experiment. This setting enables participants to debug their systems, run their own evaluations and manipulate the results immediately in a direct feedback cycle.

In this chapter, we report the evaluation results for the first OAEI/SEALS campaign. We describe the data sets and criteria used in this evaluation (§5.2) and detail how the evaluation itself was conducted (§5.3). Next, we present the participants of the campaign (§5.4) and summarize their results, per SEALS track (§5.5). Finally, we comment on the lessons learnt (§5.6) and conclude the chapter (§**??**).

[1] http://oaei.ontologymatching.org
[2] http://alignapi.gforge.inria.fr/tutorial/tutorial5/

5.2 Data sets and evaluation criteria

5.2.1 OAEI data sets

OAEI data sets have been extended and improved over the years. In OAEI 2010, the following tracks and data sets have been selected:

The benchmark test aims at identifying the areas in which each matching algorithm is strong and weak. The test is based on one particular ontology dedicated to the very narrow domain of bibliography and a number of alternative ontologies of the same domain for which alignments are provided.

The anatomy test is about matching the Adult Mouse Anatomy (2744 classes) and the NCI Thesaurus (3304 classes) describing the human anatomy. Its reference alignment has been generated by domain experts.

The conference test consists of a collection of ontologies describing the domain of organising conferences. Reference alignments are available for a subset of test cases.

The directories and thesauri test cases propose web directories (matching website directories like open directory or Yahoo's), thesauri (three large SKOS subject heading lists for libraries) and generally less expressive resources.

The instance matching test cases aim at evaluating tools able to identify similar instances among different data sets. It features web data sets, as well as a generated benchmark.

Anatomy, *Benchmark* and *Conference* have been included in the SEALS evaluation modality. The reason for this is twofold: on the one hand these data sets are well known to the organizers and have been used in many evaluations contrary to the test cases of the instance data sets, for instance. On the other hand these data sets come with a high quality reference alignment which allows for computing the compliance based measures, such as precision and recall. These data sets are detailed in the following. Full description of these tests can be found on the OAEI web site.[3]

5.2.2 Evaluation criteria and metrics

The diverse nature of OAEI data sets, specially in terms of the complexity of test cases and presence/absence of (complete) reference alignments, requires to use different evaluation measures. For the three data sets in the SEALS modality, compliance of matcher alignments with respect to the reference alignments is evaluated. In the case of *Conference*, where the reference alignment is available only for a subset of test cases, compliance is measured over this subset. The most relevant measures are precision (true positive/retrieved), recall (true positive/expected) and f–measure (aggregation of precision and recall). These metrics are also partially considered or approximated

[3] http://oaei.ontologymatching.org/2010/

for the other data sets which are not included in the SEALS modality (standard modality).

For *Conference*, alternative evaluation approaches have been applied. These approaches include manual labelling, alignment coherence [49] and correspondence patterns mining. These approaches require a more deep analysis from experts than traditional compliance measures. For the first version of the evaluation service, we concentrate on the most important compliance based measures because they do not require a complementary step of analyse/interpretation from experts, which is mostly performed manually and outside an automatic evaluation cycle. However, such approaches will be progressively integrated into the SEALS infrastructure.

Nevertheless, for 2010, the generated alignments are stored in the results database and can be retrieved by the organizers easily. It is thus still possible to exploit alternative evaluation techniques subsequently, as it has been done in the previous OAEI campaigns.

All the criteria above are about alignment quality. A useful comparison between systems also includes their efficiency, in terms of runtime and memory consumption. The best way to measure efficiency is to run all systems under the same controlled evaluation environment. In previous OAEI campaigns, participants have been asked to run their systems on their own and to inform about the elapsed time for performing the matching task. Using the web based evaluation service, runtime cannot be correctly measured due the fact that the systems run in different execution environments and, as they are exposed as web services, there are potential network delays.

5.3 Evaluation process

Once the data sets and evaluation criteria have been specified, the evaluation campaign takes place in four main phases:

Preparatory phase ontologies and alignments are provided to participants, which have the opportunity to send observations, bug corrections, remarks and other test cases;

Preliminary testing phase participants ensure that their systems can load the ontologies to be aligned and generate the alignment in the correct format (the Alignment API format [17]);

Execution phase participants use their algorithms to automatically match the ontologies;

Evaluation phase the alignments provided by the participants are evaluated and compared.

The four phases are the same for both standard and SEALS modality. However, different tasks are required to be performed by the participants of each modality. In the *preparatory phase*, the data sets have been published on web sites and could be downloaded as zip-files. In the future, it will be possible to use the SEALS portal to upload and describe new data sets. In addition, the test data repository supports

versioning, which is an important issue regarding bug fixes and improvements that have taken place over the years.

In the *phase of preliminary testing*, the SEALS evaluation service pays off in terms of reduced effort. In the past years, participants submitted their preliminary results to the organizers, who analyzed them semi-automatically, often detecting problems related to the format or to the naming of the required results files. Via a time-consuming communication process these problems have been discussed with the participants. It is now possible to check these and related issues automatically.

In the *execution phase*, standard OAEI participants run their tools on their own machines and submit the results via mail to the organizers, while SEALS participants run their tools via web service interfaces. They get a direct feedback on the results and can also discuss and analyse this feedback in their results paper[4]. Prior to the hard deadlines, for many of the data sets, results could not be delivered in the past to participants by the organizers in time.

Finally, in the *evaluation phase*, organizers are in charge of evaluating the received alignments. For the SEALS modality, this effort has been minimized due the fact that the results are automatically computed by the services in the infrastructure, as detailed in the next section.

5.4 Participants

The campaign had 15 participants in 2010 [19]: AgrMaker[5], AROMA[6], ASMOV[7], BLOOMS[8], CODI, Ef2Match, Falcon-AO[9], GeRoMeSMB, LNR2, MapPSO[10], NBJLM, ObjectRef, RiMOM[11], SOBOM and TaxoMap[12]. Regarding the SEALS tracks, 11 participants have registered their results for Benchmark, 9 for Anatomy and 8 for Conference. Some participants in Benchmark have not participated in Anatomy or Conference and vice-versa. The list of participants is summarized in Table 2. In this table, confidence stands for the type of result returned by a system: it is ticked when the confidence has been measured as non boolean value.

[4]Notice that each participant in the OAEI, independently of the modality, has to write a paper that contains a system description and an analysis of results from the point of view of the system developer.

[5]http://agreementmaker.org/

[6]http://aroma.gforge.inria.fr/

[7]http://infotechsoft.com/products/asmov.aspx

[8]http://wiki.knoesis.org/index.php/BLOOMS

[9]http://iws.seu.edu.cn/page/english/projects/matching/

[10]mappso.sourceforge.net/

[11]http://keg.cs.tsinghua.edu.cn/project/RiMOM/

[12]http://www.lri.fr/~hamdi/TaxoMap/TaxoMap.html

System	AgrMaker	AROMA	ASMOV	BLOOMS	CODI	Ef2Match	Falcon-AO	GeRoMeSMB	LNR2	MapPSO	NBJLM	ObjectRef	RiMOM	SOBOM	TaxoMap	Total=15
Confidence	√	√	√	√		√	√	√	√	√	√	√	√	√	√	
benchmarks	√	√	√		√	√	√	√		√			√	√	√	11
anatomy	√		√	√	√	√		√			√			√	√	9
conference	√	√	√		√	√	√	√						√		8
directory			√					√		√					√	4
iimb			√		√				√			√	√			5
Total	3	2	5	1	4	3	2	4	1	2	1	1	2	3	3	37

Table 5.1: Participants and the state of their submissions.

5.5 Evaluation results

5.5.1 Benchmark results

Eleven systems have participated in the benchmark track of this year's campaign (see Table 5.1). Table 5.2 shows the results, by groups of tests (harmonic means). Relaxed precision and recall correspond to the three measures of [16]: symmetric proximity, correction effort and oriented. The same results have been obtained using these three measures. Weighted precision and recall takes into account the confidence associated to correspondence by the matchers. We display the results of participants as well as those given by some simple edit distance algorithm on labels (edna). The full results are in [19].

As shown in Table 5.2, two systems achieve top performances: ASMOV and RiMOM, with AgrMaker as a close follower, while SOBOM, GeRMeSMB and Ef2Match, respectively, had presented intermediary values of precision and recall. In the 2009 campaign, Lily and ASMOV had the best results, with aflood and RiMOM as followers, while GeRoME, AROMA, DSSim and AgrMaker had intermediary performance. The same group of matchers has been presented in both campaigns. No system had strictly lower performance than edna.

In general, systems have improved their performance since last year: ASMOV and RiMOM improved their overall performance, AgrMaker and SOBOM significantly improved their recall, while MapPSO and GeRMeSBM improved precision. Only AROMA has significantly decreased in recall. There is no unique set of systems achieving the best results for all cases, which indicates that systems exploiting different features of ontologies perform accordingly to the features of each test case.

The results have also been compared with the relaxed measures proposed in [16], namely symmetric proximity, correction effort and oriented measures ("Relaxed measures" in Table 5.2). They are different generalisations of precision and recall in order to better discriminate systems that slightly miss the target from those which are grossly wrong. We have used strict versions of these measures (as published in [16] and contrary to previous years). As Table 5.2 shows, there is no improvement when comparing classical and relaxed precision and recall. This can be explained by the fact that participating algorithms miss the target, by relatively far (the false negative

system	refalign		edna		AgrMaker		AROMA		ASMOV		CODI		Ef2Match		Falcon		GeRMeSMB		MapPSO		RiMOM		SOBOM		TaxoMap	
test	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R
1xx	1.00	1.00	1.00	1.00	0.98	1.00	1.00	0.98	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.34
2xx	1.00	1.00	0.43	0.57	0.95	0.84	0.94	0.46	0.99	0.89	0.83	0.42	0.98	0.63	0.81	0.63	0.96	0.66	0.67	0.59	0.99	0.83	0.97	0.74	0.86	0.29
3xx	1.00	1.00	0.51	0.65	0.88	0.58	0.83	0.58	0.88	0.84	0.95	0.45	0.92	0.75	0.89	0.76	0.90	0.42	0.72	0.39	0.94	0.76	0.79	0.75	0.71	0.32
H-mean	1.00	1.00	0.45	0.58	0.95	0.84	0.94	0.48	0.98	0.89	0.84	0.44	0.98	0.65	0.82	0.65	0.96	0.67	0.68	0.60	0.99	0.84	0.97	0.75	0.86	0.29
Relaxed	1.00	1.00	0.45	0.58	0.95	0.84	0.94	0.48	0.99	0.89	0.84	0.44	0.98	0.65	0.82	0.65	0.96	0.67	0.68	0.60	0.99	0.84	0.97	0.75	0.86	0.29
Weighted	1.00	1.00	0.68	0.57	0.95	0.83	0.94	0.42	0.98	0.61	0.84	0.44	0.98	0.64	0.96	0.46	0.96	0.64	0.86	0.56	0.99	0.83	0.98	0.37	0.87	0.28

Table 5.2: Results obtained by participants on the benchmark test case.

correspondences found by the matchers are not close to the correspondences in the reference alignment) so the gain provided by the relaxed measures has no impact.

As last year, many algorithms provided their results with confidence measures. It is thus possible to draw precision/recall graphs in order to compare them. Figure 5.1 shows the precision and recall graphs of this year. These results are only relevant for the results of participants who provide confidence measures different from 1 or 0 (see Table 5.1). These graphs show the real precision at n% recall and they stop when no more correspondences are available (then the end point corresponds to the precision and recall reported in Table 5.2). The values are not anymore an average but a real precision and recall over all the tests. The numbers in the legend are the Mean Average Precision (MAP): the average precision for each correct retrieved correspondence. These new graphs represent well the effort made by the participants to keep a high precision in their results, and to authorize a loss of precision with a few correspondences with low confidence.

The results presented in Table 5.2 and those displayed in Figure 5.1 single out the same group of systems, ASMOV, RiMOM and AgrMaker, which perform these tests at the highest level. Out of these, ASMOV has slightly better results than the two others. So, this confirms the previous observations on raw results.

5.5.2 Anatomy results

Anatomy track[13] had in 2010 more systems that participated for the first time (5 systems) than in previous years (in average 2 systems). See Table 5.1 for the list of the participants in 2010.

As in previous years, the matching task was divided into four subtasks:

Subtask #1 The matcher has to be applied with its standard settings.

Subtask #2 An alignment has to be generated that favours precision over recall.

Subtask #3 An alignment has to be generated that favours recall over precision.

Subtask #4 A partial reference alignment has to be used as additional input.

Subtask #1 is compulsory for participants of the anatomy track, while subtask #2, #3 and #4 are again optional tasks. Detailed results on runtime can be found in [19].

Main results for subtask #1.

The results for subtask #1 are presented in Table 5.3 ordered with respect to the achieved F-measure. Systems marked with a * do not participate in other tracks or have chosen a setting specific to this track. Note that ASMOV modified its standard setting in a very restricted way (activating UMLS as additional resource). Thus, we did not mark this system.

In 2010, AgreementMaker (AgrMaker) generated the best alignment with respect to F-measure. This result is based on a high recall compared to the systems on the

[13]This section is a summary from the Antomy Results section in [19], which has been provided by Christian Meilicke.

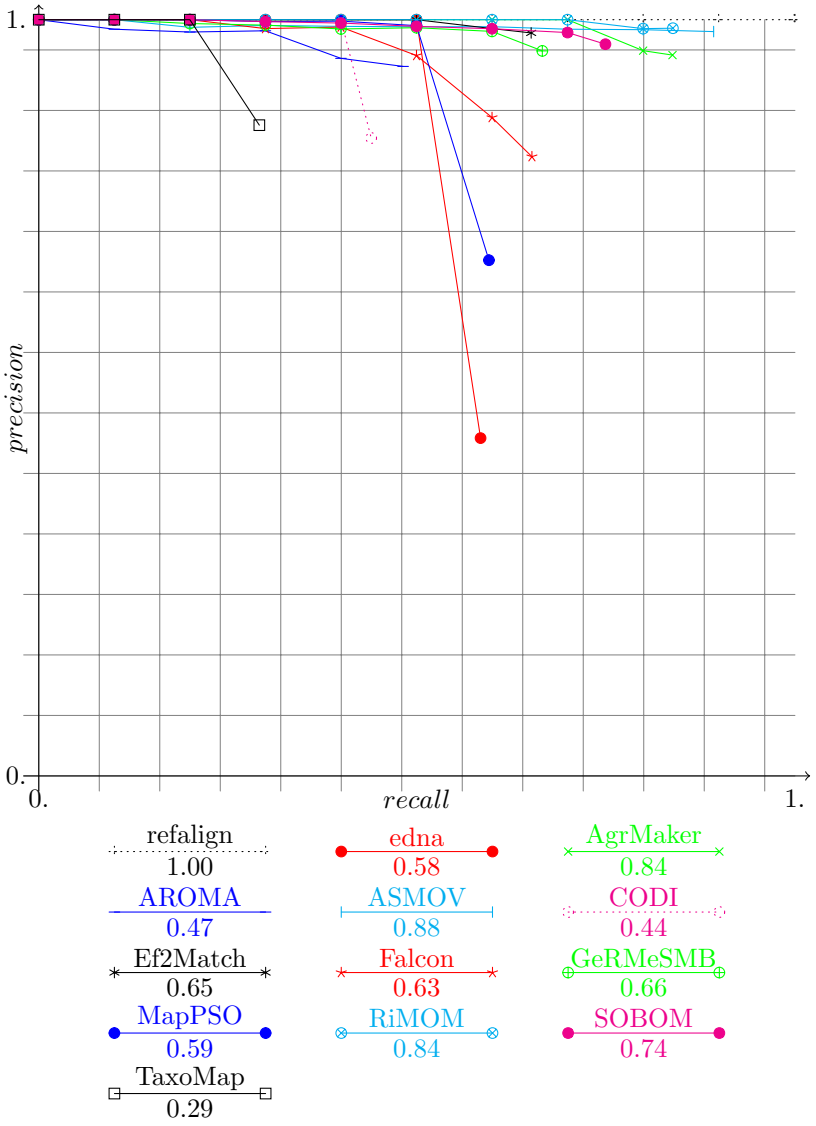

Figure 5.1: Precision/recall graphs for benchmarks.

	Task #1			Task #2			Task #3			Recall+	
System	**Prec.**	**F**	**Rec.**	**Prec.**	**F**	**Rec.**	**Prec.**	**F**	**Rec.**	**#1**	**#3**
AgrMaker*	0.903	0.877	0.853	0.962	0.843	0.751	0.771	0.819	0.874	0.630	0.700
Ef2Match	0.955	0.859	0.781	0.968	0.842	0.745	0.954	0.859	0.781	0.440	0.440
NBJLM*	0.920	0.858	0.803	-	-	-	-	-	-	0.569	-
SOBOM	0.949	0.855	0.778	-	-	-	-	-	-	0.433	-
BLOOMS	0.954	0.828	0.731	0.967	0.829	0.725	-	-	-	0.315	-
TaxoMap	0.924	0.824	0.743	0.956	0.801	0.689	0.833	0.802	0.774	0.336	0.414
ASMOV	0.799	0.785	0.772	0.865	0.808	0.757	0.717	0.753	0.792	0.470	0.538
CODI	0.968	0.779	0.651	0.964	0.785	0.662	0.782	0.736	0.695	0.182	0.383
GeRMeSMB	0.884	0.456	0.307	0.883	0.456	0.307	0.080	0.147	0.891	0.249	0.838

Table 5.3: Results for subtasks #1, #2 and #3 in terms of precision, F-measure, and recall (in addition recall+ for #1 and #3).

following positions. Even the SAMBO system of 2007 could not generate a higher recall with the use of UMLS.

AgreementMaker is followed by three participants (Ef2Match, NBJLM and SOBOM) that clearly favour precision over recall. Notice that these systems obtained better scores or scores that are similar to the results of the top systems in the previous years. One explanation can be seen in the fact that the organizers of the track made the reference alignment available to the participants. More precisely, participants could at any time compute precision and recall scores via the SEALS services to test different settings of their algorithms. This allows to improve a matching system in a direct feedback cycle, however, it might happen that a perfect configuration results in problems for different data sets.

Recall+ and further results.

We use again the recall+ measure as defined in [20]. It measures how many non trivial correct correspondences, not detectable by string equivalence, can be found in an alignment. The top three systems with respect to recall+ regarding subtask #1 are AgreementMaker, NBJLM and ASMOV. Only ASMOV has participated in several tracks with the same setting. Obviously, it is not easy to find a large amount of non-trivial correspondences with a standard setting.

In 2010, six systems participated in subtask #3. The top three systems regarding recall+ in this task are GeRoMe-SMB (GeRMeSMB), AgreementMaker and ASMOV. Since a specific instruction about the balance between precision and recall is missing in the description of the task, the results vary to a large degree. GeRoMe-SMB detected 83.8% of the correspondences marked as non trivial, but at a precision of 8%. AgreementMaker and ASMOV modified their settings only slightly, however, they were still able to detect 70% and 53.8% of all non trivial correspondences.

In subtask #2, seven systems participated. It is interesting to see that systems like ASMOV, BLOOMS and CODI generate alignments with slightly higher F-measure for this task compared to the submission for subtask #1. The results for subtask #2 for AgreementMaker are similar to the results submitted by other participants for subtask #1. This shows that many systems in 2010 focused on a similar strategy that exploits the specifics of the data set resulting in a high F-measure based on a high precision.

System	Δ-Precision	Δ-F-measure	Δ-Recall
AgrMaker	+0.025 $_{0.904 \rightarrow 0.929}$	−0.002 $_{0.890 \rightarrow 0.888}$	−0.025 $_{0.876 \rightarrow 0.851}$
ASMOV	+0.029 $_{0.808 \rightarrow 0.837}$	+0.006 $_{0.816 \rightarrow 0.822}$	−0.016 $_{0.824 \rightarrow 0.808}$
CODI	−0.002 $_{0.970 \rightarrow 0.968}$	+0.019 $_{0.824 \rightarrow 0.843}$	+0.030 $_{0.716 \rightarrow 0.746}$
SAMBOdtf$_{2008}$	+0.021 $_{0.837 \rightarrow 0.856}$	+0.011 $_{0.852 \rightarrow 0.863}$	+0.003 $_{0.867 \rightarrow 0.870}$

Table 5.4: Changes in precision, F-measure and recall based on comparing $A_1 \cup R_p$ and A_4 against reference alignment R.

Subtask #4.

In the following, we refer to an alignment generated for subtask #n as A_n. In our evaluation we use again the method introduced in 2009. We compare both $A_1 \cup R_p$ and $A_4 \cup R_p$ with the reference alignment R.[14] Thus, we compare the situation where the partial reference alignment is added after the matching process against the situation where the partial reference alignment is available as additional resource exploited within the matching process. Note that a direct comparison of A_1 and A_4 would not take into account in how far the partial reference alignment was already included in A_1 resulting in a distorted interpretation.

Results are presented in Table 5.4. Three systems participated in task #4 in 2010. Additionally, we added a row for the 2008 submission of SAMBOdtf. This system had the best results measured in the last years. AgreementMaker and ASMOV use the input alignment to increase the precision of the final result. At the same time these systems filter out some correct correspondences, finally resulting in a slightly increased F-measure. This fits with the trend observed in the past years (compare with the results for SAMBOdtf in 2008). The effects of this strategy are not very strong. However, as argued in previous years, the input alignment has a characteristics that makes hard to exploit this information. CODI has chosen a different strategy. While changes in precision are negligible, recall increases by 3%. Even though the overall effect is still not very strong, the system exploits the input alignment in the most effective way. However, the recall of CODI for subtask #1 is relatively low compared to the other systems. It is unclear whether the strategy of CODI would also work for the other systems where a ceiling effect might prevent the exploitation of the positive effects.

5.5.3 Conference results

Eight systems have participated in the conference track (Table 5.5), whose results are presented below. In this chapter, we focus on the evaluation based on the reference alignments and coherence, while the results of using other evaluation approaches (i.e., manual labelling and matching patterns) can be found in [19].

[14] We use $A_4 \cup R_p$ – instead of using A_4 directly – to ensure that a system, which does not include the input alignment in the output, is not penalized.

matcher	confidence threshold	Prec.	FMeas.	Rec.
AgrMaker	0.66	.53	.58	.62
AROMA	0.49	.36	.42	.49
ASMOV	0.22	.57	.60	.63
CODI	*	.86	.62	.48
Ef2Match	0.84	.61	.60	.58
Falcon	0.87	.74	.59	.49
GeRMeSMB	0.87	.37	.43	.51
SOBOM	0.35	.56	.56	.56

Table 5.5: Confidence threshold, precision and recall for optimal F-measure for each matcher.

Evaluation based on the reference alignments.

We evaluated the results of participants against reference alignments (all pairwise combinations between 7 different ontologies, i.e. 21 alignments). For a better comparison, we established the confidence threshold which provides the highest average F-measure (Table 5.5). Precision, recall, and F-measure are given for this optimal confidence threshold. The dependency of F-measure on the confidence threshold can be seen from Figure 5.2. There is one asterisk in the column of confidence threshold for matcher CODI which did not provide graded confidence.

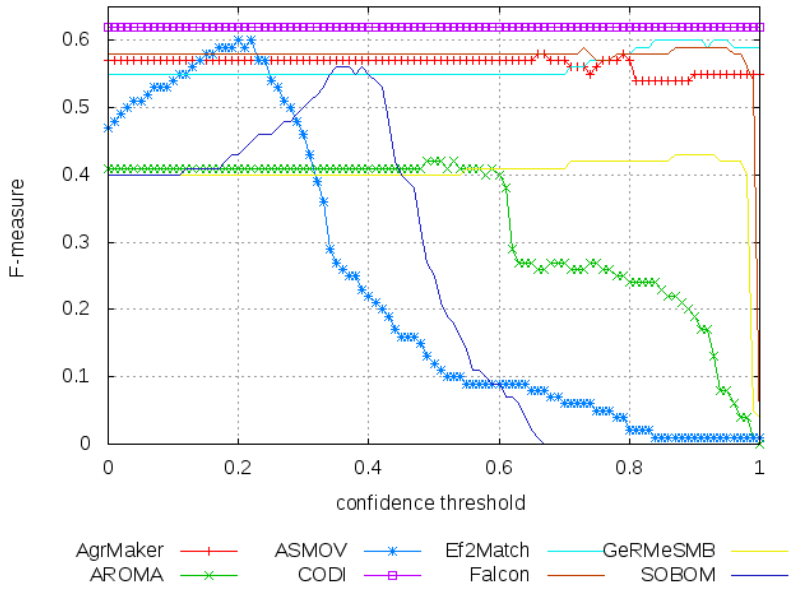

Figure 5.2: F-measures depending on confidence.

In conclusion, the matcher with the highest average F-measure (62%) is CODI which did not provide graded confidence values. Other matchers are very close to this score (e.g. ASMOV with F-Measure 0.60, Ef2Match with F-Measure 0.60, Falcon with F-Measure 0.59). However, we should take into account that this evaluation has been made over a subset of all alignments (one fifth).

Matcher	AgrMaker	AROMA	ASMOV	CODI	Ef2Match	Falcon	GeRMeSMB	SOBOM
Max-Card %	>14.8%	>17.5%	5.6%	0.1%	7.2%	>4.8%	>12.6%	>10.7
N	17.1	16.4	18.2	6.9	12.8	8.9	18.2	11.7

Table 5.6: Degree of incoherence and size of alignment in average for the optimal a posteriori threshold.

Evaluation based on alignment coherence.

In the following we apply the Maximum Cardinality measure as proposed in [49] to measure the degree of alignment incoherence. Results are depicted in Table 5.6 which shows the average for all test cases of the conference track except the test cases where the ontologies confious and linklings are involved (the prefix > is added whenever the search algorithm stopped in one of the test case due to a timeout of 1000 seconds prior to finding the solution). These ontologies resulted in some combinations of ontologies and alignments in reasoning problems. Note that we did not use the original alignments, but the alignments with optimal threshold. However, the average size of the resulting alignment still varies to a large degree.

Compared to the other participants CODI generates the lowest degree of incoherence. This result is partially caused by the small size of alignments that make the occurrence of an incoherence less probable. Taking this into account, the ASMOV system achieves a remarkable result. Even though the alignments of ASMOV comprise the highest number of correspondences, the degree of incoherence 5.6% is relatively small due to the verification component built into the system [38]. Overall it is a surprising result that still only few matching systems take alignment incoherence into account.

5.6 Lessons learnt

In the following, we highlight some of the outcomes from the experiences made with this campaign.

OAEI continuity. We were not sure that switching to an automated evaluation would preserve the success of OAEI, given that the effort of implementing a web service interface was required from participants. This has been the case. The number of participants is similar to the numbers we observed in the last years. Moreover, the conference track has more participants than in the last years. This might be caused by the fact that many participants mainly focus on participating in the benchmark track. However, once the interface is implemented it is just one additional mouse click to participate also in the conference track.

Implementing the interface. Implementing the web service interface requires some effort on side of a tool developer. We stayed in contact with some of the tool developers during this process. Thereby, we observed that the time required for implementing the interface varied between several hours and several days depending on the technical

skills of each developer. We also became aware that the first version of the provided tutorial contained some unclear information resulting in problems for some participants. From the feedback of the developers, we have improved the tutorial. Another typical problem is related to the fact that some tool developer had only restricted access to a machine that is available from the Internet. These problems could finally be solved, however, system administrators of the particular company or research institute should be contacted early by participants.

Usage by participants. Once technical problems had been solved, the evaluation service has been used by some of the participants in the phase of preliminary testing extensively. Obviously, the direct feedback of the evaluation service has supported the process of a formative evaluation well. Other participants used the service only for submitting their final results. This might have been caused partially by a suboptimal runtime performance during the first weeks. Even though we finally solved the underlying problems, these problems might have been the reason for some participants to abandon from the use during the first weeks. Once the problems have been solved, we contacted each participant in order to explain the problems and many of them started to use the system again.

Organizational effort. On side of the organizers, the evaluation service reduced the effort of checking the formal correctness of the results to a large degree. In the past, it was required to communicate many of the problems in a time-consuming multilevel process. Typical examples are invalid xml, missing or incorrect namespace information, unsupported types of relations in generated alignments, incorrect directory structure and an incorrect naming style used for the submissions. All of these problems are now directly forwarded to the tool developer in an error message or in a preliminary result interpretation that does not fit with the expectations. Moreover, the organizers could analyze the results so far submitted at any time and had an overview on the participants using the system.

Evaluation and analysis services. While some analysis methods are already available, a number of specific services and operations is still missing. The graphical support of the OLAP visualization does, for example, not support the generation of precision and recall graphs frequently used by OAEI organizers. In particular, evaluation and visualization methods specific for ontology matching are not supported. However, most of these operations are already implemented in the Alignment API and will be made available in the future. On the other hand we already developed and tested a component for measuring the degree of incoherence of an alignment. We will try to include this additional metric in the next version of the SEALS platform.

Automatic test generation. The benchmark test case is not discriminant enough between systems. The results presented above have shown this. Next year, we plan to introduce controlled automatic test generation in the SEALS platform. This will improve the situation and allow OAEI organizers and SEALS campaign organizers to construct test suites with the required type of test cases.

Configuration and control over matching systems. We have seen that not all systems followed the general rule to use the same set of parameters in all tracks. This problem will be solved when we deploy the systems in the SEALS platform in the following campaign. However, we also have to support different system configurations in a controlled environment. To run a system with specific settings is an explicit requirement of subtask #2 and #3 of the anatomy track. It should thus be possible to run a system deployed on the SEALS platform with different parameter setting.

The new technology introduced in the OAEI affected both tool developers and organizers to a large degree and has been accepted positively on both sides. For the next campaign, we plan to measure runtime and memory consumption, which cannot be correctly measured because a controlled execution environment is missing. The same holds for the reproducibility of the results. We also plan to integrate additional metrics and visualization components. Finally, we will try to find more well suited data sets to be used as test suites in the platform, what includes the development of a test generator that allows a controlled automatic test generation of high quality data sets.

6. Semantic search tools evaluation campaign

The SEALS Semantic Search Tool Campaign followed a 'two phase' approach for evaluating search tool usability and performance. This chapter introduces the criteria of interest followed by the associated metrics and the test sets used in each phase. Finally, we provide an overview of the key results followed by a detailed discussion of the usability findings.

6.1 Introduction

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) 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 perceived performance and usability of the tool. This highlights the need for a formalised and consistent evaluation which is capable of dealing with this diversity. It is essential that we do not forget that searching is a user-centric process and that the evaluation mechanism should capture the usability of a particular approach.

In previous evaluation efforts, Kaufmann evaluated four approaches to querying ontologies [39]. Three were based on natural language input (with one employing a restricted query formulation grammar); the fourth employed a formal query approach which was hidden from the end user by a graphical query interface. A comprehensive usability study was conducted which focused on comparing the different query languages employed by the tools. It was shown that users preferred approaches based around full natural language sentences to all other formats and interfaces. It was also noted that users favour query languages and interfaces in which they can naturally communicate their information need without restrictions on the grammar used or having to rephrase their queries. Users were also found to express more semantics (e.g., relations between concepts) using full sentences rather than keywords.

Another work evaluated a "hybrid search" approach [7]. Their search approach consisted of an intelligent combination of keyword-based search and semantically motivated knowledge retrieval. To assess the effectiveness and the performance of the approach, an *in vitro* evaluation was conducted to compare it against keyword-based alone and ontology-based alone searching approaches. Additionally, the authors conducted an *in vivo* evaluation which involved 32 subjects who gave their opinion and comments regarding the efficiency, effectiveness, and satisfaction of the system. In both cases, the hybrid approach was observed to be superior.

The goal of the evaluation was to create a consistent and standard evaluation that can be used for assessing and comparing the strengths and weaknesses of Semantic Search approaches. This allows tool adopters to select appropriate tools and technologies for their specific needs and helps developers identify gaps and limitations with

their own tools which will facilitate improving them. Furthermore, the evaluation outcomes identify new requirements of search approaches with the aim of more closely matching users' needs.

6.2 Evaluation design

6.2.1 Two-phase approach

The evaluation of each tool is split into two complementary phases: the Automated Phase and the User-in-the-loop Phase. The user-in-the-loop phase comprises a series of experiments involving human subjects who are given a number of tasks (questions) to solve and a particular tool and ontology with which to do it. The subjects in the user-in-the-loop experiments are guided throughout the process by bespoke software – the *controller* – which is responsible for presenting the questions and gathering the results and metrics from the tool under evaluation. Two general forms of metrics are gathered during such an experiment. The first type of metrics are directly concerned with the operation of the tool itself such as time required to input a query, and time to display the results. The second type is more concerned with the 'user experience' and is collected at the end of the experiment using a number of questionnaires.

The outcome of these two phases will allow us to benchmark each tool both in terms of its raw performance but also the ease with which the tool can be used. Indeed, for semantic search tools, it could be argued that this latter aspect is the most important. In addition to usability questionnaires, demographics data will be collected from the subjects enabling tool adopters to assess whether a particular tool is suited for their target user group(s).

6.2.2 Criteria

Scalability

The recent emergence of linked data and the trend of publishing open datasets not only increased the amount of semantically annotated data on the Web but also changed a core characteristic of the queried ontologies and datasets: their size. Heterogeneous datasets such as dbpedia[1] or domain specific ones such as GeoNames[2] are commonly characterised by their significant sizes; GeoNames contains over eight million place names. This highlights the need to test the ability of semantic search approaches to scale over large data sets. This includes the ability to query large repositories in reasonable time; the ability to cope with a large ontology; and the ability to cope with a large amount of results returned in terms of readability/accessibility of those results. This is assessed in the automated phase.

[1] http://dbpedia.org/
[2] http://www.geonames.org/

Usability of Input Style

Different input styles (e.g., form-based, natural language, etc.) can be compared with respect to the input query language's expressiveness and usability. These related concepts are assessed by capturing feedback regarding the user experience and assess the usefulness of the query language in supporting the user to express their information needs and formulate searches [64]. Additionally, the expressive power of a query language specifies what queries a user is able to pose [2]. Any data sets and associated questions must be designed to fully investigate these issues.

Performance

Users are familiar with the performance of commercial search engines (e.g., Google) in which results are returned within fractions of a second; therefore, it is a core criterion to measure the tool's performance with respect to the speed of execution. In addition, measures such as the precision and recall of the results returned inform us about the quality of the underlying search technique. Performance is assessed in both phases, although the automated phase is the dominant phase.

6.2.3 Metrics and Analyses

In addition to 'standard' metrics such as the result set and the time required to execute a query, a number of phase-specific metrics are also collected (see [67] for more details).

For the *automated phase*, we captured the success/failure of loading an ontology by a tool. On successful loading, we store the actual set of results returned by the tool in response to a query and the time required to return this set of results. We also check for more general error conditions related to the tool throughout the evaluation. The scalability criterion is assessed by both examining the success/failure of the tool to load a given ontology as well as examining the average time to execute a query with respect to the ontology size. Tool robustness is represented by the ratio between the number of tests executed and the number of failed executions. Speed of response is calculated based on the average time required by the tool to execute the given queries. Finally, performance measures such as precision, recall and f-measure are calculated using the set of results returned by the tool in response for a query.

For the *user-in-the-loop phase*, we captured if the user could, in their opinion, find a satisfactory answer to their query and the number of attempts they made to obtain that answer. We also retained user-specific statistics such as the time required to formulate the query. Data regarding the user experience and satisfaction with the tool was collected using questionnaires (see Sec. 6.2.4). Performance measures including the precision, recall and also the speed of response are evaluated in a similar way as in the automated phase. Usability and the expressiveness of the input query language adopted were evaluated based on the user-specific statistics together with the responses of the questionnaires.

6.2.4 Questionnaires

Three different types of questionnaires are used in the user-in-the-loop phase which serve different purposes. The first is the System Usability Scale (SUS) questionnaire [8]. The test consists of ten normalized questions and covers a variety of usability aspects, such as the need for support, training, and complexity and has proven to be very useful when investigating interface usability [4].

We developed a second, extended, questionnaire which includes further questions regarding the satisfaction of the users. This encompasses the design of the tool, the input query language, the tool's feedback, and the user's emotional state during the work with the tool. An example of a question used is *'The query language was easy to understand and use'* with answers represented on a scale from 'disagree' to 'agree'. Finally, a demographics questionnaire collected information regarding the participants.

6.3 Datasets and Questions

For the first evaluation campaign we have taken the decision to focus on purely ontology-based tools. More complex test data (document-based, chaotic data, data with partially known schemas) will be considered for later evaluation campaigns. Indeed, the SEALS consortium actively encourages community participation in the specification of subsequent campaigns.

The Automated Phase used EvoOnt[3]. This is a set of software ontologies and data exchange format based on OWL. It provides the means to store all elements necessary for software analyses including the software design itself as well as its release and bug-tracking information. For scalability testing it is necessary to use a data set which is available in several different sizes. In the current campaign, it was decided to use sets of sizes 1k, 10k, 100k, 1M, 10M triples. The EvoOnt data set lends itself well to this since tools are readily available which enable the creation of different ABox sizes for a given ontology while keeping the same TBox. Therefore, all the different sizes are variations of the same coherent knowledge base.

The test questions for the automated phase ranged in their level of complexity including simple ones like "Does the class x have a method called y?" and more complex ones like "Give me all the issues that were reported in the project x by the user y and that are fixed by the version z?". The inspiration for the question templates was taken from [12], [57] and [58]. The groundtruth SPARQL queries were generated manually.

The main requirement for the user-in-the-loop dataset is that it be from a simple and understandable domain: it should be sufficiently simple and well-known that casual end-users are able to reformulate the questions into the respective query language without having trouble to understand them. Additionally, a set of questions are required which subjects will use as the basis of their input to the tool's query language or interface. The Mooney Natural Language Learning Data[4] fulfils these requirements and is comprised of three data sets each supplying a knowledge base, English questions, and corresponding logical queries. They cover three different domains: geographical

[3] http://www.ifi.uzh.ch/ddis/evo/
[4] http://www.cs.utexas.edu/users/ml/nldata.html

data, job data, and restaurant data. We chose to apply only the geography data set, because it defines data from a domain immediately familiar to casual users. The geography OWL knowledge base contains 9 classes, 11 datatype properties, 17 object properties and 697 instances. An advantage of using the Mooney data for the user-in-the-loop evaluation is the fact that it is a well-known and frequently used data set (e.g., [39], [55] and [61]). Furthermore, its use allowed the possibility of making the findings comparable with other evaluations of tools in this area, such as *Cocktail* [61], *PANTO* [55] and *PRECISE* [55].

The complete set of questions for each dataset can be found in [66].

6.4 Participants

The list of the participants and the phases in which they participated is shown in Table 7.2. The last two columns in the table indicate whether the tool participated in the user-in-the-loop (UITL) and / or the automated (Auto) phase.

Table 6.1: Tools which participated in the semantic search evaluation campaign.

Tool	Description	UITL	Auto
K-Search	K-Search allows flexible searching of semantic concepts in ontologies and documents using a form-based interface. http://www.k-now.co.uk/	x	x
Ginseng	Guided Input Natural Language Search Engine (Ginseng) is a natural language interface based question answering system that restricts the user input via word lists extracted from the underlying ontology. http://www.ifi.uzh.ch/ddis/research/talking-to-the-semantic-web/ginseng/	x	x
NLP-Reduce	NLP-Reduce is a natural language query interface that allows its users to enter full English questions, sentence fragments and keywords. http://www.ifi.uzh.ch/ddis/research/talking-to-the-semantic-web/nlpreduce/	x	x
Jena Arq 2.8.2	Arq is a query engine for Jena that supports the SPARQL RDF Query language. This tool has been used as a 'baseline' for the automated phase. http://jena.sourceforge.net/ARQ/		x
PowerAqua	PowerAqua is an open multi-ontology Question Answering (QA) system for the Semantic Web (SW) using a Natural Language (NL) user interface. http://technologies.kmi.open.ac.uk/poweraqua/	x	x

All registered tools attempted to run the automated phase of the evaluation campaign. The Jena Arq tool did not participate in the user-in-the-loop phase because this tool was only included in the campaign to act as a baseline within the automated phase. Jena Arq does not have a user interface: it only provides programmatic access to the underlying SPARQL query engine.

6.5 Results

This section presents the results and analyses from the first SEALS Evaluation Campaign which was conducted during Summer 2010.

6.5.1 Automated Phase

The automated evaluation results are shown in Table 6.2. In order to facilitate the discussion, the responses to each of the ten questions per dataset size have been averaged. Also, due to space constraints, we include the results of the the smallest and largest datasets only. Full results and analyses can be found in [66].

Table 6.2: Semantic search tool performances on the automated scenario.

	Arq	K-Search	PowerAqua	Ginseng	NLP-Reduce
EvoOnt 1k					
Load successful	true	false	true	true	false
Load time (ms)	1516.0	-	4522.0	43919.0	-
Num queries completed	10	-	10	10	-
Mean query time (ms)	88.4	-	1593.1	28.8	-
Mean num results	5	-	19	14	-
Mean time / results	17.68	-	83.85	2.06	-
Mean precision	1	-	0.29	0.66	-
Mean recall	1	-	0.8	0.97	-
Mean f-measure	1	-	0.42	0.78	-
EvoOnt 10M					
Load successful	true	false	true	false	false
Load time (ms)	473367.0	-	1234.0	-	-
Num queries completed	10	-	6	-	-
Mean query time (ms)	4442.3	-	82625.67	-	-
Mean num results	21844	-	5943	-	-
Mean time / results	0.2	-	13.9	-	-
Mean precision	1	-	0	-	-
Mean recall	1	-	0.17	-	-
Mean f-measure	1	-	0	-	-

The most unexpected outcome of the automated phase was the failure of many of the participating tools to load even the smallest ontology. The EvoOnt ontologies have certain interesting characteristics which, although complex, are valid and commonly

Table 6.3: Semantic search tool performance on the user-in-the-loop scenario.

Criterion	K-Search	Ginseng	NLP-Reduce	PowerAqua
Mean experiment time (s)	4313.84	3612.12	4798.58	2003.9
Mean SUS (%)	44.38	40	25.94	72.25
Mean ext. questionnaire (%)	47.29	45	44.63	80.67
Mean number of attempts	2.37	2.03	5.54	2.01
Mean answer found rate	0.41	0.19	0.21	0.55
Mean execution time (s)	0.44	0.51	0.51	11
Mean input time (s)	69.11	81.63	29	16.03
Max input time (s)	300.17	300.16	278.95	202.82
Mean overall question time (s)	257.25	216.19	246.95	109.51
Mean precision	0.44	0.32	0.16	0.57
Mean recall	0.61	0.32	0.55	0.68
Mean f-measure	0.46	0.27	0.21	0.57

found on the Semantic Web. One of these characteristics is importing of external ontologies from the web. Also, the ontologies include orphan object and datatype properties, and finally some concepts have cyclic relations with concepts in remote ontologies. This informs the tools' scalability, conformance with standards and suitability to the Semantic Web. Unfortunately, many of the participating tools were not able to cope with these standards. Such interoperability issues are critical to the wide adoption of any search approach and associated tool.

6.5.2 User-in-the-Loop Phase

Table 6.3 shows the results for the four tools participating in this phase. The *mean number of attempts* shows how many times the user had to reformulate their query using the tool's interface in order to obtain answers with which they were satisfied (or indicated that they were confident a suitable answer could not be found). This latter distinction between finding the appropriate answer after a number of attempts and the user 'giving up' after a number of attempts is shown by the *mean answer found rate*. *Input time* refers to the amount of time the subject spent formulating their query using the tool's interface, which acts as a core indicator of the tool's usability.

Here, we focus solely on usability (see [66] for more detailed results analysis). According to the ratings of SUS scores [5], none of the four participating tools fell in either the best or worst category. Only one of the tools had a 'Good' rating with a SUS score of 72.25, other two tools fell in the 'Poor' rating while the last one was classified as 'Awful'.

The results of the questionnaires were confirmed by the recorded usability measures. Subjects using the tool with the lowest SUS score required more than twice the number of attempts of the other tools before they were satisfied with the answer or moved on. Similarly, subjects using the two tools with the highest SUS and extended scores found satisfying answers to their queries twice the times as for the other tools. Altogether,

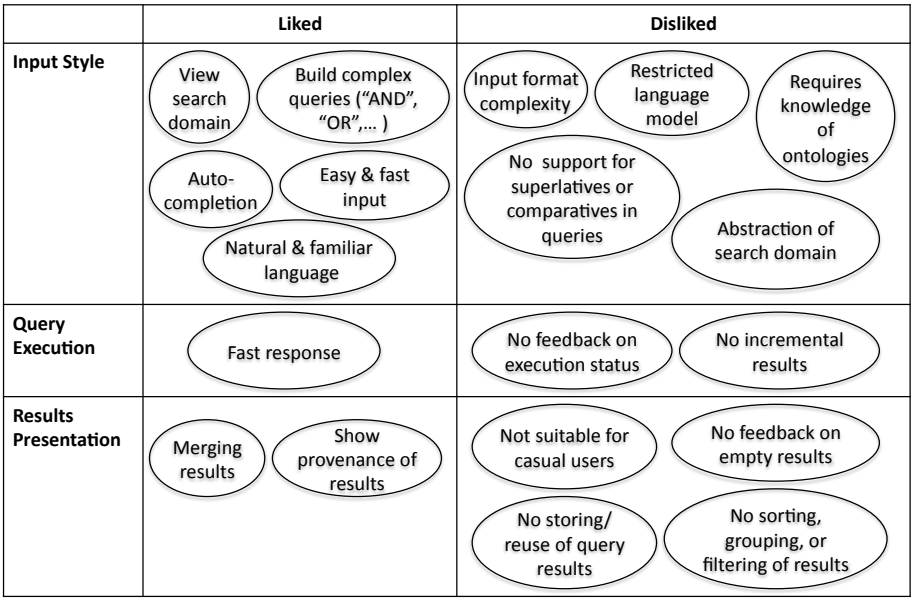

Figure 6.1: Summary of the semantic search evaluation feedback.

this confirms the reliability of the results and the feedback of the users and also the conclusions based on them.

6.6 Usability Feedback and Analysis

This section discusses the results and feedback of the user-in-the-loop phase. Figure 6.1 summarises the features most liked and disliked by users based on their feedback.

6.6.1 Input Style

Uren et al. [64] state that forms can be helpful to explore the search space when it is unknown to the users. However, Lei et al. [42] see this exploration as a burden on users that requires them to be (or become) familiar with the underlying ontology and semantic data. We found that form-based interfaces allow users to build more complex queries than the natural language interfaces. However, building queries by exploring the search space is usually time consuming especially as the ontology gets larger or the query gets more complex [39]. Our analysis suggests a more nuanced behaviour. While freeform natural language interfaces are generally faster in terms of query formulation, we found this did not hold for approaches employing a very restricted language model. This is further supported by user feedback in which it was reported that they would prefer typing a natural language query because it is faster than forms or graphs.

Although natural language interfaces are often preferred [39] and offer more expressivity [14], such approaches suffer from both syntactic as well as semantic ambiguities. This makes the overall performance of such approaches heavily dependent upon the performance of the underlying natural language processing techniques responsible for

parsing and analysing the users' natural language sentences. This was supported both by our evaluation's user feedback as well as the lowest precision scores.

Using a restricted grammar as employed by Ginseng is an approach to limit the impact of these problems. The 'autocompletion' provided by the system based on the underlying grammar attempts to bridge the domain abstraction gap and also resembles the form-based approach in helping the user to better understand the search space. However, it still lacks the power of visualising the structure of the used ontology. The impact of this 'intermediate' functionality can be observed in the user feedback with a lower degree of dissatisfaction regarding the ability to conceptualise the underlying data but still not completely eliminated. The restricted language model also prevents unacceptable/invalid queries in the used grammar by employing a guided input natural language approach. However, only accepting specific concepts and relations – found in the grammar – limits the flexibility and expressiveness of the user queries. User coercion into following predefined sentence structures proves to be frustrating and too complicated [67, 39].

The results of our evaluation showed that using superlatives or comparatives in the user queries was not supported by any of the participating tools which was disappointing for the users.

6.6.2 Processing feedback

The lack of feedback on the status of the execution process increased the sense of dissatisfaction: no tool indicated the execution progress or whether a problem had occurred in the system. This lack of feedback resulted in users suspecting that something had gone wrong with the system – even if the search was progressing as normal – and try to start a new search. Furthermore, some tools made it impossible to distinguish between an empty result set, a problem with the query formulation or a problem with the search. This not only affected the users experience and satisfaction but also the approach's measured performance.

6.6.3 Results Presentation

Semantic Search tools tend to be used by casual users (i.e., users who may be experts in the domain of the underlying data but may have no knowledge of semantic technologies). Such users usually have specific requirements and expectations of *what* and *how* results should be presented to them.

In contrast, a number of the search tools did not present their results in a user-friendly manner (collections of URIs or difficult to interpret instance labels) and this was reflected in the feedback. Although potentially useful to an expert in the semantic web field, this was not helpful to casual users.

The other commonly reported limitation of all the tools was the degree to which a query's results could be stored or reused. Users often wanted to use previous results as the basis of a further query or to temporarily store the results in order to perform an intersection or union operation with the current result set. Unfortunately, this was not supported by any of the participating tools. Another means of managing the results that users requested was the ability to filter results according to some

suitable criteria and checking the provenance of the results; only one tool provided the latter. Indeed, even basic manipulations such as sorting were requested – a feature of particular importance for tools which did not allow query formulations to include superlatives.

There is still work to be done to ensure semantic search tools can load or use as wide a range of ontologies and data sets as possible. The usability phase identified a number of features that end users would like: a hybrid approach to browsing the ontology and creating queries which would combine both a visual representation of the underlying ontology and natural language input; better feedback regarding the processing state of the tool (e.g., to distinguish between a query failure and an empty result set); improved result set management (sorting, filtering, ability to use as the target of a subsequent query, etc.) and the inclusion of 'related' information (possibly drawn from other data sets).

7. Semantic web service tools evaluation campaign

The SEALS Evaluation Campaign for Semantic Web Service (SWS) Tools consisted of the Discovery scenario. This was an experimental evaluation that included existing tools and datasets available from previous initiatives. We have also provided a common evaluation SWS API and integrated a number of evaluation metrics in our implementation.

7.1 Introduction

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 provide a layer of semantics for service interoperability by relying on a number of reference service ontologies (e.g., OWL-S[1], WSMO[2] , WSMO-Lite[3]) and semantic annotation extension mechanisms (e.g., SAWSDL[4], SA-REST[5], MicroWSMO[6]). The work performed in SEALS regarding SWS tools is based upon the Semantic Web Service standardization effort that is currently ongoing within the OASIS Semantic Execution Environment Technical Committee (SEE-TC)[7]. 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.

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 chapter we describe the approach for the auto-

[1]http://www.w3.org/Submission/OWL-S/
[2]http://www.wsmo.org/
[3]http://www.w3.org/Submission/2010/SUBM-WSMO-Lite-20100823/
[4]http://www.w3.org/2002/ws/sawsdl/
[5]http://www.w3.org/Submission/SA-REST/
[6]http://www.wsmo.org/TR/d38/v0.1/
[7]www.oasis-open.org/committees/ex-semantics

matic evaluation of Semantic Web Services using the SEALS platform and the results of the SWS evaluation campaign (2010).

Within SEALS, we propose an evaluation approach that is informed by and improves existing SWS tool evaluation initiatives. In this sense, our approach shares the goals and objectives of these initiatives. 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. The evaluation of SWS tools uses metadata, described via ontologies, about the evaluation scenario, tools, test data 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 test data metadata informs how the test data 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 test data; and interpretations, which are the results obtained after the evaluation measures are applied over the raw results. The format of the results is also described via ontologies.

7.2 Previous Evaluations

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[8](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[9]. 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 conven-

[8]http://sws-challenge.org
[9]http://www.w3.org/2005/Incubator/swsc/XGR-SWSC-20080331

tional 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.

The Semantic Service Selection[10] (S3) contest 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[11]); SAWSDL matchmaker evaluation (over SAWSDL-TC[12]); cross evaluation (using JGD[13] collection). The participation in the S3 contest consists of: a) implementing the SME2[14] plug-in API for the participants 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 participants 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[15] (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.

7.3 Evaluation Design

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.

[10] http://www-ags.dfki.uni-sb.de/ klusch/s3/index.html
[11] http://projects.semwebcentral.org/projects/owls-tc/
[12] http://projects.semwebcentral.org/projects/sawsdl-tc/
[13] http://fusion.cs.uni-jena.de/professur/jgd
[14] http://www.semwebcentral.org/projects/sme2/
[15] http://ws-challenge.georgetown.edu/wsc09/technical_details.html

- 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.
- 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.

7.4 Evaluation Scenario

In the first SEALS evaluation campaign (2010) we executed the SWS discovery evaluation scenario[16]. This evaluation scenario was performed experimentally in order to

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

test services of the SEALS platform. Note that not all capabilities were available at the moment. Therefore, the results were also experimental and comparable to the results obtained when participating for example in the S3 contest.

Basically, participants register their tool via the Web interface provided in the SEALS website[17] and the organizers download and run their tools as part of the evaluation campaign scenario. Participants are also required to implement the SWS Tool plugin API according to the instructions provided in the website. The organizers make instructions available for the participants about the scenario and perform the evaluation automatically by executing the evaluation workflow. The results become available in the Results Repository.

We focused 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 SEALS platform, through the provided services, measures the rate of matching correctness based on a number of metrics.

7.4.1 The SWS Plugin

In SEALS we provide the SWS plugin API, available from the campaign website, that must be implemented by tool providers participating in the SWS tool evaluation. The SWS Plugin API has been derived from the SEE API 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. 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, which can be used as the score value. It is expected that the services that do not match are returned with match degree 'NONE'.

7.4.2 Implemented Measures

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

[17] http://www.seals-project.eu/registertool

measures. For a survey of existing measures we refer the reader to existing published papers, for example [13] and [40].

In SEALS we make the measurement services available as a plugin (currently Java interfaces). That is, we can implement the interface using any publicly available metrics APIs. In the interface, *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 *Discovery Result* class is part of the SWS plugin API (Section above). The *DiscoveryReferenceSet* class 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 or confidence (score) value returned by a tool. The *MetricsResult* class 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.

In our current implementation of the *DiscoveryMeasurement* interface we use the evaluation metrics available from **Galago**[18] as described in Table 7.1. Galago is a open source toolkit for experimenting with text search. It is based on small, pluggable components that are easy to replace and change, both during indexing and during retrieval. In particular, we used the retrieval evaluator component, which computes a variety of standard information retrieval metrics commonly used in TREC. Galago is written in Java and works on any system with a JDK version 1.5 or later.

7.5 Test Data

For our evaluation we used the OWLS-TC 4.0 test collection[19], which 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 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. 160 services and 18 queries contain Precondition and/or Effect as part of their descriptions.

In order to make the OWLS-TC4.0 test collection available via the SEALS test data repository, we created metadata described with the *Suite* ontology and the *DiscoveryTestSuite* ontology as described in D5.4 [47] and D14.2 [10]. The test collection suite was encapsulated in a ZIP file, including the metadata, which has the file name *Metadata.rdf* and registered in the SEALS Test Data Repository.

According to the *Suite* ontology, the suite metadata is described with the concepts *Suite, SuiteItem and DataItem*, where a Suite consist of multiple SuiteItems, which itself consists of various DataItems. In the metadata for OWLS-TC4.0, the Suite

[18] http://www.galagosearch.org/galagosearch-core/apidocs/
[19] http://projects.semwebcentral.org/projects/owls-tc/

Table 7.1: Description of Metrics as implemented by Galago (reproduced for convenience).

Measure	Description
Num Ret	Number of retrieved documents
Num Rel	Total number of documents judged relevant
Num Rel Ret	Number of retrieved documents that were judged relevant
Aver Prec	Average Precision. "Suppose the precision is evaluated once at the rank of each relevant document in the retrieval. If a document is not retrieved, we assume that it was retrieved at rank infinity. The mean of all these precision values is the average precision".
NDCG @ N	Normalized Discounted Cumulative Gain at cutoff point N. "This measure was introduced in Jarvelin, Kekalainen, "IR Evaluation Methods for Retrieving Highly Relevant Documents" SIGIR 2001. The formula is copied from Vassilvitskii, "Using Web-Graph Distance for Relevance Feedback in Web Search", SIGIR 2006. Score = $N \sum_i (2^{r(i)} - 1)/\log(1+i)$, where N is such that the score cannot be greater than 1. We compute this by computing the DCG (unnormalized) of a perfect ranking".
NDCG	Normalized Discounted Cumulative Gain. "NDCG at (Math.max(retrieved.size(), judgments.size())".
R-prec	R-Precision. "Returns the precision at the rank equal to the total number of relevant documents retrieved. This method is equivalent to precision(relevantDocuments().size())".
Bpref	Binary Preference. "The binary preference measure, as presented in Buckley, Voorhees "Retrieval Evaluation with Incomplete Information", SIGIR 2004. The formula is: $1/R \sum_r 1 - \|nrankedgreaterthanr\|/R$ where R is the number of relevant documents and n is a member of the set of first R judged irrelevant documents retrieved".
Recip Rank	Reciprocal Rank. "Returns the reciprocal of the rank of the first relevant document retrieved, or zero if no relevant documents were retrieved".
Precision @ N	Precision at cutoff point N. "Returns the precision of the retrieval at a given number of documents retrieved. The precision is the number of relevant documents retrieved divided by the total number of documents retrieved".
Recall @ N	Recall at cutoff point N. "Returns the recall of the retrieval at a given number of documents retrieved. The recall is the number of relevant documents retrieved divided by the total number of relevant documents for the query".

Items in Metadata.rdf consist of 42 discovery tests, which correspond to the number of reference sets in the test collection. Each Discovery Test Suite consists of a goal document and a list of judged service documents associated with it. Each service has a relevance element, indicating the relevance value between the service document and the goal document. The test collection can be accessed via the Test Data Repository Service[20] We selected a number of queries and used the service descriptions (from the test collection) included in their reference sets. The test collection was also deployed locally in a Glassfish Server and made available as prescribed by OWLS-TC. For example, service descriptions were available at http://127.0.0.1/services/1.1/.

7.6 Tools Evaluated

The list of participating tools is shown in Table 7.2. These tools are variants of OWLS-MX[21] and are publicly available. Each OWLS-MX variant runs a different similarity measure algorithm and can be adjusted to run with certain parameters, which include type of sorting (Semantic (2); Syntactic(1); hybrid (0)) and syntactic similarity threshold (0.0 - 1.0) .

For our evaluation we packed each variant as a different tool, according to the instructions provided in the website[22]. We downloaded the source code and created a jar of the OWLS-MX API not including the GUI APIs.

Regarding the SWS Plugin API, we implemented the required methods (*init()*, *loadServices()*, *discover()*) by calling the appropriate methods of the OWLS-MX API.

7.6.1 Tool Parameter Settings

OWLS-M0 was configured in a way that only semantic based results (degree of match equals to 0 (*exact*), 1 (*plugin*), 2 (*subsumes*) or 3 (*subsumed-by*)) were included in the set of retrieved matches (raw results). Thus, it used the semantic type of sorting mentioned previously.

OWLS-M2 and OWLS-M3 were configured in a way that all results with syntactic similarity greater than zero were included in the set of retrieved matches (raw results). The degree of match was not taken into account. They used the syntactic type of sorting mentioned previously.

OWLS-M4 was configured in a way that only syntactic based results (degree of match equals 4 (*nearest neighbour*)) with syntactic similarity greater than 0.7 were included in the set of retrieved matches (raw results). It used the syntactic type of sorting mentioned previously.

The goals for these setting were: a) to compare the performance of a semantic based variant (OWLS-M0) with the performance of a syntactic based variant (OWLS-M4); and b) compare the performance of two syntactic based variants (OWLS-M2 and OWLS-M3) using different matching algorithms (similarity measures).

[20] http://seals.sti2.at/tdrs-web/testdata/persistent/OWLS-TC/4.0/suite/
[21] http://projects.semwebcentral.org/projects/owls-mx/
[22] http://www.seals-project.eu/seals-evaluation-campaigns/semantic-web-services

Table 7.2: Evaluation campaign participating tools.

Tool	Description
OWLS-M0	OWLS-MX variant - Constraint Similarity Measure
OWLS-M2	OWLS-MX variant - Extended Jaccard Similarity Measure
OWLS-M3	OWLS-MX variant - Cosine Similarity Measure
OWLS-M4	OWLS-MX variant - Jensen Shannon Similarity Measure

7.7 Evaluation Results

We show the results for three individual Goals (service requests) within OWLS-TC 4.0 as shown in Table 7.3. We have not performed overall measures (over all goals).

Table 7.3: Goals (service requests) in evaluation.

Goal	Name	URI
request id="5"	car price service	http://127.0.0.1/queries/1.1/car_price_service.owls
request id="8"	2 For 1 Price service	http://127.0.0.1/queries/1.1/ dvd-playermp3player_price_service.owls
request id="21"	Researcher address service	http://127.0.0.1/queries/1.1/researcher-in-academia_address_service.owls

Results from our evaluation for the test data and tools explained in the previous sections are given in Table 7.4. This table shows the comparative retrieval performance per Goal (service request) of OWLS-M0, OWLS-M2, OWLS-M3, OWLS-M4 over OWLS-TC4 datasets. The two first rows indicate the parameter settings of the tools as explained previously. The metrics have been explained in Table 7.1.

For reference purposes, we added a reference implementation (*Ref*), which accessed the dataset and retrieved all the relevant services from the reference set. Thus, the expected values for Precision and Recall of the *Ref* tool is one. We used the number of documents retrieved as the cutoff point for all tests. Loading time was calculated by measuring the time to execute the *loadService()* method in the SWS plugin.

We downloaded the test suites metadata from the SEALS repository and performed the evaluation in a local machine. The configuration of the local machine was an Intel Core 2 Duo CPU, 3 GHz processor, 64-bit OS, 4GB RAM, running Windows 7.

	Ref	OWLS-M0	OWLS-M2	OWLS-M3	OWLS-M4
Semantic Sorting	-	yes	no	no	no
Syntactic Threshold	-	-	0	0	0.70
Request Id=05					
Load time (ms)	243	22358	21713	21745	21592

continued on next page

continued from previous page					
	Ref	OWLS-M0	OWLS-M2	OWLS-M3	OWLS-M4
Dataset size	176	176	176	176	176
Num Rel	93	93	93	93	93
Num Ret	93	21	174	174	5
Num Rel Ret	93	20	93	93	4
Aver Prec	47.00	2.26	47.00	47.00	0.11
NDCG	4.68	1.01	4.68	4.69	0.20
NDCG at Ret	4.68	2.75	4.68	4.68	1.36
R-prec	1.00	0	1.00	1.00	0
Bpref	0	0	0	0	0
Recip Rank	1.00	1.00	1.00	1.00	1.00
Precision at Ret	1.00	0.95	0.54	0.54	0.80
Recall at Ret	1.00	0.22	1.00	1.00	0.04
Request Id=08					
Load time (ms)	715	29910	29892	30448	30558
Dataset size	171	171	171	171	171
Num Rel	27	27	27	27	27
Num Ret	27	24	164	164	44
Num Rel Ret	27	20	27	27	6
Aver Prec	14.00	7.78	14.00	14.00	0.78
NDCG	3.16	2.34	3.16	3.16	0.70
NDCG at Ret	3.16	2.53	3.16	3.16	0.70
R-prec	1.00	0	1.00	1.00	0.22
Bpref	0	0	0	0	0
Recip Rank	1.00	1.00	1.00	1.00	1.00
Precision at Ret	1.00	0.95	0.17	0.17	0.14
Recall at Ret	1.00	0.74	1.00	1.00	0.22
Request Id=21					
Load time (ms)	756	144176	144421	144875	145062
Dataset size	283	283	283	283	283
Num Rel	72	72	72	72	72
Num Ret	72	19	76	76	5
Num Rel Ret	72	18	32	32	5
Aver Prec	36.50	2.38	7.33	7.33	0.21
NDCG	4.34	1.09	1.93	1.93	0.30
NDCG at Ret	4.34	2.64	1.93	1.93	1.70
R-prec	1.00	0	0.44	0.44	0
Bpref	0	0	0	0	0
Recip Rank	1.00	1.00	1.00	1.00	1.00
Precision at Ret	1.00	0.95	0.42	0.42	1.00
Recall at Ret	1.00	0.25	0.44	0.44	0.07

Table 7.4: Comparative tool performance on the OWLS-TC4 dataset.

SWS tools evaluation produces two types of results, stored in the Result Repository Service: *raw results*[23], which are generated by running a tool with specific test data, and *interpretations*[24], which are obtained after the evaluation metrics are applied over the raw results.

7.8 Lessons Learnt

We have performed an experimental evaluation with the objective of testing the SEALS infrastructure. The evaluation results are not meant to be conclusive in terms of tool performance, but instead count as input to requirements for the SEALS infrastructure. Currently, we do not provide comparative results in graphical form. Instead, evaluation results are stored in RDF and can be visualized in HTML.

From our analysis, we find that public intermediate results and repeatability are important for studying the behaviour of the tools under different settings (not only best behaviour). In addition, the SEALS infrastructure can help in several steps of the evaluation process, including generating and accessing datasets and results via metadata.

With respect to the datasets we noticed that all variants of OWLS-MX could retrieve all the same relevant services for a given reference set, provided that they were set with the same parameters. Thus, either the algorithms are not being invoked properly or the measures seem not to account for the different matching algorithms (similarity measures) used by the different variants (e.g. execution time). When recall at number retrieved is equal to 1 for a high number of queries, this indicates bias of the test suite against the tool. It would be important to check whether OWLS-TC has bias towards the OWLS-MX tools. It is important that the SWS community get more engaged in creating non-biased datasets.

In addition, given that small changes in parameters can produce different results, it is also important to make intermediate results available (via evaluation services) and provide alternative metrics. Overall, it was not easy to say why some tools fail to certain queries. But, we have noticed that some ontologies could not be read. Thus, it is important to introduce some validation procedure.

With respect to measures, we have implemented for this evaluation the API provided by Galago, which provides several measures covering the ones provided by SME2 (S3 Contest), and also additional ones. The SEALS infrastructure allows to plugin different source APIs as metrics services.

In this chapter we have shown how to use the ongoing approach and services for SWS tools evaluation using the SEALS platform. The SWS plugin API is currently very similar to SME2'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

[23] http://seals.sti2.at/rrs-web/results/
[24] http://seals.sti2.at/rrs-web/interpretations/

and sharable metadata specifications for test data, tools and evaluation descriptions as well as respective public repositories. In addition, results are publicly available and can be used for alternative visualizations by users.

For future campaigns we plan to provide new datasets not yet available from existing evaluation initiatives and add new evaluation scenarios such as SWS composition

8. Conclusions

This report has presented an overview of the first series of evaluation campaigns organised in the SEALS project for the five types of technologies covered in it: ontology engineering tools, ontology reasoning systems, ontology matching tools, semantic search tools, and semantic web services.

In these five evaluation campaigns, 32 tools from all around the world were evaluated using common evaluation methods and test data. In some cases, we followed existing evaluation methods and used available test data; in other cases, we defined new evaluations methods and test data to enhance the evaluations performed in the evaluation campaigns.

We have established as a result of our experiences from the first evaluation campaigns that the chosen evaluation methodologies and test data are an appropriate basis for discovering useful evaluation results from the participating tools. In general, it can be seen that semantic tools are reaching maturity with respect to the key characteristics for their domains, and hence there is a real value to be had in comparative evaluation to guide tool selection, since different tools in the same domain still exhibit significant differences in implementation or functionality which are of importance in differing usage scenarios.

We aim in the second campaign to broaden the extent of involved tools in the evaluations, since this will improve the possibility to determine the current state of the art of the tools in the given domain, and how they compare to one another.

All the resources used in the SEALS Evaluation Campaigns as well as the results obtained in them will be publicly available through the SEALS Platform. This way, anyone interested in evaluating one of the technologies covered in the project will be able to do so, and to compare to others, with a small effort. The SEALS evaluation infrastructure will be open to all via the SEALS website, requiring only a simple preregistration in order to be able to access our Community Area. Within the Community Area, there is the possibility to register a tool, describe it, upload it to SEALS and execute evaluations upon it, gaining immediately an insight into how it compares to the previously evaluated tools.

Our future plans are to extend the evaluations defined for the different types of technologies and, once these extensions are ready, we plan to conduct a second edition of the SEALS Evaluation Campaigns. This second Campaign is scheduled to begin in the summer of 2011, and by the close of the SEALS project in early 2012 we will publish a second white paper on semantic tool evaluation which is intended for potential tool adopters, in order to guide them with respect to their choice of tools when seeking to benefit from the use of semantic technology within their systems and IT projects.

References

[1] J. Angele and Y. Sure, editors. *Proceedings of the 1st International Workshop on Evaluation of Ontology-based Tools (EON2002)*, volume 62, Sigüenza, Spain, September 2002. CEUR-WS.

[2] Renzo Angles and Claudio Gutierrez. The expressive power of SPARQL. In *ISWC '08: Proceedings of the 7th International Conference on The Semantic Web*, pages 114–129, Berlin, Heidelberg, 2008. Springer-Verlag.

[3] Benhamin Ashpole, Marc Ehrig, Jérôme Euzenat, and Heiner Stuckenschmidt, editors. *Integrating Ontologies'05, Proc. of the K-Cap Workshop on Integrating Ontologies*, Banff (Canada), 2005.

[4] A. Bangor, P. T. Kortum, and J. T. Miller. An empirical evaluation of the system usability scale. *International Journal of Human-Computer Interaction*, 24(6):574–594, 2008.

[5] A. Bangor, P. T. Kortum, and J. T. Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4(3):114–123, 2009.

[6] B.Glimm, M. Horridge, B. Parsia, and P. F. Patel-Schneider. A syntax for rules in owl 2. In *Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED 2009)*, volume 529. CEUR, 2009.

[7] Ravish Bhagdev, Sam Chapman, Fabio Ciravegna, Vitaveska Lanfranchi, and Daniela Petrelli. Hybrid search: Effectively combining keywords and ontology-based searches. In Manfred Hauswirth, Manolis Koubarakis, and Sean Bechhofer, editors, *Proceedings of the 5th European Semantic Web Conference*, LNCS, Berlin, Heidelberg, June 2008. Springer Verlag.

[8] John Brooke. SUS: a quick and dirty usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, and I. L. McClelland, editors, *Usability Evaluation in Industry*, pages 189–194. Taylor and Francis, 1996.

[9] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics*, 6(4):309–322, November 2008.

[10] L. Cabral, I. Toma, and Adrian Marte. D14.2. Services for the Automatic Evaluation of Semantic Web Service Tools v1. Technical report, SEALS Project, August 2010.

[11] Caterina Caracciolo, Jérôme Euzenat, Laura Hollink, Ryutaro Ichise, Antoine Isaac, Véronique Malaisé, Christian Meilicke, Juan Pane, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, and Vojtech Svátek. Results of the ontology alignment evaluation initiative 2008. In *Proc. of the 3rd International Workshop on Ontology Matching (OM-2008), collocated with ISWC-2008*, pages 73–120, Karlsruhe (Germany), 2008.

[12] Brian de Alwis and Gail C. Murphy. Answering conceptual queries with ferret. In *ICSE '08: Proceedings of the 30th international conference on Software engineering*, pages 21–30, New York, NY, USA, 2008. ACM.

[13] G. Demartini and S. Mizzaro. A Classification of IR Effectiveness Metrics. In *Proceedings of ECIR 2006*. LNCS 3936, pp. 488 to 491, Springer, 2006.

[14] Elena Demidova and Wolfgang Nejdl. Usability and expressiveness in database keyword search : Bridging the gap. In *In Proceedings of the PhD Workshop at VLDB*, 2009.

[15] David J. DeWitt. The wisconsin benchmark: Past, present, and future. In Jim Gray, editor, *The Benchmark Handbookfor Database and Transaction Systems (2nd Edition)*. Morgan Kaufmann, 1993.

[16] Marc Ehrig and Jérôme Euzenat. Relaxed precision and recall for ontology matching. In *Proc. of the K-Cap Workshop on Integrating Ontologies*, pages 25–32, Banff (Canada), 2005.

[17] Jérôme Euzenat. An API for ontology alignment. In *Proc. of the 3rd International Semantic Web Conference (ISWC-2004)*, pages 698–712, Hiroshima (Japan), 2004.

[18] Jérôme Euzenat, Alfio Ferrara, Laura Hollink, Antoine Isaac, Cliff Joslyn, Véronique Malaisé, Christian Meilicke, Andriy Nikolov, Juan Pane, Marta Sabou, François Scharffe, Pavel Shvaiko, Vassilis Spiliopoulos, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, Cássia Trojahn dos Santos, George Vouros, and Shenghui Wang. Results of the ontology alignment evaluation initiative 2009. In *Proc. of the 4th Workshop on Ontology Matching (OM-2009), collocated with ISWC-2000*, pages 73–126, Chantilly (USA), 2009.

[19] Jérôme Euzenat, Alfio Ferrara, Christian Meilicke, Juan Pane, François Scharffe, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, and Cássia Trojahn dos Santos. Results of the ontology alignment evaluation initiative 2010. In Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Natasha Noy, and Arnon Rosenthal, editors, *Proc. 5th ISWC workshop on ontology matching (OM), Shanghai (Chine)*, pages 1–35, 2010.

[20] Jérôme Euzenat, Antoine Isaac, Christian Meilicke, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Svab, Vojtech Svatek, Willem Robert van Hage, and Mikalai Yatskevich. Results of the ontology alignment evaluation initiative 2007. In *Proc. of the 2nd International Workshop on Ontology Matching (OM-2008), collocated with ISWC-2007*, pages 96–132, Busan (Korea), 2007.

[21] Jérôme Euzenat, Malgorzata Mochol, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Svab, Vojtech Svatek, Willem Robert van Hage, and Mikalai Yatskevich. Results of the ontology alignment evaluation initiative 2006. In *Proc. of the 1st International Workshop on Ontology Matching (OM-2006), collocated with ISWC-2006*, pages 73–95, Athens, Georgia (USA), 2006.

[22] R. García-Castro. *Benchmarking Semantic Web technology*, volume 3 of *Studies on the Semantic Web*. AKA Verlag – IOS Press, January 2010.

[23] R. García-Castro and A. Gómez-Pérez. Guidelines for benchmarking the performance of ontology management APIs. In Y. Gil, E. Motta, R. Benjamins, and M. Musen, editors, *Proceedings of the 4th International Semantic Web Conference (ISWC2005)*, number 3729 in LNCS, pages 277–292, Galway, Ireland, November 2005. Springer-Verlag.

[24] 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.

[25] 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.

[26] R. García-Castro, S. Grimm, M. Schneider, M. Kerrigan, and G. Stoilos. D10.1. Evaluation design and collection of test data for ontology engineering tools. Technical report, SEALS Project, November 2009.

[27] R. García-Castro and F. Martín-Recuerda. D3.1 SEALS Methodology for Evaluation Campaigns v1. Technical report, SEALS Consortium, 2009.

[28] R. García-Castro, I. Toma, A. Marte, M. Schneider, J. Bock, and S. Grimm. D10.2. Services for the automatic evaluation of ontology engineering tools v1. Technical report, SEALS Project, July 2010.

[29] Raúl García-Castro, Asunción Gómez-Pérez, Oscar Muñoz-García, and Lyndon J.B. Nixon. Towards a Component-Based Framework for Developing Semantic Web Applications. In J. Domingue and C. Anutariya, editors, *3rd Asian Semantic Web Conference (ASWC 2008)*, Lecture Notes in Computer Science, pages 197–211, Bangkok, Thailand, December 2008. Springer-Verlag.

[30] T. Gardiner, I. Horrocks, and D. Tsarkov. Automated benchmarking of description logic reasoners. In *Proc. of the 2006 Description Logic Workshop (DL 2006)*, volume 189, 2006.

[31] Y. Guo, Z. Pan, and J. Heflin. Lubm: A benchmark for owl knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):158 – 182, 2005.

[32] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):158–182, October 2005.

[33] M. Horridge and S. Bechhofer. The OWL API: A Java API for Working with OWL 2 Ontologies. In Rinke Hoekstra and Peter F. Patel-Schneider, editors, *OWLED*, volume 529 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[34] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.

[35] I. Horrocks, P. F. Patel-Schneider, and R. Sebastiani. An analysis of empirical testing for modal decision procedures. *Logic Journal of the IGPL*, 8(3):293–323, 2000.

[36] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[37] ISO/IEC. *ISO/IEC 9126-1. Software Engineering – Product Quality – Part 1: Quality model*. 2001.

[38] Yves R. Jean-Mary, E. Patrick Shironoshitaa, and Mansur R. Kabuka. Ontology matching with semantic verification. *Journal of Web Semantics*, 7(3):235–251, 2009.

[39] Esther 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.

[40] U. Kuester and B. Koenig-Ries. Measures for Benchmarking Semantic Web Service Matchmaking Correctness. In *Proceedings of ESWC 2010*. LNCS 6089, Springer, June 2010.

[41] P. Lambrix, M. Habbouche, and M. Pérez. Evaluation of ontology development tools for bioinformatics. *Bioinformatics*, 19(12):1564–1571, 2003.

[42] Yuangui Lei, Victoria Uren, and Enrico Motta. Semsearch: A search engine for the semantic web. In *Proc. 5th International Conference on Knowledge Engineering and Knowledge Management Managing Knowledge in a World of Networks, Lect. Notes in Comp. Sci., Springer, Podebrady, Czech Republic*, pages 238–245, 2006.

[43] L.Ryan. Scalability report on triple store applications. Technical report, SIMILE Project, November 2004.

[44] Marko Luther, Thorsten Liebig, Sebastian Bhm, and Olaf Noppens. Who the heck is the father of bob? In *6th Annual European Semantic Web Conference (ESWC2009)*, pages 66–80, June 2009.

[45] L. Ma, Y. Yang, Z. Qiu, G. T. Xie, Y. Pan, and S. Liu. Towards a complete OWL ontology benchmark. In *ESWC*, pages 125–139, 2006.

[46] Li Ma, Yang Yang, Zhaoming Qiu, GuoTong Xie, Yue Pan, and Shengping Liu. Towards a complete OWL ontology benchmark. In Y. Sure and J. Domingue, editors, *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006)*, volume 4011 of *LNCS*, pages 125–139, Budva, Montenegro, June 11-14 2006. Springer-Verlag.

[47] A. Marte and D. Winkler. D5.4 Iterative evaluation and implementation of the Test Data Repository Service. Technical report, SEALS Project, November 2010.

[48] 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.

[49] Christian Meilicke and Heiner Stuckenschmidt. Incoherence as a basis for measuring the quality of ontology mappings. In *Proc. of the 3rd Workshop on Ontology Matching (OM-2008), collocated ISWC-2008*, pages 1–12, Karlsruhe (Germany), 2008.

[50] B. Motik, R. Shearer, and I. Horrocks. Hypertableau reasoning for description logics. *J. of Artificial Intelligence Research*, 2009. To appear.

[51] B. Motik, R. Shearer, and I. Horrocks. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research*, 36:165–228, 2009.

[52] OntoWeb. Ontoweb deliverable 1.3: A survey on ontology tools. Technical report, IST OntoWeb Thematic Network, May 2002.

[53] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. Technical report, W3C Recommendation 10 February 2004, 2004.

[54] P. F. Patel-Schneider and R. Sebastiani. A new general method to generate random modal formulae for testing decision procedures. *J. Artif. Intell. Res. (JAIR)*, 18:351–389, 2003.

[55] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. Towards a theory of natural language interfaces to databases. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157, New York, NY, USA, 2003. ACM.

[56] A. L. Rector and J. Rogers. Ontological and practical issues in using a description logic to represent medical concept systems: Experience from galen. In *Reasoning Web*, pages 197–231, 2006.

[57] Jonathan Sillito, Gail C. Murphy, and Kris De Volder. Questions programmers ask during software evolution tasks. In *SIGSOFT '06/FSE-14: Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 23–34, New York, NY, USA, 2006. ACM.

[58] Jonathan Sillito, Gail C. Murphy, and Kris De Volder. Asking and answering questions during a programming change task. *IEEE Transactions on Software Engineering*, 34:434–451, 2008.

[59] Y. Sure and O. Corcho, editors. *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003)*, volume 87, Sanibel Island, Florida, USA, October 2003. CEUR-WS.

[60] York Sure, Oscar Corcho, Jérôme Euzenat, and Todd Hughes, editors. *Proc. of the Workshop on Evaluation of Ontology-based Tools (EON-2004), collocated with ISWC-2004*, Hiroshima (Japan), 2004.

[61] Lappoon R. Tang and Raymond J. Mooney. Using multiple clause constructors in inductive logic programming for semantic parsing. In *In Proceedings of the 12th European Conference on Machine Learning*, pages 466–477, 2001.

[62] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.

[63] D. Tsarkov, I. Horrocks, and P. F. Patel-Schneider. Optimizing terminological reasoning for expressive description logics. *J. Autom. Reasoning*, 39(3):277–316, 2007.

[64] Victoria Uren, Yuangui Lei, Vanessa Lopez, Haiming Liu, Enrico Motta, and Marina Giordanino. The usability of semantic search tools: a review. *The Knowledge Engineering Review*, 22(04):361–377, 2007.

[65] T. Wang, B. Parsia, and J. Hendler. A survey of the web ontology landscape. In *Proceedings of the International Semantic Web Conference, ISWC*, 2006.

[66] S. N. Wrigley, K. Elbedweihy, D. Reinhard, A. Bernstein, and F. Ciravegna. D13.3 Results of the first evaluation of semantic search tools. Technical report, SEALS Consortium, 2010.

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

[68] M. Yatskevich, T. Tserendorj, J. Bock, and A. Marte. D11.2 Services for the automatic evaluation of advanced reasoning systems. Technical report, SEALS Consortium, 2010.