Interoperability results for Semantic Web technologies using OWL as the interchange language

Raúl García-Castro*, Asunción Gómez-Pérez

Ontology Engineering Group, Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid Campus de Montegancedo s/n, Boadilla del Monte, 28660 Madrid, Spain

A B S T R A C T

Using Semantic Web technologies in complex scenarios requires that such technologies correctly interoperate by interchanging ontologies using the RDF(S) and OWL languages. This interoperability is not straightforward because of the high heterogeneity in Semantic Web technologies and, while the number of such technologies grows, affordable mechanisms for evaluating Semantic Web technology interoperability are needed to comprehend the current and future interoperability of Semantic Web technologies.

This paper presents the OWL Interoperability Benchmarking, an international benchmarking activity that involved the evaluation of the interoperability of different Semantic Web technologies using OWL as the interchange language. It describes the evaluation resources used in this benchmarking activity, the OWL Lite Import Benchmark Suite and the IBSE tool, and presents how to use them for evaluating the OWL interoperability of Semantic Web technologies. Moreover, the paper offers an overview of the OWL interoperability results of the eight tools participating in the benchmarking: one ontology-based annotation tool (GATE), three ontology frameworks (Jena, KAON2, and SWI-Prolog), and four ontology development tools (Protégé Frames, Protégé OWL, SemTalk, and WebODE).

1. Introduction

Even though the number of Semantic Web tools is already rather large,[1] many more tools are created every year. The heterogeneity of Semantic Web tools exists not only because they have different functionalities (ontology development tools, ontology repositories, ontology matching tools, etc.) but also because different tools use different representation formalisms, such as Frames or Description Logics. These representation formalisms, in turn, have different knowledge representation expressiveness and different reasoning capabilities that are needed in different use scenarios.

The use of ontologies in applications requires to cope with this heterogeneity in representation formalisms when ontologies are interchanged between different tools. Similarly, in the case of dealing with multiple ontologies, applications also require to integrate such ontologies, but this latter topic is out of the scope of this paper.

A sample ontology reuse scenario can be one where an ontology is developed using an ontology development tool and, after checking its consistency using a reasoner, it is stored into an ontology repository so it can be visualized using an ontology browser. In such interchanges, ontologies are usually stored in a shared resource (e.g., the ontology repository) and represented using the RDF(S) and OWL languages, which were proposed by the W3C in 2004.

Current Semantic Web tools have problems in interchanging RDF(S) and OWL ontologies, either when these ontologies come from other tools or when they are downloaded from the Web. Such problems sometimes are due to the different representation formalisms used by the tools, since not every tool natively supports RDF(S) or OWL. However, very often the problems are due to other causes such as defects in the tools. Not to be aware of these interoperability problems prevents a correct selection of Semantic Web technologies and leads to problems when combining different technologies in complex applications.

Besides, the actual interoperability between the existing Semantic Web technologies is unknown, and this is so mainly because such interoperability is not evaluated, since there is no easy way of performing such evaluations. Therefore, we need methods and tools for evaluating Semantic Web technology interoperability at a large scale and in an easy and economical way, which requires defining evaluations focused on their reusability.

A previous benchmarking activity covered the interoperability of Semantic Web tools using RDF(S) as the interchange language [16]. As a result, we obtained a clear picture of the RDF(S) interoperability of the tools participating in the benchmarking, namely, Corese, Jena, KAON, Sesame, Protégé, and WebODE.

* Corresponding author. Tel.: +34 91 336 3670.
E-mail addresses: rgarcia@fi.upm.es (R. García-Castro), asun@fi.upm.es (A. Gómez-Pérez).

[1] 810 tools are listed in http://www.mkbergman.com/?page_id=325 by 19/01/2010.

Now the objective is to analyse the OWL interoperability of any type of Semantic Web technology in a cheap and reproducible way. To this end, the OWL Interoperability Benchmarking was organised with the goals of providing mechanisms for automatic evaluation of the interoperability of Semantic Web technologies using OWL as the interchange language and of assessing and improving the current OWL interoperability of Semantic Web technologies.

This paper presents a summary of this benchmarking activity and an overview of the OWL interoperability results of the eight tools participating in it: one ontology-based annotation tool (GATE), three ontology frameworks (Jena, KAON2, and SWI-Prolog), and four ontology development tools (Protégé Frames, Protégé OWL, SemTalk, and WebODE).

This paper is structured as follows: Section 2 states the problem of the interoperability of Semantic Web technologies. Sections 3 and 4 present previous interoperability evaluations and the UPM framework for benchmarking interoperability that can be used in interoperability evaluation activities, respectively. Section 5 introduces the OWL Interoperability Benchmarking and Section 6 describes the experiment performed in this benchmarking activity. Section 7 concerns the set of ontologies to use as input for the experiment, namely, the OWL Lite Import Benchmark Suite. Section 8 deals with IBSE, the automatic evaluation infrastructure, and with how it can be used. Section 9 provides the analysis of the OWL compliance of the Semantic Web tools participating in the benchmarking and Section 10 presents the analysis of their OWL interoperability. Finally, Section 11 draws the conclusions from this work and proposes future lines of work.

2. Semantic Web technology interoperability

According to the Institute of Electrical and Electronics Engineers (IEEE), interoperability is the ability of two or more systems or components to exchange information and to use this information [20]. Duval proposes a similar definition by stating that interoperability is the ability of independently developed software components to exchange information so they can be used together [10]. For us, interoperability is the ability that Semantic Web tools have to interchange ontologies and use them.

One of the factors that affects interoperability is heterogeneity. Sheth [30] classifies the levels of heterogeneity of any information system into information heterogeneity and system heterogeneity. In this paper, only information heterogeneity (and, therefore, interoperability) is considered, whereas system heterogeneity, which includes heterogeneity due to differences in information systems or platforms (hardware or operating systems) is disregarded.

Furthermore, interoperability is treated in this paper in terms of knowledge reuse and must not be confused with the interoperability problem caused by the integration of resources. This alternative notion of interoperability is related to the ontology alignment problem [11], that is, the problem of how to find relationships between entities in different ontologies.

2.1. The interoperability problem

Semantic Web technologies appear in different forms (ontology development tools, ontology repositories, ontology matching tools, reasoners, etc.) and interoperability is a must for these technologies because they need to interchange ontologies and use them in the distributed and open environment of the Semantic Web.

On the other hand, interoperability is a problem for the Semantic Web due to the heterogeneity of the knowledge representation formalisms of the different existing systems, since each formalism provides different knowledge representation expressivity and different reasoning capabilities, as it occurs in knowledge-based systems [5].

Current Semantic Web technologies manage different representation formalisms, e.g., the W3C recommended languages RDF(S) and OWL, models based in Frames or in the different families of Description Logics, or other models such as the Unified Modeling Language[2] (UML), the Ontology Definition Metamodel[3] (ODM), or the Open Biomedical Ontologies[4] (OBO) language.

Fig. 1 shows the two common ways of interchanging ontologies within Semantic Web tools: directly by storing the ontology in the destination tool, or indirectly by storing the ontology in a shared resource, such as a fileserver, a web server, or an ontology repository.

The ontology interchange should pose no problems when a common representation formalism is used by all the systems involved in the interchange and there should be no differences between the original and the final ontologies (i.e., the αs and βs in the figure should be null).

However, in the real world, it is not feasible to use a single system, since each system provides different functionalities, nor is it to use a single representation formalism, since some representation formalisms are more expressive than others and different formalisms provide different reasoning capabilities, as previously mentioned.

Most of the Semantic Web systems natively manage a W3C recommended language, either RDF(S), OWL, or both; but some systems manage other representation formalisms. If the systems participating in an interchange (or the shared resource) have different representation formalisms, the interchange requires at least a translation from one formalism to the other. These ontology translations from one formalism to another formalism with different expressiveness cause information additions or losses in the ontology (the αs and βs in Fig. 1), once in the case of a direct interchange and twice in the case of an indirect one.

Besides, when systems manage ontologies using existing ontology management frameworks (e.g., Jena, Sesame, etc.), further translations may take place when the representation formalisms of the tool and of the framework are different.

Due to the heterogeneity between representation formalisms in the Semantic Web scenario, the interoperability problem is highly related to the ontology translation problem that occurs when common ontologies are shared and reused over multiple representation systems [17].

2.2. Categorising ontology differences

The differences between an ontology and the translated one can happen at different levels. Sometimes changes in one level cause changes in other levels; in other cases, changes in one level do not cause further changes in other levels.

Barrasa [2] summarises the different ontology heterogeneity levels according to the different classifications found in the literature [4,7,9,19,21,22,32,33]. These levels and classifications can be seen in Fig. 2. The levels are

- *Lexical.* At this level we encounter all the differences related to the ability of segmenting the representation into characters and words (or symbols).
- *Syntactic.* Here we encounter all forms of heterogeneity that depend on the choice of the representation format. Some mismatches are syntactic sugar while others are caused

[2] http://www.uml.org/.
[3] http://www.omg.org/ontology/.
[4] http://obofoundry.org/.

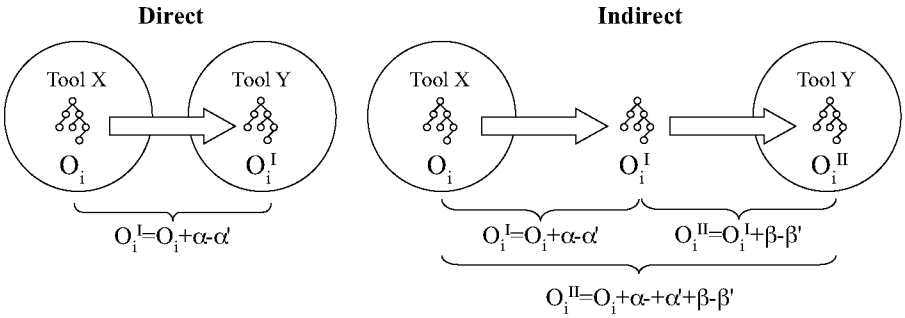


Fig. 1. Ontology interchanges within Semantic Web tools.

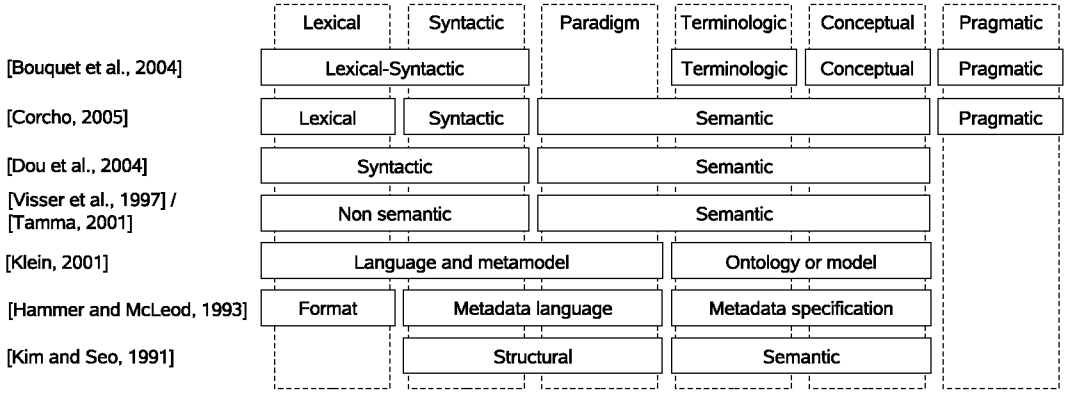


Fig. 2. Classification of ontology heterogeneity levels [2].

by expressing the same thing through a totally different syntax.

- *Paradigm.* Here we encounter mismatches caused by the use of different paradigms to represent concepts such as time, action, plans and causality.
- *Terminological.* At this level, we encounter all forms of mismatches related to the process of naming the entities (e.g., individuals, classes, properties, relations) that occur in an ontology.
- *Conceptual.* Here we encounter mismatches that have to do with the entities chosen to model a domain and that present differences in coverage, granularity and perspective.
- *Pragmatic.* Finally, at this level we encounter all the discrepancies that result from the fact that different individuals/communities may interpret the same ontology in different ways in different contexts.

3. Previous interoperability evaluations

In the Semantic Web area, technology interoperability has been scarcely evaluated. Some qualitative analyses have been performed in [29] concerning ontology development tools, ontology merge and integration tools, ontology evaluation tools, ontology-based annotation tools, and ontology storage and querying tools; and in [25] concerning ontology-based annotation tools. These analyses provide information about the interoperability capabilities of the tools (such as the platforms where they run, the tools they interoperate with, or the data and ontology formats they manage), but they give no empirical studies to support their conclusions.

The only exception is the experiment carried out in the *Second International Workshop on Evaluation of Ontology-based Tools* (EON2003). The central topic of this workshop was the evaluation of ontology development tool interoperability using an interchange language [31].

In this workshop, the participants were asked to model ontologies with their ontology development tools and to perform different tests for evaluating the import, export and interoperability of the tools.

The experiment had no restrictions on the interchange language, different languages (RDF(S), OWL, DAML, and UML) were used in different experiments, or on how to model the ontology to be interchanged, a natural language description of a domain was provided and each experimenter modelled the ontology in different ways.

Corcho's conclusions [7] are extracted from the results of these experiments and from an analysis of the main features of RDF(S) and OWL. These conclusions are the following:

- RDF(S) and OWL allow representing the same knowledge in different ways, making knowledge exchange difficult. This is so because RDF(S) and OWL ontologies can be serialized with different syntaxes (such as RDF/XML,[5] Notation3[6] or N-Triples[7]) and there are several ways to express knowledge in these syntaxes. Most of the existing tools can manage these syntaxes or use programming libraries to manage them. Nevertheless, some tools still use the serialized files directly, which can cause problems.
- The standard knowledge models of RDF(S), OWL Lite and OWL DL are not expressive enough to represent some of the knowledge that can be represented with traditional ontology languages and tools. Therefore, translations from more expressive knowledge models to these knowledge models usually involve knowledge loses.
- The translators to RDF(S) and OWL are usually written taking into account a specific language or tool. Two solutions have been

[5] http://www.w3.org/TR/rdf-syntax-grammar/.
[6] http://www.w3.org/DesignIssues/Notation3.
[7] http://www.w3.org/TR/rdf-testcases/#ntriples.

adopted to avoid the loss of knowledge when translating from a more expressive model to a less expressive one:
  . To represent the knowledge that could be lost with annotation properties (such as *rdfs:comment*) using a specific structure for this information.
  . To extend the RDF(S) and OWL vocabularies with ad-hoc properties not defined in the specifications.
- Current translation systems in ontology development tools still have many errors when exporting and/or importing RDF(S) and OWL.

The EON2003 experiment was a first and valuable step toward evaluating interoperability, since it highlighted interoperability problems in the existing tools using the W3C recommended languages for ontology interchange. Nevertheless, further evaluations of Semantic Web technology interoperability are required because

- Interoperability is a main problem for the Semantic Web that still requires substantial work.
- The workshop experiments concerned only a few tools and focused only on ontology development tools.
- Some experiments evaluated export functionalities, others, import functionalities, and only a few evaluated interoperability. Furthermore, interoperability from one tool to the same tool using an interchange language was not considered.
- No systematic evaluation was performed; each experiment used different evaluation procedures, interchange languages, and principles for modelling ontologies. Therefore, the results were not comparable and only specific comments and recommendations for each ontology development tool participating were made.

4. The UPM framework for benchmarking interoperability

As mentioned above, in the Semantic Web area we can find many tools that provide specific and limited functionalities. However, not being aware of the interoperability capabilities of the existing Semantic Web technologies causes important problems when more complex technologies and applications are built reusing existing technologies, and this ignorance regarding interoperability is mainly due to the fact that tool interoperability has not been evaluated because there is no easy way of making this evaluation.

The UPM framework for benchmarking interoperability[8] (UPM-FBI) aims to support the interoperability of Semantic Web technologies by providing all the resources needed for benchmarking the interoperability of these technologies using RDF(S) and OWL as interchange languages.

As Fig. 3 shows, the UPM-FBI provides four benchmark suites that contain the ontologies to be used in interoperability evaluations and two approaches for performing interoperability experiments (one manual and another automatic), each of them including different software tools that support the experiment execution and the result analysis.

The manual experimentation approach was followed in the *RDF(S) Interoperability Benchmarking*, a benchmarking of the interoperability of Semantic Web tools using RDF(S) as the interchange language, which was organised before we started the benchmarking presented in this paper. A description of this benchmarking activity and its results can be found in [16].

For the RDF(S) Interoperability Benchmarking effort, experiments were conducted by accessing the tools manually, using the RDF(S) Import, Export and Interoperability Benchmark Suites. Two different tools support this approach, namely, the *rdfsbs* tool, which automates part of the experiment execution for some tools, and the *IRIBA*[9] web application, which provides an easy way of dynamically analysing the results.

The manual experimentation and analysis of the results has the advantage of yielding highly detailed results, which permits diagnosing problems in the tools and, consequently, improving them, but the disadvantage is that it makes the experimentation costly. Some tool developers automated the execution of the experiments but not all of them. Furthermore, the results obtained may be influenced by human mistakes and they depend on the people performing the experiments and on their expertise with the tools.

The next sections deal with the OWL Interoperability Benchmarking, which follows the automatic approach of the UPM-FBI; in such an approach, the OWL Lite Import Benchmark Suite is used and the experiments and the results analysis are automated by means of the IBSE (Interoperability Benchmark Suite Executor) tool.

5. The OWL Interoperability Benchmarking

In the OWL Interoperability Benchmarking, we have followed the Knowledge Web benchmarking methodology [15], a methodology used before in the RDF(S) Interoperability Benchmarking and also employed for benchmarking the performance and the scalability of ontology development tools [14].

The most common way for Semantic Web technologies to interoperate is the indirect interchange of ontologies by storing them in a shared resource, which is the way considered here. A direct interchange of ontologies would require developing interchange mechanisms for each pair of tools, which would be very costly.

In our case, the representation formalism used to interchange ontologies is OWL, whereas the shared resource is a local filesystem in which ontologies are stored in text files serialized with the RDF/XML syntax, because this is the syntax most used by Semantic Web technologies.

Therefore, the two main goals that we want to achieve in the benchmarking are (1) to provide mechanisms for automatic evaluation of the interoperability of Semantic Web technologies using OWL as the interchange language, and (2) to assess and improve the OWL interoperability of Semantic Web technologies.

Although the goals here are similar to those of the RDF(S) Interoperability Benchmarking, this time our approach is quite different, thanks in part to the lessons learnt while carrying out the previous benchmarking activity. The main changes performed are the following:

- *Broadening the scope of the benchmarking* by contemplating any Semantic Web tool able to read and write ontologies from/to OWL files.
- *Diminishing the cost of the benchmarking by automating the experiments.* The cost of organising the benchmarking is unavoidable because it involves defining the experiments from scratch, since no previous ones exist.
- *Facilitating result analysis.* Full automation of the result analysis is not possible since this requires a person to interpret them; nevertheless, the automatic generation of different visualizations and summaries of the results in different formats (such as HTML or SVG) allows us to draw some conclusions at a glance.
- *Including new tools easily*, because the effort to be spent in the benchmarking is a main criteria for an organisation when deciding whether to participate in the benchmarking.

[8] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/.

[9] http://knowledgeweb.semanticweb.org/iriba/.

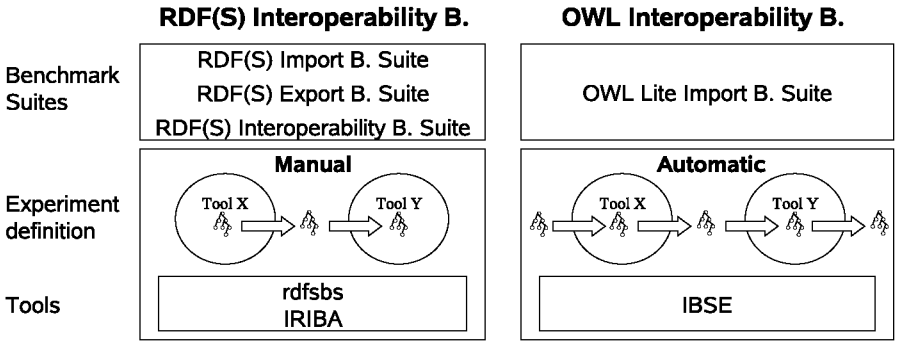


Fig. 3. The UPM framework for benchmarking interoperability.

In our scenario, interoperability depends on two different tool functions, one that reads an ontology stored in the tool and writes it into an OWL file (OWL exporter from now on), and another that reads an OWL file with an ontology and stores this ontology into the tool (OWL importer from now on). Therefore, our experiments provided data not only on the interoperability but also on the OWL importers and exporters of the tools.

To obtain detailed information on tool interoperability using OWL as the interchange language, we need to know (a) the components of the knowledge model of a tool that can be interchanged with others; (b) the secondary effects of interchanging ontologies that include these components, such as insertion or loss of information; (c) the subset of the tools' knowledge models that can be used to correctly interoperate; and (d) the problems that arise when ontologies are interchanged between two tools and the causes of these problems.

Participation in the benchmarking is open to any Semantic Web tool capable of importing and exporting OWL. A public call for participation was issued and many tool developers were directly contacted to participate in it.

Eight tools took part in the benchmarking: one ontology-based annotation tool (GATE[10]), three ontology frameworks (Jena,[11] KAON2,[12] and SWI-Prolog[13]), and four ontology development tools (Protégé Frames,[14] Protégé OWL,[15] SemTalk,[16] and WebODE[17]).

These tools present a variety of knowledge models, which are next enumerated:

- GATE's knowledge model consists of a class hierarchy with an added level of expressivity aimed at being broadly equivalent to OWL Lite [3].
- Jena's knowledge model supports RDF and ontology formalisms built on top of RDF. Specifically this means RDF(S), the varieties of OWL, and the now-obsolete DAML+OIL [26].
- KAON2's knowledge model is capable of manipulating the $\mathcal{SHIQ}(\mathbf{D})$ subset of OWL-DL and F-Logic [27].
- Protégé Frame's knowledge model is based on a flexible metamodel, which is comparable to object-oriented and frame-based systems [28].
- Protégé OWL's knowledge model supports RDF(S), OWL Lite, OWL DL and significant parts of OWL Full [23].
- SemTalk's knowledge model supports modelling RDF(S) and OWL using Visio [12].
- SWI-Prolog's knowledge model supports RDF(S) and OWL on top of Prolog [34].
- WebODE's knowledge model is based in frames and is extracted from the intermediate representations of METHONTOLOGY [1].

6. Experiment definition

As previously mentioned, participation in the benchmarking is open to any Semantic Web tool. Nevertheless, the experiment requires that the tools participating be able to import and export OWL ontologies. This is so because in the experiment we need an automatic and uniform way of accessing the tools, and the operations performed to access the tools must be supported by most of the Semantic Web tools. Due to the high heterogeneity in Semantic Web tools, ontology management APIs vary from one tool to another. Therefore, the way that we chose to automatically access the tools is through the following two operations commonly supported by most Semantic Web tools: to import an ontology from a file, and to export an ontology to a file.

During the experiment, a common group of benchmarks is executed and each benchmark describes one input OWL ontology that has to be interchanged between a single tool and the others (including the tool itself).

Each benchmark execution comprises two sequential steps, shown in Fig. 4. Starting with a file that contains an OWL ontology (O_i), the first step (*Step 1*) consists in importing the file storing the ontology into the origin tool and then exporting the ontology into an OWL file (O_i^{II}). The second step (*Step 2*) consists in importing the file storing the ontology exported by the origin tool (O_i^{II}) into the destination tool and then exporting the ontology into another file (O_i^{IV}).

In these steps, there is no common way for the tools to check how good the importers (by comparing O_i with O_i^I and O_i^{II} with O_i^{III}) and exporters (by comparing O_i^I with O_i^{II} and O_i^{III} with O_i^{IV}) are. We only have the results of combining the import and export operations (the files exported by the tools), so these two operations

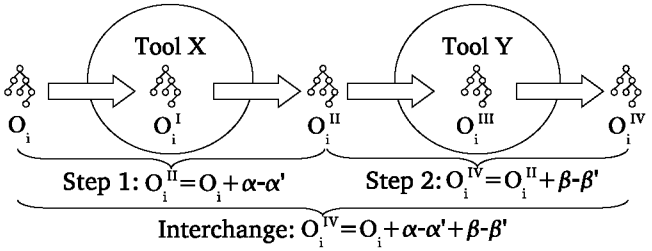


Fig. 4. The two steps of a benchmark execution.

[10] Version 4.0 http://gate.ac.uk/.
[11] Version 2.3 http://jena.sourceforge.net/.
[12] Version 2006-09-22 http://kaon2.semanticweb.org/.
[13] Version 5.6.35 http://www.swi-prolog.org/packages/semweb.html.
[14] Version 3.3 build 395 http://protege.stanford.edu/.
[15] Version 3.3 build 395 http://protege.stanford.edu/overview/protege-owl.html.
[16] Version 2.3 http://www.semtalk.com/.
[17] Version 2.0 build 192 http://webode.dia.fi.upm.es/.

are considered as an atomic operation. It must be noted here that if a problem arises in one of these steps, we cannot know whether it was originated when importing or when exporting the ontology, because we are totally unaware of the state of the ontology inside each tool.

After a benchmark execution, we have three ontologies to compare, namely, the original ontology (O_i), the intermediate ontology exported by the first tool (O_i^{II}), and the final ontology exported by the second tool (O_i^{IV}). From these results, the following evaluation criteria for a benchmark execution can be defined:

- *Execution* (*OK*/*FAIL*/*C.E.*/*N.E.*) informs of the correct execution of a step or of the whole interchange. Its value is *OK* if the step or the whole interchange is carried out with no execution problem; *FAIL* if the step or the whole interchange is carried out with some execution problem; *C.E.* (Comparer Error) if the comparer launches an exception when comparing the original and the final ontologies; and *N.E.* (Not Executed) if the second step is not executed because the first step execution failed.
- *Information added or lost* informs of the information that is added to or lost from the ontology in terms of triples in each step or in the whole interchange. We can know the triples added or lost in *Step 1*, in *Step 2*, and in the whole interchange by comparing the original ontology with the intermediate one, the intermediate ontology with the final one, and the original with the final ontology, respectively. As will be explained in Section 8.3, we state that two ontologies are the same when they are logically equivalent; therefore, the triples added or lost will be those that make the two ontologies logically different.
- *Interchange* (*SAME*/*DIFFERENT*/*NO*) informs whether the ontology has been interchanged correctly with no addition or loss of information. From the previous basic measurements, we can define *Interchange* as a derived measurement that is *SAME* if *Execution* is *OK* and *Information added* and *Information lost* are null; *DIFFERENT* if *Execution* is *OK* but *Information added* or *Information lost* are not null; and *NO* if *Execution* is *FAIL*, *N.E.* or *C.E.*

For evaluating the interoperability of the tools, the OWL Lite Import Benchmark Suite has been used, described in the following section, which is common for all the tools and contains ontologies with simple combinations of OWL Lite components.

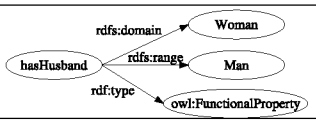
7. The OWL Lite Import Benchmark Suite

The ontologies used in the experiment are those defined for the OWL Lite Import Benchmark Suite and described in detail in [8]. This benchmark suite was intended to evaluate the OWL import capabilities of Semantic Web tools by checking the import of ontologies with simple combinations of components of the OWL Lite knowledge model. It is composed of 82 benchmarks and is available on the Web.[18]

Each benchmark of the benchmark suite, as Table 1 shows, is described by a unique *identifier*, a *description* in natural language, a *formal description* in Description Logics notation of the ontology, a *graphical representation* of the ontology, and a *file* with the ontology in the RDF/XML syntax.

Since the RDF/XML syntax allows serializing ontology components in different ways while maintaining the same semantics, the benchmark suite includes two kinds of benchmarks: one to check the import of the different combinations of the OWL Lite vocabulary terms, and another to check the import of OWL ontologies with the different variants of the RDF/XML syntax. Table 2 shows the groups of the OWL Lite Import Benchmark Suite and the number of benchmarks in each group.

Table 1
The description of a benchmark of the OWL Lite Import Benchmark Suite.

Identifier	**ISG03**
Description	Import a single functional object property whose domain is a class and whose range is another class
Formal description	$\top \sqsubseteq \leq 1\ hasHusband$ $\top \sqsubseteq \forall hasHusband^{-}$.Woman $\top \sqsubseteq \forall hasHusband$.Man
Graph	

RDF/XML file	`...`
	`<owl:Class rdf:about="&ex;Woman"/>`
	`<owl:Class rdf:about="&ex;Man"/>`
	`<owl:ObjectProperty rdf:about="&ex;hasHusband">`
	`<rdf:type rdf:resource="&owl;FunctionalProperty"/>`
	`<rdfs:domain rdf:resource="&ex;Woman"/>`
	`<rdfs:range rdf:resource="&ex;Man"/>`
	`</owl:ObjectProperty>`
	`...`

Table 2
Benchmark groups of the OWL Lite Import Benchmark Suite.

Group	No.
A – class hierarchies	17
B – class equivalences	12
C – classes defined with set operators	2
D – property hierarchies	4
E – properties with domain and range	10
F – relations between properties	3
G – global cardinality constraints and logical property characteristics	5
H – single individuals	3
I – named individuals and properties	5
J – anonymous individuals and properties	3
K – individual identity	3
L – syntax and abbreviation	15
Total	82

The OWL Lite Import Benchmark Suite is here used to evaluate the interoperability of Semantic Web tools. Nevertheless, any group of ontologies could be used as input for the experiment. For example, we could employ a group of real ontologies in a certain domain, ontologies synthetically generated such as the Lehigh University Benchmark (LUBM) [18] or the University Ontology Benchmark (UOB) [24], or the OWL Test Cases [6] (developed by the W3C Web Ontology Working Group).

However, these other ontologies were designed with specific goals and requirements, such as that of performance evaluation or correctness evaluation. Since our goal was to improve interoperability, these ontologies could complement our experiments but, in our circumstances, we aimed at evaluating interoperability with simple OWL ontologies that, even though they do not cover exhaustively the OWL specification, are simple and allow isolating problem causes and highlighting problems in the tools.

8. Experiment execution: the IBSE tool

The experiments to perform in the benchmarking consist in interchanging each of the ontologies of the OWL Lite Import Benchmark Suite between all the tools (including interchanges from one tool to itself) and in collecting the results of these interchanges.

Although the results of the experiment described above could be obtained manually, the goal of the benchmarking is to automate

[18] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/owl/import.html.

all the experimentation. Hence, we need some software application that can perform all the experiments automatically.

IBSE (Interoperability Benchmark Suite Executor) is the evaluation infrastructure that automates the execution of the experiments of the OWL Interoperability Benchmarking. It offers a simple way of executing the experiments between any selected group of tools and of analysing the results and permits smoothly including new tools into the infrastructure.

The IBSE tool has been implemented with Java; its source code and binaries are publicly available and can be downloaded from its web page.[19] The only requirements for executing IBSE are to have both a Java Runtime Environment and the IBSE binaries. The IBSE distribution includes the tools mentioned in this paper, ready to be evaluated; however, to evaluate either SemTalk or WebODE, these tools must be previously installed in the system.

The main requirements taken into account in the development of the IBSE tool surge from the benchmarking requirements described in Section 5, and are the following: (a) to perform the experiments with any tool able to import and export OWL files; (b) to automate the experiment execution and the analysis of the results; (c) to define benchmarks and results through ontologies, since the automation mentioned above requires benchmarks and results to be machine-processable; (d) to use any group of ontologies as input for the experiments; and (e) to separate benchmark execution and report generation.

A normal execution of IBSE comprises the three consecutive steps shown in Fig. 5, although they can also be executed independently.

These steps are the following:

(i) *To generate machine-readable benchmark descriptions from a group of ontologies.* In this step, and from a group of ontologies located in a URI, one RDF file with one benchmark for each ontology is generated.

(ii) *To execute the benchmarks.* In this step, considering all the different combinations of ontology interchanges between the tools, each benchmark described in the RDF file is executed and its results are stored in another RDF file.

To execute a benchmark between an origin tool and a destination one; as described in Section 6, first the file storing the ontology is imported into the origin tool and then exported into an intermediate file and, second, this intermediate file is imported into the destination tool and then exported into the final file.

Once we have the original, intermediate and final files with their corresponding ontologies, we can extract the results by comparing these ontologies. This comparison and its output depend on an external ontology comparer.

(iii) *To generate reports with different visualizations of the results.* In this step, different HTML files are generated with different visualizations, summaries and statistics of the results.

The next sections present some implementation details about how benchmarks and results are represented using ontologies, how tools are inserted in the IBSE tool, and how ontologies are compared.

[19] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/ibse/.

8.1. Representation of benchmarks and results

The IBSE tool employs two OWL ontologies: the *benchmarkOntology*[20] one and the *resultOntology*[21] one, which define the vocabulary for representing the benchmarks and the results of a benchmark execution, respectively. These ontologies are lightweight since their main goal is to be user-friendly.

Figs. 6 and 7 show the graphical representation of the *benchmarkOntology* and of the *resultOntology* ontologies, respectively. Next, the section presents the classes and properties that these ontologies contain. All the datatype properties have as range *xsd:string*, with the exception of *timestamp* whose range is *xsd:dateTime*.

8.1.1. benchmarkOntology ontology

The *Document* class represents a document containing one ontology. A document can be further described by properties that have *Document* as domain. Such properties are the following: *documentURL* (the URL of the document), *ontologyName* (the ontology name), *ontologyNamespace* (the ontology namespace), and *representationLanguage* (the language used to implemented the ontology).

The *Benchmark* class represents a benchmark to be executed. A benchmark can be further described with properties that have *Benchmark* as domain. Such properties are the following: *id* (the benchmark identifier), *usesDocument* (the document that contains one ontology used as input), *interchangeLanguage* (the interchange language used), *author* (the benchmark author), and *version* (the benchmark version number).

8.1.2. resultOntology ontology

The *Tool* class represents a tool that has participated as origin or destination of an interchange in a benchmark. A tool can be further described with properties that have *Tool* as domain. Such properties are the following: *toolName* (the tool name) and *toolVersion* (the tool version number).

The *Result* class represents a result of one step or of the whole benchmark execution. A result can be further described with properties that have *Result* as domain. Such properties are the following: *execution* (if the whole interchange, the first step or the second step are carried out without any execution problem), *informationAdded* (the triples added in the whole interchange, in the first step, or in the second step), *informationRemoved* (the triples removed in the whole interchange, in the first step, or in the second step), and *interchange* (if the ontology has been interchanged correctly from the origin tool to the destination tool, in the first step or in the second step with no addition or loss of information).

The *BenchmarkExecution* class represents a result of a benchmark execution. A benchmark execution can be further described with properties that have *BenchmarkExecution* as domain. Such properties are the following: *ofBenchmark* (the benchmark to which the result corresponds), *originTool* (the tool origin of the interchange), *destinationTool* (the tool destination of the interchange), and *timestamp* (the date and time when the benchmark is executed).

8.2. Inserting tools in the evaluation infrastructure

As the experiment requires no human intervention, we can only run the tools by accessing them through application programming

[20] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/owl/benchmarkOntology.owl.

[21] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/owl/resultOntology.owl.

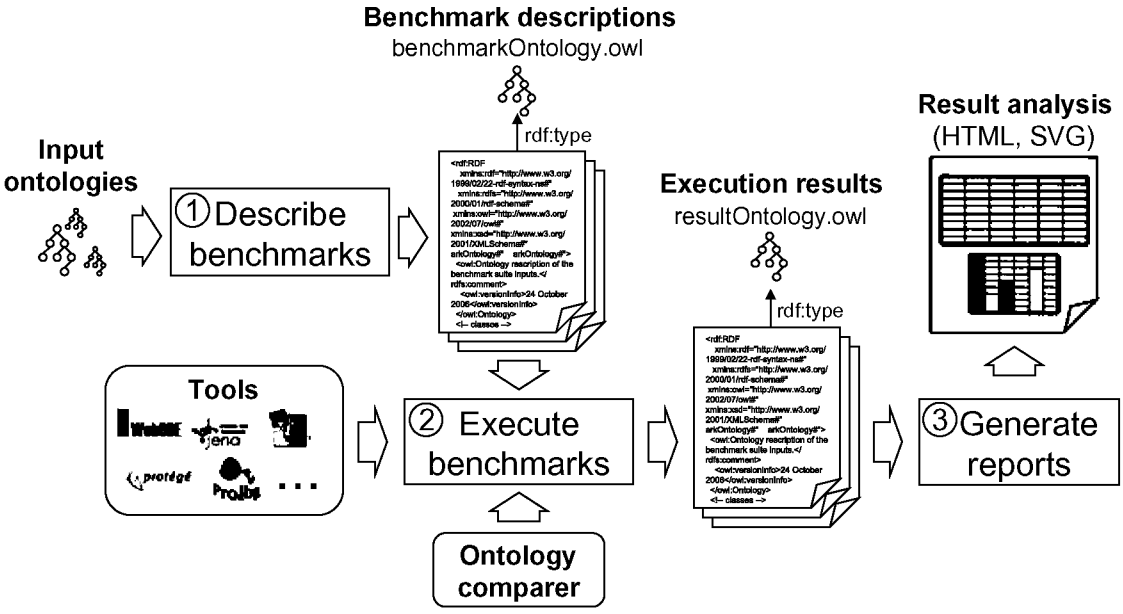


Fig. 5. Automatic experiment process.

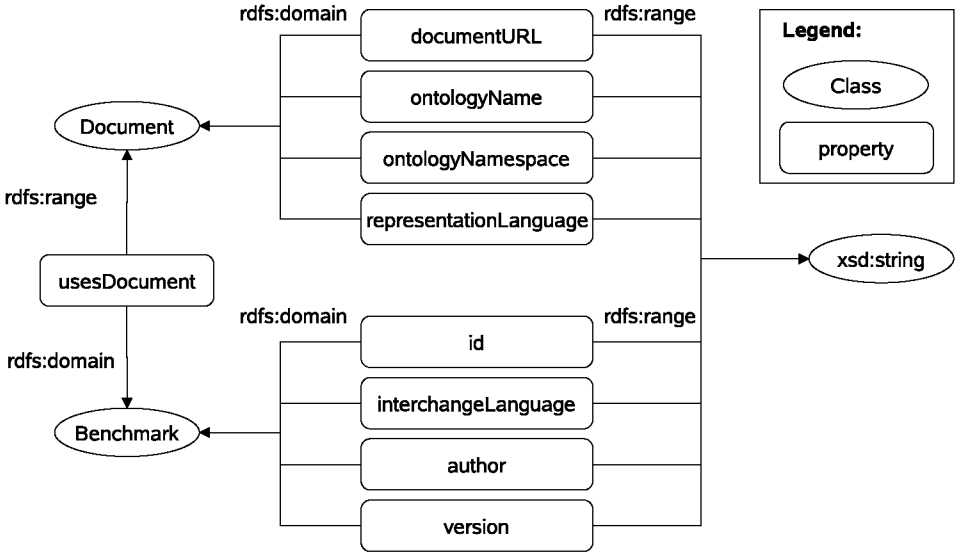


Fig. 6. Graphical representation of the *benchmarkOntology* ontology.

interfaces (APIs) or through batch executions. There are other ways of executing an application automatically (e.g., Web Service executions) but they are not present in the current tools. Nevertheless, to adapt the IBSE tool in order to include other types of executions should be quite straightforward.

Inserting a tool in the evaluation infrastructure is quite easy, it can be performed either by implementing a Java interface in IBSE or by building a program that imports an ontology from a file and exports the imported ontology into another file.

Most of the tools have implemented the Java interface since they provide Java methods for performing the import and export operations. With non-Java tools (SemTalk and SWI-Prolog), these operations are performed by executing precompiled binaries.

8.3. Comparing OWL ontologies

In our experiment, we need to automatically compare ontologies from the lexical to the conceptual level (see Section 2.2) and we state that two ontologies are the same when they are logically equivalent (i.e., each of them entails the other). We do not aim to compare ontologies at the pragmatic level; finding these differences automatically is not possible because it requires a human to interpret the ontologies.

As mentioned above, the IBSE tool uses external software for comparing the ontologies resulting from the experiment. To be used within IBSE, an ontology comparer just has to implement a Java interface.

After researching existing tools, we found no tool able of comparing two OWL ontologies and of providing the differences between them, as required for our case. Therefore, we aimed for a combined solution using several tools.

We decided to use Jena and Pellet[22] because they are freely available and provide well documented programming interfaces.

[22] Version 1.5.2 http://pellet.owldl.com/.

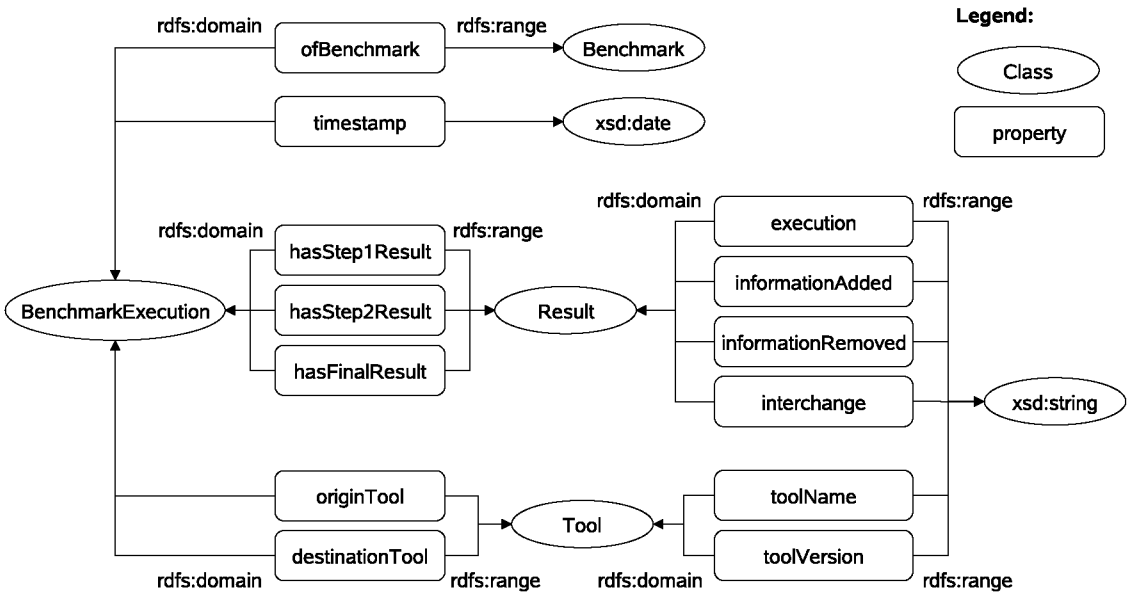


Fig. 7. Graphical representation of the *resultOntology* ontology.

Moreover, we had to use both Jena and Pellet because in some cases they do not compare OWL ontologies correctly and in specific cases they have execution problems.

This way, Pellet provides a comparison at the conceptual level and Jena provides a structural (isomorphic) comparison that is faster than the previous one, syntax-checking capabilities, and the ability of extracting the differences between two ontologies.

The process followed for comparing two OWL ontologies (*O1* and *O2*) combining Jena and Pellet is described below. We can see that it is a process that depends on the deficiencies of the tools for performing the required task.

(i) To check with Jena that the ontologies are syntactically valid.
(ii) To check with Jena if the ontologies are isomorphic and preserve edge and node labels. If *O1* is isomorphic with *O2*, they are structurally identical and, therefore, equivalent.
(iii) To check with Pellet if the ontologies are entailed by the other. If *O1* entails *O2* and *O2* entails *O1*, they are equivalent.
(iv) To extract with Jena the differences between the ontologies (*diff1* and *diff2*). Jena extracts these differences as RDF graphs.
(v) To check with Jena if the graphs with the differences between the ontologies are isomorphic. If *diff1* is isomorphic with *diff2*, the ontologies are equivalent.

During the process, we also check for the following issues when the tools compare ontologies:

- In some cases, Jena and Pellet say that one literal value and the same literal value with a *xsd:string* datatype (e.g., "Peter" and "Peter"^^<xsd:string>) are different. But according to the RDF Datatype entailment rules,[23] these literals are equivalent.
- Jena considers that two ontologies named "onto" and "onto#" are different.
- Pellet considers that two ontologies entail each other when one has the "*ontName rdf:type owl:Ontology.*" triple and the other does not have it; however, we want to detect if ontology names are lost during interchanges.

Furthermore, since any software used for ontology comparison could have execution problems or affect the experiment results, we have evaluated it in two steps:

(i) The interoperability experiment was carried out with the tools participating in the benchmarking whose knowledge model is the same as that of the interchange language. In theory, these tools should interchange all the ontologies correctly because no ontology translation is required for doing so.
(ii) The interoperability experiment was carried out with all the tools participating in the benchmarking. In this step, we analysed if the comparison of two ontologies caused an execution error in the comparer.

Although we did not make an exhaustive evaluation of the ontology comparer, after analysing all the benchmarking results, we found no errors in it.

9. OWL compliance results

Once the IBSE tool was adapted to include all the tools participating in the benchmarking, the experiments were performed using the ontologies from the OWL Lite Import Benchmark Suite. As mentioned in Section 5, results were obtained for eight tools: GATE, Jena, KAON2, Protégé-Frames, Protégé-OWL, SemTalk, SWI-Prolog, and WebODE.

A detailed analysis of the interoperability results, including results specific for each tool, can be found at [13]. The HTML and RDF files generated by the IBSE tool are available on the Web.[24]

Since the OWL interoperability results highly depend on the compliance of the tools with the OWL specification and because of the large number of benchmark executions,[25] the analysis of the interoperability of the tools is divided into two consecutive steps:

(i) The analysis of the compliance of the tools with the OWL specification, taking into account the results of the tool when

[23] http://www.w3.org/TR/rdf-mt/#DtypeRules.

[24] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/owl/2008-07-06_Results/.

[25] For 8 tools we have 64 possible interoperability scenarios, each composed of 82 benchmark executions, which results in 5.248 benchmark executions.

Table 3
Results in *Step 1* (for 82 benchmarks).

	GA	JE	K2	PF	PO	ST	SP	WE
Same	76	82	48	4	82	34	82	26
More			11					14
Less	6		23	78		48		42
Tool fails								

managing OWL ontologies in the combined operation of importing an OWL ontology and exporting it again (a step of the experiment, as defined in Section 6). This analysis is included in this section.

(ii) The analysis of the OWL interoperability of the tools with all the tools participating in the benchmarking (including the tool itself). This analysis is presented in Section 10.

To analyse the OWL compliance of the tools, we have taken into account the tool results when such tool is the origin of the interchange (*Step 1*), irrespective of the tool being the destination of the interchange. This step has as input an original ontology that is imported by the tool (O_i) and then exported into a resultant ontology (O_i^{II}). This analysis has been performed by comparing the original and the resultant ontologies.

This is not an exhaustive evaluation of the OWL compliance because the ontologies used in the experiments belong to the OWL Lite sublanguage and do not represent every possible combination of ontology components. However, it provides useful information about the behaviour of the tools when dealing with OWL ontologies.

Table 3 presents the results of a step execution for each tool.[26] It shows the number of benchmarks in each category in which these results can be classified:

- *The original and the resultant ontologies are the same.* The only tools that always produce the same ontologies are Jena, Protégé OWL and SWI-Prolog. The other tools in some cases insert and remove information when importing and exporting.
- *The resultant ontology includes more information than the original one.* This only happens with KAON2, since it inserts the triple "*rdfs:Literal rdf:type owl:Datatype.*" when the ontology contains datatype properties, and with WebODE, since it inserts anonymous classes for modelling multiple domains or ranges in properties and inserts value constraints in classes that are the domain of datatype properties.
- *The resultant ontology includes less information than the original one.* In this case, information is sometimes inserted into the resultant ontology.
- *The execution fails in the import and export operation.* The tools do not have execution problems.

Table 4 is a breakdown of the row "*Same*" in Table 3, according to the combination of components present in the ontology; it shows the number of benchmarks in each group and the percentage of benchmarks whose original (O_i) and resultant (O_i^{II}) ontologies are the same in *Step 1*. It can be observed that some tools work better with some component combinations than with others.

If we classify the mismatches found when comparing the original (O_i) and the resultant (O_i^{II}) ontologies according to the ontology heterogeneity levels described in Section 2.2, we can see that mismatches are found in all the levels except in the *Lexical* and *Paradigm* ones, being most of them in the *Conceptual* level. Besides, in some cases mismatches occur in two levels (e.g., in the *Syntactic* and *Conceptual* levels).

Next, we show for each level the tools that present mismatches and the causes of these mismatches:

- *Pragmatic level.* Mismatches found at this level occur when the two ontologies are semantically equivalent but their interpretation may be different in different contexts (e.g., an ontology contains two classes that are subclass of the other, thus forming a cycle, and another ontology contains two classes that are equivalent). The tools with mismatches at this level are Protégé-Frames, Protégé-OWL, SemTalk, and WebODE.
- *Conceptual level.* Mismatches found at this level are due to differences in conceptualizations and are the most frequent in the tools. The tools with mismatches at this level are GATE, KAON2, Protégé-Frames, SemTalk, and WebODE.
- *Terminological level.* The tools with mismatches at this level are Protégé-Frames, which changes the names of ontologies, classes, properties, instances, and gives name to anonymous individuals; SemTalk, which loses the name of the ontology; and WebODE, which gives name to anonymous individuals.
- *Syntactic level.* The tools with mismatches at the *Syntactic* level are GATE, and Protégé-Frames because they change the ontology into OWL Full, and WebODE because it redefines datatypes.

In GATE, the change to OWL Full occurs because it does not explicitly define classes as *owl:Class*. In Protégé-Frames, the change to OWL Full occurs because it generates RDF properties (*rdf:Property*) instead of OWL datatype properties (*owl:DatatypeProperty*). In WebODE, mismatches are due to the redefinement of datatypes or the insertion of them in values that were not typed.

10. OWL interoperability results

With the previous information about the OWL compliance of the tools, we provide the analysis of their interoperability with all the tools participating in the benchmarking. In this analysis we consider all the tools because when in *Step 1* a tool produces an ontology different from the original one, this tool may be working correctly, as intended by its developers.

To analyse the interoperability between two tools (i.e., T1 and T2), we have considered the interchange from one tool to another (from T1 to T2) and vice versa (from T2 to T1).

Table 5 provides an overview of the interoperability between the tools; it shows the percentage of benchmarks in which the original (O_i) and the resultant (O_i^{IV}) ontologies in an interchange are the same. For each cell, the row indicates the tool origin of the interchange and the column indicates the tool destination of the interchange.

The first thing that can be glanced at is that the interoperability between the tools is low, even in interchanges between a tool and the tool itself.

It is also clear from the results that interoperability using OWL as the interchange language depends on the knowledge model of the tools, and that the more similar the knowledge model of a tool is to OWL, the more interoperable the tool is. Nevertheless, the way of serializing the ontologies in the RDF/XML syntax also has a high influence in the results, as the interoperability issues identified above show.

Correctly working tool importers and exporters do not ensure interoperability. In Section 9 we saw that Jena, Protégé-OWL and SWI-Prolog always produced the same ontologies in *Step 1*, but not all of these tools also produce the same ontologies after interchanging them. Interchanges between Jena and Protégé-OWL and

[26] The tool names have been abbreviated in the tables: GA=GATE, JE=Jena, K2=KAON2, PF=Protégé Frames, PO=Protégé OWL, ST=SemTalk, SP=SWI-Prolog, and WE=WebODE.

Table 4
Percentage of identical ontologies per group in *Step 1*.

Benchmark group	GA	JE	K2	PF	PO	ST	SP	WE
A – class hierarchies	53	100	71	0	100	41	100	29
B – class equivalences	75	100	75	0	100	0	100	0
C – classes defined with set operators	100	100	100	0	100	100	100	0
D – property hierarchies	75	100	50	0	100	50	100	25
E – properties with domain and range	60	100	70	0	100	90	100	40
F – relations between properties	100	100	67	0	100	67	100	0
G – global cardinality constraints and logical property characteristics	80	100	80	0	100	60	100	60
H – single individuals	0	100	100	0	100	67	100	67
I – named individuals and properties	40	100	60	0	100	80	100	60
J – anonymous individuals and properties	67	100	0	0	100	0	100	0
K – individual identity	33	100	100	0	100	0	100	0
L – syntax and abbreviation	60	100	7	27	100	13	100	53

Table 5
Percentage of identical ontologies after the interchange.

		Destination							
		JE	PO	SP	GA	K2	ST	WE	PF
Origin	JE	100	100	100	70	58		31	4
	PO	100	100	95	78	58		31	4
	SP	100	100	100	91	58	46	31	4
	GA	92	92	75	60	56		29	25
	K2	58	58	58	67	58	45	19	13
	ST	41	41	46	39	36	40	34	
	WE	31	31		29	19	20	31	20
	PF	4	4		3			4	4

Table 6
Percentage of identical interchanged ontologies for *Class hierarchies*.

		Destination							
		JE	PO	SP	GA	K2	ST	WE	PF
Origin	JE	100	100	100	82	71		29	
	PO	100	100	100	88	71		29	
	SP	100	100	100	88	71	41	29	
	GA	88	88	88	53	71		29	41
	K2	71	71	71	59	71	47	24	35
	ST	41	41	65	41	29	41	29	
	WE	29	29		29	18	29	29	29
	PF								

interchanges between Jena and SWI-Prolog do produce the same ontologies. In the interchanges between Protégé-OWL and SWI-Prolog, when the interchange is from SWI-Prolog to Protégé-OWL the ontologies produced are the same, but when the interchange is from Protégé-OWL to SWI-Prolog some problems arise.[27]

This leads us to a second fact, that interoperability between two tools is usually different depending on the direction of the interchange. This can be clearly observed in Table 5 and in the previous example about Protégé-OWL and SWI-Prolog.

If we classify the mismatches found when comparing the original (O_i) and the final (O_i^{IV}) ontologies according to the ontology heterogeneity levels described in Section 2.2, we can draw the same conclusions as those for the OWL compliance of the tools presented in Section 9. The only comment we can add is that mismatches in the first step cause further mismatches in the second step and even tool failure, as we will show later in the robustness results.

Furthermore, it cannot be said that there is a group of "typical" interoperability problems, since the interoperability results highly depend on the tools that participate in the interchange and, on the other hand, the behaviours of each tool are quite different.

To analyse the interoperability of the tools regarding the combination of components present in the ontology and to know in which cases interchanges can be performed in one direction but not in both, we have analysed for each group of the OWL Lite Import Benchmark Suite the percentage of benchmarks in which the original (O_i) and the resultant (O_i^{IV}) ontologies in an interchange are the same. Table 6 provides an example of these results for the *Class hierarchies* group.[28]

We have also identified the clusters of interoperable tools according to the OWL interoperability results. Table 7 shows the percentage of benchmarks in which the original (O_i) and the resultant (O_i^{IV}) ontologies in an interchange are the same, according to the combination of components present in the ontology. Each column shows the average of the percentages for every tool in the cluster and in all directions.[29] From left to right, the table shows a cluster with all the tools and successive clusters which have removed the tool with lower percentages from the previous cluster.

In the table we can see that Jena, Protégé-OWL, and SWI-Prolog can interchange correctly all the component combinations; how-

[27] SWI-Prolog produces ontologies with an incorrect namespace identifier (*[]*) when it imports ontologies that contain default namespaces (*xmlns="namespaceURI"*).

[28] Tables with detailed results for each group can be found in http://knowledgeweb.semanticweb.org/benchmarking_interoperability/owl/2008-07-06_Results/per_group.html.

[29] i.e., For a cluster of two tools, A and B, we considered: A to B, B to A, A to A, and B to B.

Table 7
Percentage of identical interchanged ontologies per group.

Group	All	GA, JE, K2, PF, PO, ST, SP	GA, JE, K2, PO, ST, SP	GA, JE, K2, PO, SP	JE, K2, PO, SP	JE, PO, SP	JE, PO	JE, SP
Class hierarchies	42	48	64	83	88	100	100	100
Class equivalences	35	45	60	83	90	100	100	100
Classes defined with set operators	50	64	86	100	100	100	100	100
Property hierarchies	39	46	61	77	80	100	100	100
Properties with domain and range	48	54	72	83	88	100	100	100
Relations between properties	42	54	71	87	87	100	100	100
Global cardinality constraints and logical property characteristics	47	50	67	83	92	100	100	100
Single individuals	47	49	65	76	100	100	100	100
Named individuals and properties	46	49	65	76	84	100	100	100
Anonymous individuals and properties	23	30	40	56	60	100	100	100
Individual identity	38	49	65	91	100	100	100	100
Syntax and abbreviation	35	35	42	59	60	96	100	100

Table 8
Percentage of benchmarks in which tool execution fails in *Step 2*.

		Destination							
		GA	JE	K2	PF	PO	ST	SP	WE
Origin	GA				2	2	37		
	JE						37		
	K2			9	9	9			5
	PF			22			18		
	PO						37		
	ST	12		12					
	SP								
	WE								

ever, since some problems appear when Protégé-OWL interchanges ontologies with SWI-Prolog in the *Syntax and abbreviation* benchmarks, the only two clusters of fully-interoperable tools are Jena with Protégé-OWL and Jena with SWI-Prolog. If KAON2 or GATE are included in these clusters, correct interchanges can only be achieved with few combinations of components.

Finally, regarding the robustness of the tools, as mentioned above tools have no execution problems when processing the ontologies of the benchmark suite in the first step of the experiment, but some of them do have problems when processing ontologies generated by other tools. Table 8 provides an overview of the robustness of the tools and shows the percentage of benchmarks in which tool execution fails in the second step.

All the ontologies that make these tools fail are syntactically valid. In some cases the ontologies are OWL Lite ontologies; however, in other cases they are OWL Full ontologies or they present modelling errors (e.g., to use as the object of the *rdfs:subClassOf* property a blank node that is not constrained in the rest of the ontology or to use *rdf:Property* instead of *owl:ObjectProperty* or *owl:DatatypeProperty*). Needless to say, this lack of robustness in the tools also has a negative effect in interoperability.

11. Conclusions and future work

In this paper we show that it is feasible to evaluate different Semantic Web technologies using a common method and following a problem-focused approach instead of a tool-focused approach; our evaluation focused on the interoperability problem instead of on a certain type of technology.

Semantic Web technologies are constantly evolving and their interoperability is expected to change over time. Therefore, this paper is intended to serve not just as a summary of the OWL Interoperability Benchmarking and its results, but as a guide to perform interoperability evaluations over Semantic Web technologies that allows a continuous evaluation of such technologies.

To this end, the OWL Lite Import Benchmark Suite and the IBSE tool, presented in the paper, support a large-scale automatic evaluation of the OWL interoperability of Semantic Web technologies in a cheap, flexible and extensible way.

Since the interoperability experiment and the IBSE tool are independent of the interchange language, they can be used in other scenarios using any group of ontologies as input or other languages as interchange. Right now the tool allows performing experiments using RDF(S) as the interchange language and *rdf-utils*[30] as ontology comparer; in this case, the RDF(S) Import Benchmark Suite [16] could be used as input for the evaluation.

The assessment of the OWL interoperability of eight well-known Semantic Web tools has provided us with detailed results of the behaviour of the tools not just regarding their interoperability with other tools but also regarding their OWL compliance. These results are publicly available on the Web in HTML and in RDF so anyone can use them.

Publishing evaluation results in RDF will allow not only to automatically reuse these results but also to integrate them with evaluation results that cover other characteristics (e.g., scalability, usability) in order to produce recommendations over semantic technologies that support semantic technology selection.

The main conclusion drawn from the interoperability results is that in general interoperability between the tools is very low, even in interchanges between a tool and the tool itself, and the clusters of interoperable tools are minimal.

Furthermore, *interoperability using an interchange language highly depends on the knowledge models of the tools.* This said, we can add that interoperability is better when the knowledge model of the tools is similar to that of the interchange language. This can be seen in the results where the tools that better interoperate are those whose knowledge models fully cover the knowledge model of the interchange language. In the cases where the knowledge

[30] Version 0.3b http://wymiwyg.org/rdf-utils/.

models differ, interoperability can be only achieved by means of lightweight ontologies.

This interoperability panorama, although disappointing, may serve to promote the second of our goals: the improvement of the tools. Even though tool improvement is out of our scope right now because each tool is developed by independent organisations, we hope, nevertheless, that the results provided may help improve the tools. We can add that, since our goal was improvement, modifications on the participating tools were allowed at any time and, in some cases, tools were improved while the experiments were being executed.

Therefore, real interoperability in the Semantic Web requires the involvement of tool developers. The developers of the tools participating in the benchmarking activities have been informed of the results of these activities and of the recommendations proposed for improving their tools.

After analysing the results, we checked that the interoperability problem not only depends on the ontology translation problem but also on some robustness and specification problems. In some cases interoperability problems are due to the representation formalisms managed by the tools, but in others they are due to defects in the tools or to the way of serializing ontologies, having the latter a high impact in interoperability.

In addition, some tools instead of parsing ontologies on their own, reuse ontology management frameworks for this task. In these cases, while there is a 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 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. For example, the version of Protégé OWL that we evaluated uses the Jena ontology framework and we can see in Table 8 that, while Jena does not have any execution problem in the second step of the experiment, Protégé OWL does.

Considering the large number of existing Semantic Web tools, only a fraction participated in the benchmarking activity presented in this paper. We think that it would be desirable for the future to continue these benchmarking activities with a higher number of tools.

In future iterations of the benchmarking activity, the evaluation could be updated to be more exhaustive, to cover other OWL sublanguages (DL, Full or OWL 2[31]), to include real-world ontologies, or to focus on user interoperability needs.

Additionally, although we have already mentioned in Section 7 that we cannot directly reuse test cases from other evaluations (e.g., the OWL 1 and OWL 2 Test Cases), we could extend this work by including those ontologies from existing test cases that are suitable for evaluating interoperability.

To increase the usability of the benchmarking results, the main improvement would be to facilitate effective ways of analysing and exploiting the results by means of a web application, such as the IRIBA application used in the RDF(S) Interoperability Benchmarking, so that users could perform dynamic and personalised analyses of the OWL interoperability results over time.

It would also be quite convenient that the IBSE tool provided results that were easier to analyse and that included specific visualizations of results for tools whose internal knowledge model does not correspond with the interchange language. With such tools, the analysis of the results is not straightforward and sometimes triples are inserted or removed as intended by their developers, but this correct functioning is difficult to evaluate or to distinguish in the current results.

[31] http://www.w3.org/TR/owl2-overview/.

Acknowledgments

This work is partially supported by a FPI grant from the Spanish Ministry of Education (BES-2005-8024), by the IST project Knowledge Web (FP6-507482) and by the CICYT project Infraestructura tecnológica de servicios semánticos para la web semántica (TIN2004-02660).

Thanks to all the people that have participated in the OWL Interoperability Benchmarking by adapting the IBSE tool for some best-in-class Semantic Web tools: Stamatia Dasiopoulou, Danica Damljanovic, Michael Erdmann, Christian Fillies, Roman Korf, Diana Maynard, Jesús Prieto-González, York Sure, Jan Wielemaker, and Philipp Zaltenbach. Without their effort, this could have not been possible.

Thanks to Rosario Plaza for reviewing the grammar of this paper.

References

[1] J. Arpírez, O. Corcho, M. Fernández-López, A. Gómez-Pérez, WebODE in a nutshell, AI Magazine 24 (3) (2003) 37–47.
[2] J. Barrasa, Modelo para la definición automática de correspondencias semánticas entre ontologías y modelos relacionales, Ph.D. thesis, Universidad Politécnica de Madrid. Facultad de Informática, January 2007.
[3] K. Bontcheva, V. Tablan, D. Maynard, H. Cunningham, Evolving GATE to meet new challenges in language engineering, Natural Language Engineering 10 (3–4) (2004) 349–373.
[4] P. Bouquet, M. Ehrig, J. Euzenat, E. Franconi, P. Hitzler, M. Krötzsch, L. Serafini, G. Stamou, Y. Sure, S. Tessaris, D2.2.1 Specification of a common framework for characterizing alignment, Tech. rep., Knowledge Web, December 2004.
[5] R. Brachmann, H. Levesque, Readings in Knowledge Representation, chap A Fundamental Tradeoff in Knowledge Representation and Reasoning, Morgan Kaufmann, San Mateo, 1985, pp. 31–40.
[6] J. Carroll, J.D. Roo (Eds.), OWL Web Ontology Language Test Cases, Tech. rep., W3C, February 2004.
[7] O. Corcho, A Layered Declarative Approach to Ontology Translation with Knowledge Preservation, vol. 116 of Frontiers in Artificial Intelligence and its Applications, IOS Press, 2005.
[8] S. David, R. García-Castro, A. Gómez-Pérez, Defining a benchmark suite for evaluating the import of OWL Lite ontologies, in: Proceedings of the OWL: Experiences and Directions 2006 workshop (OWL2006), Athens, Georgia, USA, 2006.
[9] D. Dou, D. McDermott, P. Qi, Ontology translation on the Semantic Web, Journal of Data Semantics 2 (3360) (2004) 35–57.
[10] E. Duval, Learning technology standardization: making sense of it all, International Journal on Computer Science and Information Systems 1 (1) (2004) 33–43.
[11] J. Euzenat, P. Shvaiko, Ontology Matching, Springer-Verlag, Heidelberg (DE), 2007.
[12] C. Fillies, F. Weichhardt, Semantically correct Visio drawings, in: Proceedings of the Workshop on User Aspects of the Semantic Web (UserSWeb2005), 2005, pp. 85–92.
[13] R. García-Castro, S. David, J. Prieto-González, D1.2.2.1.2 Benchmarking the interoperability of ontology development tools using OWL as interchange language, Tech. rep., Knowledge Web, September 2007.
[14] R. García-Castro, A. Gómez-Pérez, Guidelines for benchmarking the performance of ontology management APIs, in: Proceedings of the 4th International Semantic Web Conference (ISWC2005), No. 3729 in LNCS, Galway, Ireland, 2005, pp. 277–292.
[15] R. García-Castro, A. Gómez-Pérez, Semantic Web Engineering in the Knowledge Society, chap Benchmarking in the Semantic Web, IGI Global, 2008.
[16] R. García-Castro, A. Gómez-Pérez, Y. Sure, Benchmarking the RDF (S) interoperability of ontology tools, in: Proceedings of the Nineteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2007), Boston, USA, 2007, pp. 410–415.
[17] T. Gruber, A translation approach to portable ontology specifications, Knowledge Acquisition 5 (2) (1993) 199–220.
[18] Y. Guo, Z. Pan, J. Heflin, LUBM: a benchmark for OWL knowledge base systems, Journal of Web Semantics 3 (2) (2005) 158–182.
[19] J. Hammer, D. McLeod, An approach to resolving semantic heterogeneity in a federation of autonomous, heterogeneous database systems, Journal for Intelligent and Cooperative Information Systems 2 (1) (1993) 51–83.
[20] IEEE-STD-610, ANSI/IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology, IEEE, 1991.
[21] W. Kim, J. Seo, Classifying schematic and data heterogeneity in multidatabase systems, Computer 24 (12) (1991) 12–18.
[22] M. Klein, Combining and relating ontologies: an analysis of problems and solutions, in: Proceedings of the Workshop on Ontologies and Information Sharing (IJCAI2001), Seattle, USA, 2001.

[23] H. Knublauch, R. Fergerson, N. Noy, M. Musen, The Protégé OWL plugin: an open development environment for Semantic Web applications, in: Proceedings of the 3rd International Semantic Web Conference (ISWC2004), vol. 3298, Springer, 2004, pp. 229–243.
[24] L. Ma, Y. Yang, Z. Qiu, G. Xie, Y. Pan, S. Liu, Towards a complete OWL ontology benchmark, in: Proceedings of the 3rd European Semantic Web Conference (ESWC 2006), vol. 4011 of LNCS, Budva, 2006, pp. 125–139.
[25] D. Maynard, Benchmarking textual annotation tools for the Semantic Web, in: Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), European Language Resources Association (ELRA), Marrakech, Morocco, 2008.
[26] B. McBride, Jena: Implementing the RDF Model and Syntax Specification, in: Proceedings of the Second International Workshop on the Semantic Web (SemWeb2001), 2001.
[27] B. Motik, U. Sattler, A comparison of reasoning techniques for querying large description logic ABoxes, in: Proceedings of the 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR2006), Phnom Penh, Cambodia, 2006.
[28] N. Noy, R. Fergerson, M. Musen, The knowledge model of Protégé-2000: Combining interoperability and flexibility, in: Proceedings of the 2th International Conference on Knowledge Engineering and Knowledge Management (EKAW2000), Juan-les-Pins, France, 2000.
[29] OntoWeb, OntoWeb Deliverable 1.3: a survey on ontology tools, Tech. rep., OntoWeb Thematic Network, May 2002.
[30] A. Sheth, Interoperating Geographic Information Systems, chap. Changing Focus on Interoperability in Information Systems: From System, Syntax Structure to Semantics, Kluwer, 1998, pp. 5–30.
[31] Y. Sure, O. Corcho (Eds.), Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003), vol. 87 of CEUR-WS, Florida, October 2003.
[32] V. Tamma, An ontology model supporting multiple ontologies for knowledge sharing, Ph.D. thesis, University of Liverpool, 2001.
[33] P. Visser, D. Jones, T. Bench-Capon, M. Shave, An analysis of ontological mismatches: heterogeneity versus interoperability, in: AAAI 1997 Spring Symposium on Ontological Engineering, Stanford, USA, 1997.
[34] J. Wielemaker, Z. Huang, L. van der Meij, SWI-Prolog and the Web, Theory and Practice of Logic Programming, 2008, pp. 1–30.