

# RDF(S) INTEROPERABILITY RESULTS FOR SEMANTIC WEB TECHNOLOGIES

RAÚL GARCÍA-CASTRO and ASUNCIÓN GÓMEZ-PÉREZ

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

Received 31 October 2007

Revised 15 December 2008

Accepted 13 January 2009

Interoperability among different development tools is not a straightforward task since ontology editors rely on specific internal knowledge models which are translated into common formats such as RDF(S). This paper addresses the urgent need for interoperability by providing an exhaustive set of benchmark suites for evaluating RDF(S) import, export and interoperability. It also demonstrates, in an extensive field study, the state-of-the-art of interoperability among six Semantic Web tools. From this field study we have compiled a comprehensive set of practices that may serve as recommendations for Semantic Web tool developers and ontology engineers.

*Keywords:* RDF(S); interoperability; benchmarking; benchmark suite.

## 1. Introduction

Ontologies enable interoperability among heterogeneous applications. The development and deployment of ontologies and ontology-based applications follows methodological guidelines and is supported by Semantic Web tools such as ontology editors and repositories.

Ideally, one could use all existing Semantic Web technologies seamlessly and thus benefit from all the functionalities they offer. However, as shown in previous workshops on Evaluation of Ontology-based Tools (EON), interoperability among different Semantic Web technologies is not straightforward. For instance, ontology editors usually rely on specific internal knowledge models that have to be translated into common formats, such as RDF(S) [1].

Finding out why interoperability fails is cumbersome and not at all trivial, since any assumption made for the translation within one tool may easily prevent successful interoperability with other tools. Furthermore, not to be aware of the interoperability capabilities of the existing Semantic Web tools may cause important

problems when more complex tools and applications are built reusing existing tools; this ignorance regarding interoperability is mainly due to the fact that tool interoperability has not still been evaluated because we do not have an easy way of making this evaluation.

This paper addresses the urgent need for interoperability evaluation. We provide three exhaustive benchmark suites for evaluating the RDF(S) import, export and interoperability of Semantic Web technologies. These benchmark suites were designed to support the evaluation and improvement of Semantic Web tool interoperability and have been developed as part of the EU IST Knowledge Web Network of Excellence.<sup>a</sup>

In an extensive field study we have explored the state-of-the-art of interoperability among six Semantic Web tools. Three of the tools are ontology editors (KAON, Protégé, WebODE), whereas the other three are repositories (Corese, Jena, Sesame), thus we have covered a wide range of tool support for ontology development and deployment. The field study has helped us gain a deep understanding of the import and export functionalities of Semantic Web tools. Our findings may serve as guidelines for developing tools and are summarised in comprehensive best practices on interoperability.

Our work can benefit *Semantic Web tool developers* since it provides guidelines to design their import and export functionalities and a concrete set of benchmarks against which they can evaluate their import and export functionalities. Our work can also benefit *ontology engineers* since it provides an overview of to which extend interoperability is ensured when combining specific tools. It is expected that future generations of Semantic Web tools will provide smoother interoperability and thus fulfil the key promise of ontologies.

This paper is structured as follows: Sec. 2 presents the motivation behind benchmarking software, a description of the interoperability problem treated in this paper, and other interoperability evaluation initiatives. Section 3 examines how the RDF(S) Interoperability Benchmarking was conducted and how the RDF(S) benchmark suites were designed. Sections 4 and 5 summarise the results of executing the export, import and interoperability benchmarks. Section 6 provides the recommendations extracted from benchmarking for Semantic Web tool developers, ontology engineers, and all those interested in carrying out a benchmarking activity. Finally, Sec. 7 presents the conclusions derived from this work and future lines of work.

## 2. Related Work

### 2.1. *Evaluation vs. benchmarking*

According to the ISO 14598 standard [2], software evaluation is *the systematic examination of the extent to which an entity is capable of fulfilling specified requirements*,

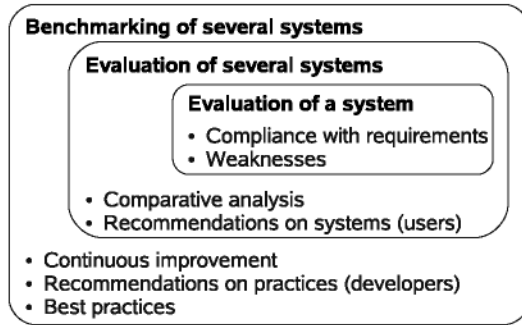


Fig. 1. Benchmarking benefits.

considering software not just as a set of computer programs but also as the produced procedures, documentation and data.

The idea of benchmarking as a process that searches for improvement and best practices derives from the idea of benchmarking in the business management community [3, 4]. This notion of benchmarking can be found in some Software Engineering approaches [5] but it differs from those in which benchmarking is viewed as a software evaluation method for system comparison [6, 7].

In this paper, software benchmarking is defined as a collaborative and continuous process for improving software products, services, and processes by systematically evaluating and comparing them to those considered to be the best [8].

The reason for benchmarking software products instead of just evaluating them is to obtain several benefits that cannot be obtained from software evaluations. As Fig. 1 illustrates, software evaluation shows the weaknesses of the software or its compliance to quality requirements. If several software products are involved in the evaluation, we also obtain a comparative analysis of these products and recommendations for users. However, when benchmarking several software products, in addition to all the benefits commented, we also gain continuous improvement of the products, recommendations for developers on the practices used when developing these products and, from these practices, those that can be considered best practices.

## 2.2. *The interoperability problem*

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 [9]. 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 technologies have to interchange ontologies and use them.

One of the factors affecting interoperability is heterogeneity. Sheth [11] 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, being the latter related to the ontology alignment problem [12], that is, the problem of how to find relationships between entities in different ontologies.

Semantic Web technologies appear in different forms (ontology development tools, ontology repositories, ontology alignment tools, reasoners, etc.); interoperability is a must for these technologies since they need to be in communication to interchange and use ontologies 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 expressiveness and different reasoning capabilities, as it occurs in knowledge-based systems [13].

Current Semantic Web technologies manage different representation models, 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<sup>b</sup> (UML), the Ontology Definition Metamodel<sup>c</sup> (ODM), or the Open Biomedical Ontologies<sup>d</sup> (OBO) language.

Figure 2 shows the two common ways of interchanging ontologies within Semantic Web tools: directly by storing the ontology in the destination tool, or indirectly

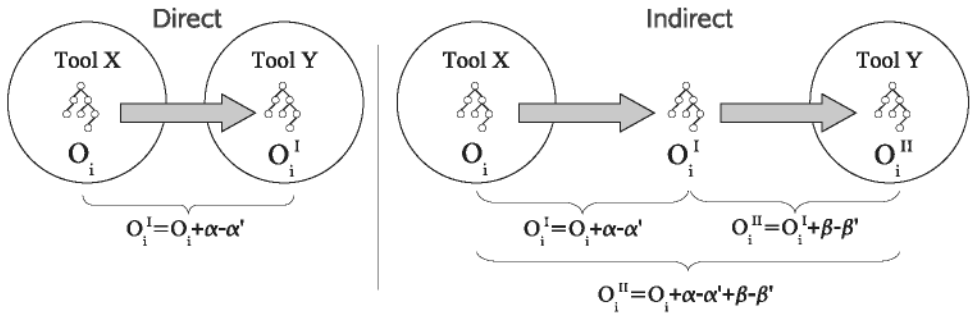


Fig. 2. Ontology interchanges within Semantic Web tools.

by storing the ontology in a shared resource, such as a fileserver, a web server, or an ontology repository.

Ontology interchange should pose no problems when a common representation formalism is used by all the systems involved in the interchange and no differences should exist between the original and the final ontologies (i.e., the  $\alpha$ s and  $\beta$ s in the figure should be null).

However, in the real world, to use a single system is not feasible, as each system provides different functionalities, nor it is feasible 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 technologies 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 with different expressiveness cause information additions or losses in the ontology (the  $\alpha$ s and  $\beta$ s in Fig. 2), once in the case of a direct interchange and twice in the case of an indirect one.

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 [14].

### 2.3. Previous interoperability evaluations

In the Semantic Web area, technology interoperability has been punctually evaluated. Some qualitative analyses have been performed in [15] concerning ontology development tools, ontology merge and integration tools, ontology evaluation tools, ontology-based annotation tools, and ontology storage and querying tools; and in [16] 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 do not provide empirical studies to support their conclusions.

The only exceptions are the experiments 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 tools interoperability using an interchange language [17].

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.

The EON2003 experiments were a first and valuable step toward evaluating interoperability since they highlighted the 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 is still unsolved.
- The workshop experiments concerned only 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 that participated were made.

We learnt many lessons from the results of the initial EON experiments, and they enabled us to do a systematic evaluation on a technical level, as it is the evaluation we present in this paper.

### 3. RDF(S) Interoperability Benchmarking

The RDF(S) Interoperability Benchmarking started in Knowledge Web as an effort to improve the interoperability of Semantic Web technologies and to provide comprehensive recommendations for industry on how to use these tools. The benchmarking was organized and carried out following the software benchmarking methodology developed in Knowledge Web [18] by the authors; this methodology provides the general guidelines that have to be adapted to each case. We now present the main decisions and outcomes of instantiating such a methodology.

The goal of the benchmarking was **to evaluate and improve the interoperability of Semantic Web technologies using RDF(S) as the interchange language**.

As mentioned above, achieving interoperability between Semantic Web technologies is not straightforward when these tools do not share a common knowledge model and, nowadays, their users do not know the effects of interchanging an ontology from one tool to another.

The benefits pursued through our goal are related to the expected outcomes of the benchmarking and involve different communities dealing with Semantic Web

technologies, namely, the research community, the industrial community, and tool developers. Such benefits are

- To create consensual processes and mechanisms for evaluating the interoperability of these tools.
- To produce user and developer recommendations on the interoperability of these tools.
- To acquire a deep understanding of the practices used to develop these tools and of how the practices used affect their interoperability.
- To extract from these practices those that can be considered best practices when developing the tools.

### 3.1. *Metrics and criteria*

As the goal presented in the previous section was too general, we refined the scope to cover a concrete interoperability scenario. From the different ways that Semantic Web technologies have to interoperate, presented in Sec. 2.2, the most commonly used and, therefore, the one considered here, is the indirect interchange of ontologies by storing them in a shared resource. A direct interchange of ontologies would require developing interchange mechanisms for each pair of tools, which would be very costly.

In our case, the shared resource is a local filesystem where ontologies are stored in text files using RDF(S); such ontologies are serialized using the RDF/XML syntax because this is the syntax most widely employed in Semantic Web technologies. A future benchmarking activity inside Knowledge Web will cover the case of using OWL [19] as the interchange language.

In this scenario, interoperability depends on two different tool functionalities: one that reads an ontology stored in the tool and writes it into an RDF(S) file (RDF(S) exporter from now on), and another that reads an RDF(S) file with an ontology and stores this ontology into the tool (RDF(S) importer from now on).

The evaluation metrics must describe thoroughly the interoperability between an origin tool and a destination one. Therefore, to obtain detailed information on tool interoperability using an interchange language, we need to know

- The components of the knowledge model of an origin tool that can be interchanged with a destination tool.<sup>e</sup>
- The secondary effects of interchanging ontologies that include these components, such as insertion or loss of information.
- The subset of the knowledge models of the tools that such tools can use to correctly interoperate.

- The problems that arise when ontologies are interchanged between two tools and the causes of these problems.

Some specific evaluation criteria should be established for each experiment to assess the interoperability of the tools. The experiments to be performed should yield data informing how the tools comply with these criteria.

### 3.2. *Partner search*

Participation in the benchmarking was open to any organisation irrespective of being a Knowledge Web partner or not. To involve other organisations in the process, with the goal of having the best-in-class tools participating, the following actions were taken:

- A benchmarking proposal, that is, a document being used as a reference along the benchmarking, was published as a public web page<sup>f</sup>; this web page includes all the relevant information about the benchmarking: motivation, goals, benefits and costs, tools and people involved, planning, related events, and a complete description of the experimentation and the benchmark suites.
- Research was performed on the existing ontology development tools, both freely available and commercial ones, which could export and import to and from RDF(S); besides, their developers were contacted.
- The interoperability benchmarking was announced with a call for participation through the main mailing lists of the Semantic Web area and through lists specific to ontology development tools.

Any Semantic Web tool capable of importing and exporting RDF(S) could participate in the RDF(S) Interoperability Benchmarking. In this benchmarking activity, not only ontology development tools, but also RDF repositories participated.

Table 1 shows the six tools that took part in the RDF(S) Interoperability Benchmarking, three of which are ontology development tools: KAON,<sup>g</sup> Protégé<sup>h</sup> (using its RDF backend), and WebODE<sup>i</sup>; the other three are RDF repositories: Corese,<sup>j</sup> Jena<sup>k</sup> and Sesame.<sup>l</sup>

As Table 1 shows, benchmarking was not always performed by the tool developers. Furthermore, the participating tools presented a variety of knowledge models. Next, we present an enumeration of the knowledge models of the tools:

- Corese's knowledge model enables to process RDF(S) and OWL Lite within the Conceptual Graphs formalism [20].



Table 1. Tools participating in the RDF(S) interoperability benchmarking.

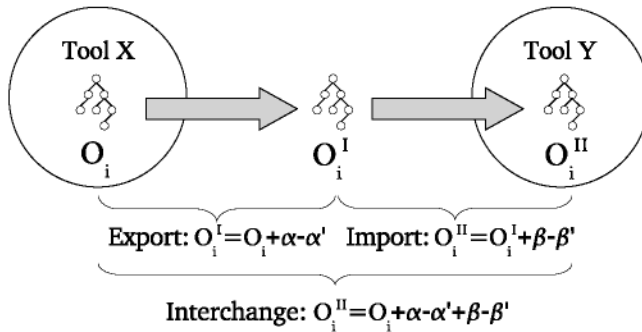
Tool	Version	Developer	Experimenter
Corese	2.1.2	INRIA	INRIA
Jena	2.3	HP	U. P. Madrid
KAON	1.2.9	U. Karlsruhe	U. Karlsruhe
Protégé	3.2 beta build 230	Stanford U.	U. P. Madrid
Sesame	2.0 alpha 3	Aduna	U. P. Madrid
WebODE	2.0 build 109	U. Politécnica de Madrid	U. P. Madrid

- Jena’s knowledge model supports RDF and the following ontology formalisms built on top of RDF: RDF(S), the varieties of OWL, and the now-obsolete DAML+OIL [21].
- KAON’s knowledge model is an extension of RDF(S) that contains the essential modelling primitives of frame-based systems [22].
- Protégé’s knowledge model is based on a flexible metamodel, which is comparable to object-oriented and frame-based systems [23].
- Sesame’s knowledge model allows managing RDF(S) [24].
- WebODE’s knowledge model is based in frames and is extracted from the intermediate representations of METHONTOLOGY [25].

### 3.3. Experiment definition

The interoperability of Semantic Web tools using an interchange language depends on the capabilities of such tools to import and export ontologies from/to this language. Therefore, as Fig. 3 shows, the experiments included not only an interoperability evaluation but also a previous evaluation of the RDF(S) importers and exporters. Thus, the RDF(S) importer and exporter evaluation results are used in helping to interpret the interoperability results.

Regarding the experiments, any group of ontologies can be used as input, but having real, large or complex ontologies is useless if we do not know whether the



tools can interchange simple ontologies correctly. However, because one of the goals of benchmarking is the improvement of the tools, the ontologies must be simple so as to isolate problem causes and to identify possible new problems.

Therefore, to obtain the required experiment data, three benchmark suites were defined for evaluating the import, export and interoperability capabilities of the tools [26], which were common for all the tools.

As the tools participating in the benchmarking had different internal knowledge models, both the experimentation and the analysis of the results were based on a common group of ontology modelling primitives, available in RDF(S) and in these tools. However, since tackling this common group exhaustively would yield a huge number of benchmarks, we only considered the components used most for modelling ontologies in ontology development tools: classes, instances, properties with domain and range, literals, and class and property hierarchies. The rest of the components have not been dealt with so far.

In this setting, it is recommended to perform the import and export experiments before the interoperability ones, because tool interoperability highly depends on the functioning of their importers and exporters and because the effort required to execute the interoperability experiments diminishes when the files produced in the export experiments are used.

The quality of the benchmark suites used is essential for the results of the benchmarking. Therefore, once the benchmark suites were defined, they were published on the benchmarking web page<sup>f</sup> so that they could be reviewed by the participants. The benchmark suites were also validated and refined in reviews performed by Knowledge Web partners in several meetings.

The benchmarking web page contains, besides the detailed description of the benchmark suites, all the files to be used in the experiments, templates for collecting the results, and the experimentation results obtained.

The benchmark suites were intended to be executed manually but, as they contain many benchmarks, it is highly recommended to execute them (or part of them) automatically. In the cases of Corese, Jena, Sesame, and WebODE, most of the experimentation was automated. In the other cases, it was performed manually.

### 3.3.1. *The RDF(S) import benchmark suite*

The RDF(S) Import Benchmark Suite contains benchmarks that define a simple RDF(S) ontology serialized in an RDF/XML file that must be loaded into the tool.

To isolate the factors influencing the correct import of an ontology, we have defined two types of import benchmarks: those that evaluate the import of the different combinations of components of the RDF(S) knowledge model, and those that evaluate the import of the different variants of the RDF/XML syntax, as stated in the RDF/XML specification.

Table 2 shows the 10 groups of the RDF(S) Import Benchmark Suite, which comprises 82 benchmarks. The table contains both the number of benchmarks and the RDF(S) components used in each group.

Table 2. Groups of the RDF(S) import benchmarks.

Group	No.	Components used
Class	2	<i>rdfs:Class</i>
Metaclass	5	<i>rdfs:Class</i> , <i>rdf:type</i>
Subclass	5	<i>rdfs:Class</i> , <i>rdfs:subClassOf</i>
Class and property	6	<i>rdfs:Class</i> , <i>rdf:Property</i> , <i>rdfs:Literal</i>
Property	2	<i>rdf:Property</i>
Subproperty	5	<i>rdf:Property</i> , <i>rdfs:subPropertyOf</i>
Property with domain and range	24	<i>rdfs:Class</i> , <i>rdf:Property</i> , <i>rdfs:Literal</i> , <i>rdfs:domain</i> , <i>rdfs:range</i>
Instance	4	<i>rdfs:Class</i> , <i>rdf:type</i>
Instance and property	14	<i>rdfs:Class</i> , <i>rdf:type</i> , <i>rdf:Property</i> , <i>rdfs:Literal</i>
Syntax and abbreviation	15	<i>rdfs:Class</i> , <i>rdf:type</i> , <i>rdf:Property</i> , <i>rdfs:Literal</i>
Total	82	<i>rdfs:Class</i> , <i>rdf:type</i> , <i>rdfs:subClassOf</i> , <i>rdf:Property</i> , <i>rdfs:domain</i> , <i>rdfs:range</i> , <i>rdfs:subPropertyOf</i> , <i>rdfs:Literal</i>

Table 3. An example of an RDF(S) import benchmark definition.

Identifier	I09
Description	Import one class that is subclass of several classes
Graphical representation	
RDF/XML file	<pre> &lt;rdf:RDF xmlns="http://www.w3.org/2000/01/rdf-schema#"   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"   &lt;Class rdf:about="http://www.nothing.org/graph09#C1"&gt;     &lt;subClassOf rdf:resource="http://www.nothing.org/graph09#C2"/&gt;     &lt;subClassOf rdf:resource="http://www.nothing.org/graph09#C3"/&gt;   &lt;/Class&gt;   &lt;Class rdf:about="http://www.nothing.org/graph09#C2"&gt;   &lt;/Class&gt;   &lt;Class rdf:about="http://www.nothing.org/graph09#C3"&gt;   &lt;/Class&gt; &lt;/rdf:RDF&gt; </pre>

The definition of each benchmark in the benchmark suite, as Table 3 shows, includes the following fields:

- An **identifier** for tracking the different benchmarks.
- A **description** of the benchmark in natural language.
- A **graphical representation** of the ontology to be imported.
- A **file** containing the ontology to be imported in the RDF/XML syntax.

The steps to follow for executing each import benchmark are

- (1) To specify the result expected from importing the file with the RDF(S) ontology into the tool, either by modelling the expected ontology in the tool or by defining the ontology informally (e.g., in natural language).

- (2) To import into the tool the RDF(S) file that contains the RDF(S) ontology defined in the benchmark.
- (3) To compare the imported ontology with the expected ontology specified in the first step and to check whether there is some addition or loss of information.

### 3.3.2. *The RDF(S) export benchmark suite*

The RDF(S) Export Benchmark Suite contains benchmarks that define an ontology that must be modelled in the tool and saved to an RDF(S) file.

We have defined two types of benchmarks for isolating the two factors that influence the correct export of an ontology. One group of benchmarks evaluates the correct export of the combinations of components of the ontology development tool knowledge model, whereas the other group evaluates the export of ontologies with concepts and properties whose names include characters restricted by RDF(S), such as those not allowed for representing RDF(S) or XML URIs.

The composition of the RDF(S) Export Benchmark Suite is similar to the composition of the Import one, but instead of taking as input the knowledge model of RDF(S), it takes the common core of knowledge modelling components of KAON, Protégé, WebODE, and RDF(S) and, therefore, we obtained a different number of benchmarks.

Table 4 shows the 11 groups of the RDF(S) Export Benchmark Suite, which comprises 66 benchmarks. The table contains the number of benchmarks and the components used in each group.

The definition of each benchmark, as Table 5 shows, includes the following fields:

- An **identifier** for tracking the different benchmarks.
- A **description** of the benchmark in natural language.
- A **graphical representation** of the ontology to be exported by the tool.
- The **instantiation** of the ontology in each of the participating tools, using the vocabulary and components of these tools.

The steps to follow for executing each of the export benchmarks are:

- (1) To specify the expected ontology that results from exporting the ontology, either in RDF(S) or by defining it informally (e.g., in natural language).
- (2) To model in the tool the ontology described in the benchmark.
- (3) To export the ontology modelled with the tool to RDF(S).
- (4) To compare the exported RDF(S) ontology with the expected RDF(S) ontology specified in the first step, examining whether there is some addition or loss of information.

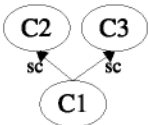
### 3.3.3. *The RDF(S) interoperability benchmark suite*

The RDF(S) Interoperability Benchmark Suite evaluates the interchange of ontologies from one origin tool to a destination one and vice versa. Each benchmark in

Table 4. Groups of the RDF(S) export benchmarks.

Group	No.	Components used
Class	2	<i>class</i>
Metaclass	5	<i>class, instanceOf</i>
Subclass	5	<i>class, subClassOf</i>
Class and object property	4	<i>class, object property</i>
Class and datatype property	2	<i>class, datatype property, literal</i>
Object property	14	<i>object property</i>
Datatype property	12	<i>datatype property</i>
Instance	4	<i>class, instanceOf</i>
Instance and object property	9	<i>class, instanceOf, object property</i>
Instance and datatype property	5	<i>class, instanceOf, datatype property, literal</i>
URI character restrictions	4	<i>class, instanceOf, object property, datatype property, literal</i>
Total	66	<i>class, instanceOf, subClassOf, object property, datatype property, literal</i>

Table 5. An example of an RDF(S) export benchmark definition.

Identifier	E09
Description	Export one class that is subclass of several classes
Graphical representation	
WebODE's instantiation	Export one concept that is subclass of several concepts
Protégé's instantiation	Export one class that is subclass of several classes
...	...

the benchmark suite defines an ontology that must be modelled in the origin tool, saved to an RDF(S) file, and loaded into the destination tool.

Since the factors influencing the correct interchange of an ontology (besides the correct functioning of the importers and exporters) as well as the knowledge model used for defining the ontologies are the same as those in the RDF(S) Export Benchmark Suite, the ontologies defined in the RDF(S) Interoperability Benchmark Suite are identical to those of the RDF(S) Export Benchmark Suite.

The steps to follow for executing each of the interoperability benchmarks are:

- (1) To specify the expected ontology resulting from interchanging the ontology in the destination tool, either by modelling the expected ontology in the destination tool or by defining it informally (e.g., in natural language).

- (2) To model the ontology described in the benchmark in the source tool.
- (3) To export the ontology modelled with the source tool to an RDF(S) file.
- (4) To import the RDF(S) file (exported by the source tool) into the destination tool.
- (5) To compare the interchanged ontology with the expected one, which is specified in the first step, checking whether there is some addition or loss of information.

If the tools have already executed the RDF(S) Export Benchmark Suite, then steps (2) and (3) can be ignored, as the RDF(S) exported files of all the tools will be available from the export experiments; therefore, participants will only have to import the exported files into their tools.

### 3.3.4. *Evaluation criteria*

The evaluation criteria are common for the three benchmark suites and are defined as follows:

- **Modelling** (*YES/NO*). The tool can model the ontology components described in the benchmark.
- **Execution** (*OK/FAIL*). The execution of the benchmark is carried out without any problem, and the tool always produces its expected result. But when an execution fails, the following information is required:
  - Reasons for the benchmark execution failure.
  - If the tool was fixed to pass a benchmark, which corrections the tool required.
- **Information added or lost**. The information added or lost during the ontology interchange.

In the export and interoperability benchmark suites, if a benchmark defines an ontology that cannot be modelled in a certain tool, such a benchmark cannot be executed in the tool, being the *Execution* result *N.E.* (Non Executed).

Since Semantic Web tools have different knowledge models, there is no *Right* or *Wrong* result. On the other hand, any combination of the *Modelling* and *Execution* results can be possible since results depend on the decisions taken by the tool developers:

- *It models and executes* (*Modelling=YES* and *Execution=OK*).
- *It does not model and executes* (*Modelling=NO* and *Execution=OK*).
- *It models and fails* (*Modelling=YES* and *Execution=FAIL*).
- *It does not model and fails* (*Modelling=NO* and *Execution=FAIL*).
- *It does not model* (*Modelling=NO* and *Execution=Non Executed*).

It is clear that different tools have different strategies for dealing with the components not allowed in their knowledge models. For example, metaclasses can be modelled in RDF(S), but a tool that cannot represent metaclasses has two alternatives when importing an RDF(S) metaclass: either to import it as a class, or not

Table 6. Fictitious results of executing the benchmark I46.

Tool	Modelling	Execution	Information added	Information lost
A	YES	OK	A label in all the components	—
B	YES	FAIL	—	The property’s range
C	NO	OK	The range <i>String</i>	The range <i>xsd:string</i>
D	NO	OK	The range <i>rdfs:Literal</i>	The range <i>xsd:string</i>
E	NO	FAIL	—	The property

to do it. However, even if a tool cannot model some components of the ontology, it should be able to import the other components correctly.

Table 6 shows an example of executing the benchmark I46 (*Import just one property that has a class as domain and the XML Schema datatype “string” as range, with the class defined in the ontology*) in five fictitious tools identified as A, B, C, D, and E.

In the example, tools A and B can model the XML Schema datatype *string* as range and, therefore, their *Modelling* result is *YES*; on the other hand, tools C, D and E cannot model such a datatype and, therefore, their *Modelling* result is *NO*.

The result expected from tools A and B is a property whose domain is a class and whose range is the XML Schema datatype *string*. Tool A imports all these components and adds a label with the name of the component to all the components; therefore, the *Execution* result of tool A is *OK*; besides, A inserts new information into the ontology. Tool B imports the property, but it does not import the range. Since tool B does not produce the expected result, its *Execution* result is *FAIL* and B loses information when it imports the ontology.

Because tools C, D and E cannot model the XML Schema datatype *string* as range, even though they can model string ranges, the expected result of these tools is to have a property whose domain is a class and whose range is *string*. Tools C and D produce the expected result and their *Execution* result is *OK*; both tools lose information about the range being the XML Schema datatype *string*, though tool C creates the range as its own datatype *String* and tool D creates the range as *rdfs:Literal*; therefore, these two tools, C and D, insert new information in the ontology. Finally, tool E does not import the property at all, although its expected result is the import of the property with a string range. The *Execution* result of tool E is *FAIL*, while E loses all the information about the property when it imports the ontology.

#### 4. RDF(S) Import and Export Results

Tables 7 and 8<sup>m</sup> present a quantitative analysis of the global results of the last execution of the RDF(S) Import and Export Benchmark Suites, carried out in

<sup>m</sup>The tool names have been abbreviated in the tables: K = KAON, P = Protégé, W = WebODE, C = Corese, J = Jena, S = Sesame.

Table 7. Final RDF(S) import results.

	K	P	W	C	J	S
Models and executes	79	48	47	82	82	82
Does not model and executes		24	25			
Models and fails	1	2	3			
Does not model and fails	2	8	7			

Table 8. Final RDF(S) export results.

	K	P	W	C	J	S
Models and executes	54	40	25	62	62	62
Models and fails	3	8				
Does not model	9	18	41	4	4	4

January 2006. The tables show the number of benchmarks that fall into each possible combination of the *Modelling* and *Execution* results for each tool.

The results obtained when importing from and exporting to RDF(S) depend mainly on the knowledge model of the tool that executed the benchmark suite.

The tools that natively support the RDF(S) knowledge model (Corese, Jena and Sesame, i.e., the RDF repositories) do not need to perform any translation in the ontologies when importing/exporting them from/to RDF(S). The RDF repositories import and export correctly from/to RDF(S) all the combinations of components, as these operations do not require any translation.

In the case of tools with non-RDF knowledge models (KAON, Protégé and WebODE, the ontology development tools), some of their knowledge model components can also be represented in RDF(S) whereas some others cannot; on the other hand, tools do need to translate ontologies between their knowledge models and RDF(S). Besides, not all the combinations of components of the RDF(S) knowledge model that have been considered in the benchmarking can be modelled into all the tools.

Next, we present an analysis of the import and export results of the participating ontology development tools. A detailed analysis of the RDF(S) import and export results can be found in [27].

#### 4.1. *Import results*

In general, ontology development tools import correctly from RDF(S) all or almost all the combinations of components that they model, seldom adding or losing information. The only exceptions are

- Protégé, which poses problems only when importing classes or instances that are instances of multiple classes.
- WebODE, which poses problems only when importing properties with a XML Schema datatype as range.



When the ontology development tools import ontologies with combinations of components that they cannot model, they lose the information on these components. Nevertheless, they usually try to represent such components by partially using other components from their knowledge models. In most cases, the import is performed correctly. The only exceptions are:

- KAON, which poses problems when it imports class hierarchies with cycles.
- Protégé, which poses problems when it imports class and property hierarchies with cycles and properties with multiple domains.
- WebODE, which poses problems when it imports properties with multiple domains or ranges.

When dealing with the different variants of RDF/XML syntax, ontology development tools

- Import correctly resources with the different URI reference syntaxes.
- Import correctly resources with the different syntaxes (shortened and unshortened) of empty nodes, of multiple properties, of typed nodes, of string literals, and of blank nodes. The only exceptions are: KAON when it imports resources with multiple properties in the unshortened syntax; and Protégé when it imports resources with empty and blank nodes in the unshortened syntax.
- Do not import language identification attributes (*xml:lang*) in tags.

#### 4.2. *Export results*

In general terms, ontology development tools export correctly to RDF(S) all or nearly all of the combinations of components that they model without losing information; whereas in specific terms:

- KAON poses problems only when exporting to RDF(S) datatype properties without range and datatype properties with multiple domains and a XML Schema datatype as range.
- Protégé presents problems only when exporting to RDF(S) classes or instances that are instances of multiple classes and template slots with multiple domains.

Furthermore, the lower the number of not-modelled benchmarks is, the more similar is the knowledge model of the tool to the common knowledge model considered, which contains the subset of common components in KAON, Protégé, WebODE, and RDF(S). In our case, the knowledge models of the RDF repositories are the most similar, followed by those of KAON, Protégé, and WebODE (in this order).

The number of benchmarks not modelled that appears in the RDF repository results corresponds with the benchmarks that check the component naming restrictions. As these restrictions cannot be modelled in RDF(S), such benchmarks cannot be executed in the RDF repositories.

When we disregard these benchmarks, we can observe that the common knowledge model is totally compatible with the RDF(S) knowledge model and partially compatible with KAON, Protégé and WebODE (listed in decreasing compatibility order).

When these tools export components that are present in their knowledge model but cannot be represented in RDF(S), as is the case of their own datatypes, they usually insert new information in the ontology even though some information is lost.

When dealing with concepts and properties whose names do not fulfil URI character restrictions, each ontology development tool behaves differently:

- When names do not start with a letter or “\_”, some tools leave the name unchanged, while others replace the first character with “\_”.
- Spaces in names are replaced by “-” or “\_”, depending on the tool.
- URI reserved characters and XML delimiter characters are left unchanged, replaced by “\_”, or encoded, depending on the tool.

## 5. Interoperability Results

The RDF repositories (Corese, Jena and Sesame) interoperate correctly among themselves as they always import and export from/to RDF(S) correctly. This fact causes that interoperability between the ontology development tools and the RDF repositories depends only on the capabilities of the former to import and export from/to RDF(S) and, therefore, the results about this interoperability are identical to those presented in the previous section.

Table 9 shows a quantitative analysis of the global results of executing the RDF(S) Interoperability Benchmark Suite. The table shows the results obtained when one tool is the origin and another tool is the destination of the interchange. The table shows the number of benchmarks that fall into each of the possible combinations of the *Modelling* and *Execution* results for each tool.

The import and export results presented in the previous section showed that few problems occur when importing and exporting ontologies. Nevertheless, interoperability results present more problems.

As a general comment, interoperability between the tools depends on:

Table 9. Global RDF(S) interoperability results.

From	K	P	W	K	P	W	K	P	W
To	KAON			Protégé			WebODE		
Models and executes	56	35	24	29	34	23	36	35	25
Does not model and executes				3	2	2	7	3	
Models and fails		13	1	14	7				
Does not model and fails				10	5		13	10	
Not executed	10	18	41	10	18	41	10	18	41

- (a) The correct working of their RDF(S) importers and exporters.
- (b) The mode chosen for serializing the exported ontologies in the RDF/XML syntax.

Furthermore, we have observed that some problems in any of these factors affect the results of not just one but of several benchmarks. This means that, in some cases, fixing a single import or export problem or changing the mode of serializing ontologies can cause significant interoperability improvements.

Next, the components that can be interchanged between the tools are listed in Table 10. The table illustrates the different combinations of components classified into categories; each column shows whether the combination of components in that category can be interchanged between a group of tools.

In the table abovementioned, “Y” means that all the benchmarks in the category have an *Execution* value of *OK*, “N” means that at least one of the benchmarks in the category has an *Execution* value of *FAIL*, and the “-” character means that the component cannot be modelled in some of the tools and, therefore, cannot be interchanged between them.

It must be noted that a benchmark can be part of several categories. For example, benchmark *In35* (Interchange just one object property that has as domain several classes, with the classes defined in the ontology) belongs to the “Object properties without domain or range” and to the “Object properties with multiple domains or ranges” categories.

Table 10. Components interchanged between the tools.

Combination of components	K-K	P-P	W-W	K-P	K-W	P-W	K-P-W
Classes	Y	Y	Y	Y	Y	Y	Y
...instance of a single metaclass	Y	Y	—	N	—	—	—
...instance of a multiple metaclasses	Y	N	—	N	—	—	—
Class hierarchies without cycles	Y	Y	Y	Y	Y	Y	Y
Datatype properties without domain or range	Y	Y	—	N	—	—	—
...with multiple domains	Y	—	—	—	—	—	—
...whose range is String	Y	Y	Y	N	N	Y	N
...whose range is a XML Schema datatype	Y	—	Y	—	Y	—	—
Object properties without domain or range	Y	Y	—	Y	—	—	—
...with multiple domains or ranges	Y	—	—	—	—	—	—
...with a domain and range	Y	Y	Y	Y	Y	Y	Y
Instances of a single class	Y	Y	Y	Y	Y	Y	Y
...of multiple classes	Y	N	—	N	—	—	—
...related via object properties	Y	Y	Y	Y	Y	Y	Y
...related via datatype properties	Y	Y	Y	N	Y	Y	N
...related via datatype properties whose range is a XML Schema datatype	—	—	Y	—	—	—	—

### 5.1. *Interoperability using the same tool*

Ontology development tools seem not to encounter problems when the source and the destination of an ontology interchange are the same tool. The only exception is Protégé when it interchanges classes that are instances of multiple metaclasses and instances of multiple classes, because Protégé does not import resources that are instances of multiple metaclasses.

### 5.2. *Interoperability between each pair of tools*

Interoperability between different tools varies depending on the tools. Furthermore, in some cases the tools are able to interchange certain components from one tool to another, but not the other way round.

When **KAON** interoperates with **Protégé**, both tools can correctly interchange some of the common components that they are able to model. But problems occur with classes that are an instance of a single metaclass or of multiple metaclasses, with datatype properties without domain or range, with datatype properties whose range is *String*, with instances of multiple classes, and with instances related through datatype properties.

When **KAON** interoperates with **WebODE**, they can correctly interchange almost all the common components that both tools can model. The only exception occurs when they interchange datatype properties with domain and whose range is *String*.

When **Protégé** interoperates with **WebODE**, they can correctly interchange all the common components that both tools can model.

### 5.3. *Interoperability between all the tools*

Interoperability between **KAON**, **Protégé** and **WebODE** can be achieved through nearly all the common components that all these tools can model: classes, class hierarchies without cycles, object properties with a domain and a range, instances of a single class, and instances related through object properties. The common components that these tools cannot use are (1) datatype properties with domain and whose range is *String* and (2) instances related through datatype properties.

### 5.4. *Interoperability regarding URI character restrictions*

Interoperability is low when tools interchange ontologies containing URI character restrictions in class and property names. This is mainly due to the fact that tools usually encode some or all the characters that do not comply with these restrictions, which provokes changes in class and property names.

## 6. Recommendations

From the benchmarking results, we have compiled a comprehensive set of practices that may serve as recommendations for Semantic Web tool developers, for ontology engineers, and for anybody interested in carrying out a benchmarking activity.

### 6.1. Recommendations for semantic web tool developers

This section includes general recommendations for improving the interoperability of Semantic Web tools when developing them. In [27], we provide more detailed recommendations to improve each of the participating tools.

Interoperability between Semantic Web tools using an interchange language depends on how the importers and exporters of these tools work. In their turn, how these importers and exporters work depends on the development decisions made by tool developers, who are different people with different needs. Therefore, it is not easy to provide general recommendations for developers since many issues are involved. However, some recommendations for Semantic Web tool developers can be extracted from the analysis of the benchmarking results.

The first requirement for achieving interoperability is that the importers and exporters of the tools be robust and work correctly when dealing with unexpected inputs. Although this is an evident recommendation, the results show that this requirement is not always fulfilled by the tools and that some tools even crash when they import some combinations of components.

Above all, tools should correctly work with the combinations of components that are present in the interchange language but that cannot be modelled in them. For example, cycles in class and property hierarchies cannot be modelled in ontology development tools; these tools, however, should be able to import these hierarchies by eliminating the cycles.

When exporting components commonly used by Semantic Web tools, they should be completely defined in the file; for example, in RDF(S), metaclasses and classes in class hierarchies should be defined as instances of *rdfs:Class*, properties should be defined as instances of *rdf:Property*, etc.

Exporting complete definitions of components seldom used by the tools can cause problems if such components are later imported by other tools; for example, not every tool deals with datatypes defined as instances of *rdfs:Datatype* in the file or with *rdf:datatype* attributes in properties.

Every exported resource should have a namespace if the document does not define a default namespace.

In a few cases, a development decision will improve interoperability with some tools but produce loss with others; for example, when exporting to RDF(S) classes that are instances of a metaclass, some tools require that the class be defined as instance of *rdfs:Class* while some other tools require the opposite, being the two options correct.

The collateral consequences of the development decisions should be analysed by the tool developers. For example, if a datatype is imported as a class in the ontology, then the literal values of this datatype should be imported as instances in the ontology, which would complicate the management of these values.

Semantic Web tool developers should be aware of the semantic equivalences and differences between the knowledge models of their tool and the interchange language; additionally, tools should notify the user when the semantics is changed.

## **6.2. *Recommendations for ontology engineers***

This section offers recommendations for ontology engineers who expect to use their ontologies in more than one tool. Depending on the tools used, the level of interoperability may be greater or lower, as can be seen in Sec. 5.

Ontology engineers should be aware of the components that can be represented in the knowledge models of the tools and in the interchange languages. Hence, they should try to use the common components of these tools in their ontologies in order to avoid the already-known knowledge losses.

Ontology engineers should also be aware of the equivalences and differences between the knowledge models of the tools and the knowledge model of the interchange language. For example, in Protégé multiple domains in template slots are considered the union of all the domains, while in RDF(S) multiple domains in properties are considered the intersection of all the domains; in WebODE instance attributes are local to a single concept, while in RDF(S) properties are global and can be used in any class.

It is not recommended to name resources by including in their names spaces or any character that is restricted in the RDF(S), OWL, URI or XML specifications.

With respect to interoperability in the RDF repositories, although these repositories export and import correctly to RDF(S), ontology engineers should consider the limitations that other tools have when exporting their ontologies to RDF(S) with the aim of interchanging them.

## **6.3. *Recommendations for benchmarking***

This section offers recommendations to perform benchmarking activities; such recommendations were extracted from the lessons learnt while instantiating the methodology.

First of all, the participation of relevant experts of the community during the whole benchmarking process is crucial, and the inclusion of the best-in-class tools is a must, even in those cases in which the organizations that develop these tools do not participate in the process.

Benchmarking takes a long time as it requires tasks that are not immediate, i.e., announcements, agreements, etc. Therefore, its planning should consider a realistic duration of the benchmarking and should provide the necessary resources.

The effort to be devoted to benchmarking is a main criterion for any organization (especially companies) when it has to decide whether to participate in benchmarking. Resources are needed mainly in four tasks: benchmarking organization, definition of the experiment, execution of the experiments, and analysis of

the results. Therefore, the tasks to be performed in benchmarking, particularly the experiment-related tasks, should be automated as much as possible.

Benchmarking is not about comparing the results of the tools but the practices leading to these results. Therefore, experiments should be designed to obtain these practices as well as the results. In our case, developer practices were obtained both by providing specific questions to the experimenters, so that they could identify the practices used to develop the tools, and by allowing the experimenters to comment on the tools behaviour.

## 7. Conclusions and Future Work

Seamless interoperability among Semantic Web technologies greatly facilitates the development and deployment of ontologies. In this paper we present a set of concrete benchmark suites for evaluating RDF(S) import, export and interoperability as well as the results and best practices obtained after employing such benchmark suites in a number of well-known Semantic Web tools.

As the three benchmark suites are publicly available on the Web, they can be used both by tool developers to evaluate and improve the interoperability of their tools, provided that these tools have import and export functionalities, and by ontology engineers to select the appropriate tool for their ontology development activities.

It must be noted that the benchmark suites presented here have been defined with the goal of evaluating interoperability. Therefore, even if these benchmark suites can be used to evaluate tool importers and exporters, they are not exhaustive for these tasks and should be extended. An exhaustive evaluation of the RDF(S) import capabilities of a tool should take into account the whole RDF(S) model, whereas an exhaustive evaluation of the export capabilities of a tool should take into account the whole knowledge model of the tool.

The benchmarking results are publicly available. Nevertheless, it must be noted that these results are valid for the specific versions of the tools in which the experiments were performed and, because the development of these tools continues, the results are expected to change. This highlights the need of a continuous evaluation of the technology.

From the RDF(S) interoperability results, we have observed that only Corese, Jena, KAON, Sesame, and WebODE can interoperate with themselves using RDF(S) as the interchange language and that the only clusters of RDF(S)-interoperable tools are Corese with Jena and with Sesame, and Protégé with WebODE; using in all the cases the common knowledge model components that both tools can model.

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 observed in the results; 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 models differ, interoperability can be only achieved by means of lightweight ontologies. For example, when Protégé interoperates with WebODE using RDF(S) as the interchange language, both tools can only interchange ontologies that include a limited set of components, and they are not able to use a richer expressiveness as their knowledge models allow.

We should add that some of the participating tools have been improved even before the *Improvement* phase of the methodology. Because the 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 have been informed of the results of these activities and of the recommendations proposed for improving their tools.

During this benchmarking activity, tool developers sometimes automated the execution of the benchmark suites, but the experimentation was mainly done by hand. Carrying out experiments manually and analysing the results involves spending significant resources. Besides, the results manually obtained depend on the expertise of the people performing the experiments and can be influenced by human errors.

Therefore, experiments in benchmarking should be automated as much as possible. This automatization can minimise human errors and, whenever human intervention is needed, mechanisms should be set up to detect this kind of errors.

With regard to participation, we should explain that some research institutions and companies chose not to participate because they could not afford the expenses. Thus, their absence affected the number of participants in the benchmarking, which was fairly low (3 organizations and 6 tools). We think that it would be desirable for the future to continue these benchmarking activities with a higher number of tools, and this involves to develop means of automatizing experimentation as much as possible.

To increase the usability of the benchmarking results, it would be important to facilitate effective ways of analysing and exploiting the results by means of a web application, so that users could perform complex analyses of these results. This requires a previous translation of the results from the spreadsheets where they were collected to a machine-processable format. The IRIBA<sup>n</sup> application allows analysing the RDF(S) interoperability results of the tools in different moments.

Future work includes the development of appropriate OWL benchmark suites for benchmarking the interoperability of Semantic Web technologies using OWL as the interchange language.



## Acknowledgements

This work is partially supported by a FPI grant from the Spanish Ministry of Education (BES-2005-8024), by the IST project Knowledge Web (IST-2004-507482) 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 RDF(S) Interoperability Benchmarking: Olivier Corby, Jesús Prieto-González, York Sure, Moritz Weiten, and Markus Zondler. Thanks to Rosario Plaza for reviewing the grammar of this paper.

## References

1. D. Brickley and R. V. Guha (eds.), RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, 10 February 2004 (2004).
2. ISO/IEC, ISO/IEC 14598-1: Software product evaluation — Part 1: General overview, 1999.
3. R. Camp, Benchmarking: *The Search for Industry Best Practices that Lead to Superior Performance* (ASQC Quality Press, Milwaukee, 1989).
4. M. Spendolini, *The Benchmarking Book* (AMACOM, New York, NY, 1992).
5. C. Wohlin, A. Aurum, H. Petersson, F. Shull and M. Ciolkowski, Software inspection benchmarking — a qualitative and quantitative comparative opportunity, in *Proceedings of 8th International Software Metrics Symposium*, 2002, pp. 118–130.
6. B. Kitchenham, DESMET: A method for evaluating software engineering methods and tools, Technical Report TR96-09, Department of Computer Science, University of Keele, Staffordshire, UK, 1996.
7. A. Weiss, Dhrystone benchmark: History, analysis, scores and recommendations, White paper, EEMBC Certification Laboratories, LLC, 2002.
8. R. García-Castro, Keynote: Towards the improvement of the Semantic Web technology, in *Proceedings of the Second International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2006)*, Athens, GA, USA (2006).
9. IEEE-STD-610: ANSI/IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, IEEE (1991).
10. E. Duval, Learning technology standardization: Making sense of it all, *Int. J. Computer Science and Information Systems* **1** (2004) 33–43.
11. A. Sheth, Changing focus on interoperability in information systems: From system, syntax, structure to semantics, in *Interoperating Geographic Information Systems* (Kluwer, 1998), pp. 5–30.
12. J. Euzenat, T. L. Bach, J. Barrasa, P. Bouquet, J. D. Bo, R. Dieng-Kuntz, M. Ehrig, M. Hauswirth, M. Jarrar, R. Lara, D. Maynard, A. Napoli, G. Stamou, H. Stuckenschmidt, P. Shvaiko, S. Tessaris, S. V. Acker and I. Zaihrayeu, D2.2.3 State of the art on ontology alignment, Technical report, Knowledge Web (2004).
13. R. Brachmann and H. Levesque, A fundamental tradeoff in knowledge representation and reasoning, in *Readings in Knowledge Representation* (Morgan Kaufmann, San Mateo, 1985), pp. 31–40.
14. T. Gruber, A translation approach to portable ontology specifications, *Knowledge Acquisition* **5** (1993) 199–220.
15. OntoWeb, OntoWeb Deliverable 1.3: A survey on ontology tools, Technical report, OntoWeb Thematic Network, 2002.

16. D. Maynard, S. Dasiopolou, S. Costache, K. Eckert, H. Stuckenschmidt, M. Dzbor and S. Handschuh, D1.2.2.1.3 Benchmarking of annotation tools, Technical report, KnowledgeWeb (2007).
17. Y. Sure, O. Corcho (eds.), *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003)*, Vol. 87 of CEUR-WS, Florida, USA.
18. R. García-Castro and A. Gómez-Pérez, Guidelines for benchmarking the performance of ontology management APIs, in *Proceedings of the 4th International Semantic Web Conference (ISWC2005)*, LNCS 3729, Galway, Ireland (Springer-Verlag, 2005), pp. 277–292.
19. D. McGuinness and F. van Harmelen, OWL Web Ontology Language Overview. W3C Recommendation 10 February 2004, Technical report (2004).
20. O. Corby and C. Faron-Zucker, Corese: A Corporate Semantic Web Engine, in *Proceedings of the International Workshop on Real World RDF and Semantic Web Applications*, 11th International World Wide Web Conference, Hawaii, USA, 2002.
21. B. McBride, Jena: Implementing the RDF Model and Syntax Specification, in *Proceedings of the Second International Workshop on the Semantic Web (SemWeb2001)*, 2001.
22. B. Motik, A. Maedche and R. Volz, A conceptual modeling approach for semantics-driven enterprise applications, in *Proceedings of the 1st International Conference on Ontologies, Databases and Application of Semantics (ODBASE2002)*, 2002.
23. N. Noy, R. Ferguson and 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.
24. J. Broekstra, A. Kampman and F. van Harmelen, Sesame: A generic architecture for storing and querying RDF and RDF schema, in *Proceedings of the 1st International Semantic Web Conference (ISWC2002)*, Vol. 2342 (Springer, 2002), pp. 54–68.
25. J. Arpíez, O. Corcho, M. Fernández-López and A. Gómez-Pérez, WebODE in a nutshell, *AI Magazine* **24** (2003) 37–47.
26. R. García-Castro and A. Gómez-Pérez, Benchmark suites for improving the RDF(S) importers and exporters of ontology development tools, in *Proceedings of the 3rd European Semantic Web Conference (ESWC2006)*, LNCS 4011, Budva, Montenegro (Springer-Verlag, 2006).
27. R. García-Castro, Y. Sure, M. Zondler, O. Corby, J. Prieto-González, E. P. Bontas, L. Nixon and M. Mochol, D1.2.2.1.1 Benchmarking the interoperability of ontology development tools using RDF(S) as interchange language, Technical report, Knowledge Web (2006).