D2.1.5 Prototypes of tools and benchmark suites for benchmarking ontology building tools

Raúl García-Castro (Universidad Politécnica de Madrid)

Abstract.
EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB
Deliverable D2.1.5 (WP2.1)

This deliverable describes the benchmark suites that are being used in the benchmarking of the interoperability of ontology development tools using RDF(S) as interchange language. It also describes a tool developed for automating the execution of the experiments in the WebODE platform.
Keyword list: benchmarking, benchmark suite, interoperability, RDF(S)

Document Identifier	KWEB/2004/D2.1.5/v1.2
Project	KWEB EU-IST-2004-507482
Version	v1.2
Date	26. January, 2006
State	final version
Distribution	public

Knowledge Web Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2004-507482.

University of Innsbruck (UIBK) - Coordinator
Institute of Computer Science
Technikerstrasse 13
A-6020 Innsbruck
Austria
Contact person: Dieter Fensel
E-mail address: dieter.fensel@uibk.ac.at

France Telecom (FT)
4 Rue du Clos Courtel
35512 Cesson Sévigné
France. PO Box 91226
Contact person : Alain Leger
E-mail address: alain.leger@rd.francetelecom.com

Free University of Bozen-Bolzano (FUB)
Piazza Domenicani 3
39100 Bolzano
Italy
Contact person: Enrico Franconi
E-mail address: franconi@inf.unibz.it

Centre for Research and Technology Hellas / Informatics and Telematics Institute (ITI-CERTH)
1st km Thermi - Panorama road
57001 Thermi-Thessaloniki
Greece. Po Box 361
Contact person: Michael G. Strintzis
E-mail address: strintzi@iti.gr

National University of Ireland Galway (NUIG)
National University of Ireland
Science and Technology Building
University Road
Galway
Ireland
Contact person: Christoph Bussler
E-mail address: chris.bussler@deri.ie

École Polytechnique Fédérale de Lausanne (EPFL)
Computer Science Department
Swiss Federal Institute of Technology
IN (Ecublens), CH-1015 Lausanne
Switzerland
Contact person: Boi Faltings
E-mail address: boi.faltings@epfl.ch

Freie Universität Berlin (FU Berlin)
Takustrasse 9
14195 Berlin
Germany
Contact person: Robert Tolksdorf
E-mail address: tolk@inf.fu-berlin.de

Institut National de Recherche en Informatique et en Automatique (INRIA)
ZIRST - 655 avenue de l'Europe -
Montbonnot Saint Martin
38334 Saint-Ismier
France
Contact person: Jérôme Euzenat
E-mail address: Jerome.Euzenat@inrialpes.fr

Learning Lab Lower Saxony (L3S)
Expo Plaza 1
30539 Hannover
Germany
Contact person: Wolfgang Nejdl
E-mail address: nejdl@learninglab.de

The Open University (OU)
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA
United Kingdom
Contact person: Enrico Motta
E-mail address: e.motta@open.ac.uk

Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo sn
28660 Boadilla del Monte
Spain
Contact person: Asunción Gómez Pérez
E-mail address: asun@fi.upm.es

University of Karlsruhe (UKARL)
Institut für Angewandte Informatik und Formale Beschreibungsverfahren - AIFB
Universität Karlsruhe
D-76128 Karlsruhe
Germany
Contact person: Rudi Studer
E-mail address: studer@aifb.uni-karlsruhe.de

University of Liverpool (UniLiv)
Chadwick Building, Peach Street
L697ZF Liverpool
United Kingdom
Contact person: Michael Wooldridge
E-mail address: M.J.Wooldridge@csc.liv.ac.uk

University of Manchester (UoM)
Room 2.32. Kilburn Building, Department of Computer Science, University of Manchester, Oxford Road
Manchester, M13 9PL
United Kingdom
Contact person: Carole Goble
E-mail address: carole@cs.man.ac.uk

University of Sheffield (USFD)
Regent Court, 211 Portobello street
S14DP Sheffield
United Kingdom
Contact person: Hamish Cunningham
E-mail address: hamish@dcs.shef.ac.uk

University of Trento (UniTn)
Via Sommarive 14
38050 Trento
Italy
Contact person: Fausto Giunchiglia
E-mail address: fausto@dit.unitn.it

Vrije Universiteit Amsterdam (VUA)
De Boelelaan 1081a
1081HV. Amsterdam
The Netherlands
Contact person: Frank van Harmelen
E-mail address: Frank.van.Harmelen@cs.vu.nl

Vrije Universiteit Brussel (VUB)
Pleinlaan 2, Building G10
1050 Brussels
Belgium
Contact person: Robert Meersman
E-mail address: robert.meersman@vub.ac.be

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to writing parts of this document:

Centre for Research and Technology Hellas
École Polytechnique Fédérale de Lausanne
Free University of Bozen-Bolzano
Institut National de Recherche en Informatique et en Automatique
Learning Lab Lower Saxony
National University of Ireland Galway
Universidad Politécnica de Madrid
University of Karlsruhe
University of Manchester
University of Sheffield
University of Trento
Vrije Universiteit Amsterdam
Vrije Universiteit Brussel

Changes

Version	Date	Author	Changes
0.1	04.12.05	Raúl García-Castro	created
0.2	10.12.05	Raúl García-Castro	inserted the definition of the benchmark suites
0.3	15.12.05	Raúl García-Castro	first complete draft of the document
0.4	20.12.05	Raúl García-Castro	first version of the document
1.0	23.12.05	Raúl García-Castro	included the comments from Rosario Plaza
1.1	10.01.06	Raúl García-Castro	included the comments from the Quality Assessor (Holger Wache)
1.2	26.01.06	Raúl García-Castro	included the comments from the Quality Controller (Diana Maynard)

Executive Summary

In 2005, an activity for benchmarking the interoperability of ontology development tools was started in Knowledge Web; its goal was to know the current interoperability between these tools and to improve it.

This deliverable presents the benchmark suites to be used in the experimentation that is being performed for benchmarking the interoperability of ontology development tools using RDF(S) as interchange language. The benchmark suites are the following:

- **RDF(S) Import Benchmark Suite.** It can be used to evaluate the RDF(S) import functionalities of ontology development tools.
- **RDF(S) Export Benchmark Suite.** It can be used to evaluate the RDF(S) export functionalities of ontology development tools.
- **RDF(S) Interoperability Benchmark Suite.** It can be used to evaluate the interoperability between two ontology development tools using RDF(S) as interchange language.

The deliverable also presents *rdfsbs*, a Java application that automates part of the execution of the benchmarking experiments for the WebODE ontology engineering workbench.

Contents

Chapter 1

Introduction

In Knowledge Web, different benchmarking activities are being (and will be) performed to improve the quality of ontology tools and to learn from the best practices performed when developing this tools. This allows achieving a great improvement in the quality of ontology tools and obtaining recommendations not just for the tool developers but also for the entire community. One of these benchmarking activities is that of the interoperability of ontology development tools using RDF(S) as interchange language[1], that has started in 2005, and to which the content of this deliverable is related.

In the Knowledge Web deliverable 2.1.1 [Wache *et al.*, 2004], an overview of benchmarking and its main related areas was presented: software experimentation and measurement. Deliverable 2.1.4 [García-Castro *et al.*, 2004] proposed the methodology for benchmarking ontology development tools being used in the interoperability benchmarking.

This deliverable presents, in Chapter 2, the definitions of the benchmark suites being used in the *Experimentation* phase of the interoperability benchmarking and guidelines on how to use these benchmark suites. The first definition of the benchmark suites underwent a consensus process. Therefore, the benchmark suites presented in this deliverable are the updated and improved versions of the first benchmark suites, according to the changes proposed by the benchmarking participants.

The benchmark suites here presented can be used by tool developers to evaluate the RDF(S) import and export capabilities of their tools, and the interoperability between their tools and other tools using RDF(S) as ontology interchange language. These benchmark suites can also be used by end users of the tools to help them to choose between several tools, as executing the benchmark suites does not require to have a deep knowledge about the tools.

During the experimentation, the execution of the benchmark suites was performed mainly manually. Nevertheless, a Java application was developed at the UPM in order to

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

automate the execution of the benchmark suites for the WebODE ontology engineering workbench [Arpírez *et al.*, 2003]. Chapter 3 presents this application as an example on how to automate the execution of the benchmark suites and is intended to serve as an inspiration for further implementations for other systems.

Finally, Appendices A.1 and A.2 include the complete definition of the benchmark suites.

Chapter 2

Benchmark suites for evaluating interoperability

This chapter presents the benchmark suites that are being used in the benchmarking of the interoperability of ontology development tools using RDF(S) for ontology interchange.

Evaluating the interoperability of ontology development tools using RDF(S) for ontology interchange requires that the importers and exporters from/to RDF(S) of these tools work accurately in order to interchange ontologies correctly. Therefore, this evaluation includes two consecutive steps:

- To evaluate the RDF(S) importers and exporters of ontology development tools using the RDF(S) Import Benchmark Suite (presented in Section 2.1) and the RDF(S) Export Benchmark Suite (presented in Section 2.2) respectively.
- To evaluate the ontology interchange between ontology development tools using the RDF(S) Interoperability Benchmark Suite (presented in Section 2.3).

Different benchmark suites must be used for evaluating the import and the export from/to RDF(S). The reason of this is that the ontologies to consider when evaluating the import of ontologies from RDF(S) to an ontology development tool and the ontologies to consider when evaluating the export of ontologies from an ontology development tool to RDF(S) must be modelled according to different knowledge models: the RDF(S) knowledge model in the case of the import and a common knowledge model of the ontology development tools in the case of the export. For example, in ontology development tools both object and datatype properties can be used for modelling ontologies while in RDF(S) only properties can be used. If the tools to evaluate were RDF(S)-based, only one benchmark suite should be defined, as the knowledge models of the tools and RDF(S) would be the same. Nevertheless, this is not the case when considering ontology development tools.

When evaluating the export and the interoperability, as the knowledge model taken

into account for defining the ontologies has been a common knowledge model of the ontology development tools, the ontologies to consider are the same. Nevertheless, the benchmark suites are different as their intended use, their input, their results, and the process to follow for executing them are different.

The three benchmark suites presented in this chapter follow the same structure, as they were designed according to the same general requirements. These requirements state that the benchmark suites must:

- Be simple and interpretable, providing different ways of representing the benchmarks: in natural language, in the RDF/XML syntax, graphically, etc.
- Be easy to use by both ontology tool users and developers.
- Be defined at a high level of abstraction, so they are not biased towards a certain tool or tools.
- Represent the different structures of ontologies commonly used when developing ontologies.

2.1 RDF(S) Import Benchmark Suite

The RDF(S) Import Benchmark Suite is a benchmark suite that can be used to evaluate the RDF(S) import functionalities of Semantic Web tools. Altough it was developed having in mind ontology development tools, it can be used to evaluate any other tool capable of importing RDF(S).

The RDF(S) Import Benchmark Suite is composed of 82 benchmarks that check the correct import of RDF(S) ontologies. These benchmarks can be of two types, those that depend on the RDF(S) knowledge model, and those that depend on the RDF(S) syntax chosen:

- **Benchmarks that depend on the knowledge model.** These benchmarks check the correct import of RDF(S) ontologies that model simple combinations of the components of the RDF(S) knowledge model. The RDF(S) components considered in this benchmark suite are the most frequently used for modelling ontologies in ontology development tools: *rdfs:Class*, *rdf:Property*, *rdfs:Literal*, *rdf:type*, *rdfs:subClassOf*, *rdfs:subPropertyOf*, *rdfs:domain*, and *rdfs:range*; not dealing with the rest of the RDF(S) components. This group of benchmarks is classified in the following categories:
  - Class benchmarks
  - Metaclass benchmarks
  - Subclass benchmarks

  - Class and property benchmarks
  - Single property benchmarks
  - Subproperty benchmarks
  - Property with domain and range benchmarks
  - Instance benchmarks
  - Instance and property benchmarks

- **Benchmarks that depend on the syntax.** These benchmarks check the correct import of RDF(S) ontologies with the different variants of the RDF/XML syntax, as stated in the RDF/XML specification. This group of benchmarks is classified in the following categories:

  - URI reference benchmarks
  - Empty node benchmarks
  - Multiple properties benchmarks
  - Types node benchmarks
  - String literal benchmarks
  - Blank node benchmarks
  - Language identification benchmarks

The method followed for defining the RDF(S) Import Benchmark Suite is described in [García-Castro and Gómez-Pérez, 2005].

Appendix A.1 includes the complete definition of the RDF(S) Import Benchmark Suite. Each benchmark is defined according to:

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

The RDF(S) Import Benchmark Suite definition is available in a public web page[1], all the RDF(S) files to import can be downloaded from a single file[2], and templates are provided for collecting the execution results[3].

In order to execute the benchmarks, the steps to follow are:

[1] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/rdfs_import_benchmark_suite.html

[2] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/files/import_files.zip

[3] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/templates/RDFS_Import_Benchmark_Suite_Template.xls

1. To model into the tool the expected ontology resulting from importing the RDF(S) ontology.
2. To import the file with the RDF(S) ontology into the tool.
3. To compare the imported ontology with the expected ontology and to check whether they are the same.

Although these steps can be performed manually, some automatic means of performing them (or part of them) is highly advised when dealing with many benchmarks, especially for comparing the expected and imported ontologies.

The evaluation criteria used for the benchmark suite are:

- **Modeling** (YES/NO). The tool is able to model the ontology components described in the benchmark.
- **Execution** (OK/FAIL). The execution of the benchmark is normally carried out without any problem, and the tool always produces its expected result. In the case of a failed execution, further information is required:
  - The reasons for failing the benchmark execution.
  - If the tool was corrected to pass a benchmark, the changes performed.
- **Information added or lost**. The information added or lost in the ontology interchange when executing the benchmark.

2.2 RDF(S) Export Benchmark Suite

The RDF(S) Export Benchmark Suite is a benchmark suite that can be used to evaluate the RDF(S) export functionalities of Semantic Web tools. Altough it was developed having in mind ontology development tools, it can be used to evaluate any other tool capable of exporting to RDF(S).

The RDF(S) Export Benchmark Suite is composed of 66 benchmarks that check the correct export of ontologies to RDF(S). These benchmarks can be of two types, those that depend on the knowledge model of the tools, and those that depend on the restrictions of RDF(S) for representing certain characters in URIs:

- **Benchmarks that depend on the knowledge model.** These benchmarks check the correct export to RDF(S) of ontologies that model simple combinations of the components of the knowledge model of the tools. The components considered in these benchmarks are the most frequently used for modelling ontologies in ontology development tools, and are present in the knowledge models of these tools:

classes and class hierarchies, object and datatype properties, instances, and literals; not dealing with the rest of the components specific to each tool. This group of benchmarks is classified in the following categories:

  - Class benchmarks
  - Metaclass benchmarks
  - Subclass benchmarks
  - Class and object property benchmarks
  - Class and datatype property benchmarks
  - Object property benchmarks
  - Datatype property benchmarks
  - Instance benchmarks
  - Instance and object property benchmarks
  - Instance and datatype property benchmarks

- **Benchmarks that depend on character restrictions.** These benchmarks check the correct export to RDF(S) of ontologies with concepts and properties whose names include characters that are not allowed for representing RDF(S) or XML URIs. This group of benchmarks is classified in the following categories:

  - Concepts and properties whose names start with a character that is not a letter or '_'
  - Concepts and properties with spaces in their names
  - Concepts and properties with URI reserved characters in their names (';', '/', '?', ':', '@', '&', '=', '+', '$', ',')
  - Concepts and properties with XML delimiter characters in their names ('¡', '¿', '#', '%', '"')

The method used to define this benchmark suite is similar to the one used for defining the RDF(S) Import Benchmark Suite [García-Castro and Gómez-Pérez, 2005].

Appendix A.2 includes the complete definition of the RDF(S) Export Benchmark Suite. Each benchmark is defined according to:

- An **identifier** for tracking the different benchmarks.
- A **description** of the benchmark in natural language.
- A **graphical representation** of the ontology to be imported in the benchmark.
- The **instantiation** of the ontology described in the benchmark for the tool, using the vocabulary and the components of the tool.

The RDF(S) Export Benchmark Suite definition is available in a public web page[4] and templates are provided for collecting the execution results[5].

In order to execute the benchmarks, the steps to follow are:

1. To define in RDF(S) the expected ontology resulting from exporting the ontology.
2. To model into the tool the ontology described in the benchmark.
3. To export the ontology modelled using the tool to RDF(S).
4. To compare the exported RDF(S) ontology with the expected RDF(S) ontology to check whether they are the same.

Although these steps can be performed manually, some automatic means of performing them (or part of them) is highly advised when dealing with many benchmarks, especially for comparing the expected and exported ontologies.

The evaluation criteria used for the benchmark suite are:

- **Modeling** (YES/NO). The tool is able to model the ontology components described in the benchmark.
- **Execution** (OK/FAIL/N.E.). The execution of the benchmark is normally carried out without any problem, and the tool always produces its expected result. As there may be a benchmark that defines an ontology that cannot be modelled in a certain tool, the result can also be *N.E.* (Non Executed) meaning that, as the tool cannot model the ontology, the benchmark cannot be executed. In the case of a failed execution, further information is required:
  - The reasons for failing the benchmark execution.
  - If the tool was corrected to pass a benchmark, the changes performed.
- **Information added or lost**. The information added or lost in the ontology interchange when executing the benchmark.

2.3 RDF(S) Interoperability Benchmark Suite

The RDF(S) Interoperability Benchmark Suite is a benchmark suite that can be used to evaluate the interoperability of Semantic Web tools using RDF(S) as interchange language

[4] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/rdfs_export_benchmark_suite.html

[5] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/templates/RDFS_Export_Benchmark_Suite_Template.xls

by testing the interchange of ontologies, from one origin tool to a destination one, and vice versa. Altough it was developed having in mind ontology development tools, it can be used to evaluate any other tool capable of importing from and exporting to RDF(S).

The RDF(S) Interoperability Benchmark Suite is composed of 66 benchmarks that check the correct interchange of ontologies between two tools. The benchmark suite considers the interchange of a common core of the knowledge modelling components most frequently used for modelling ontologies: classes and class hierarchies, object and datatype properties, instances, and literals. As these components are the same as those in the RDF(S) Export Benchmark Suite, the definition of the RDF(S) Interoperability Benchmark Suite is identical to the RDF(S) Export Benchmark Suite, as presented in Appendix A.2.

The RDF(S) Interoperability Benchmark Suite definition is available in a public web page[6] and templates are provided for collecting the execution results[7]. In the case of evaluating the interoperability from the tools that have already executed the RDF(S) Export Benchmark Suite, a file can be downloaded which contains the RDF(S) files exported by these tools[8].

In order to execute the benchmarks, the steps to follow are:

1. To define in the destination tool the expected ontology resulting from interchanging the ontology.
2. To model into the origin tool the ontology described in the benchmark.
3. To export the ontology modelled using the origin tool to RDF(S).
4. To import the RDF(S) file exported by the origin tool into the destination tool.
5. To compare the interchanged ontology with the expected ontology and to check whether they are the same.

Although these steps can be performed manually, some automatic means of performing them (or part of them) is highly advised when dealing with many benchmarks, especially for comparing the expected and interchanged ontologies.

The evaluation criteria used for the benchmark suite are:

- **Modeling** (YES/NO). The tool is able to model the ontology components described in the benchmark.

[6] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/rdfs_interoperability_benchmark_suite.html

[7] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/templates/Interoperability Templates.xls

[8] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/stage_1_results/RDFS Exported Files.zip

- **Execution** (OK/FAIL/N.E.). The execution of the benchmark is normally carried out without any problem, and the tool always produces its expected result. As there may be a benchmark that defines an ontology that cannot be modelled in a certain tool, the result can also be *N.E.* (Non Executed) meaning that, as the tool cannot model the ontology, the benchmark cannot be executed. In the case of a failed execution, further information is required:
  - The reasons for failing the benchmark execution.
  - If the tool was corrected to pass a benchmark, the changes performed.
- **Information added or lost**. The information added or lost in the ontology interchange when executing the benchmark.

Chapter 3

Automating the execution of the experiments for WebODE

The *rdfsbs* Java application was developed to diminish the effort needed for executing the benchmark suites over the WebODE ontology engineering workbench[1] and to provide an easy execution of these benchmark suites. This application allows to automatically perform most of the benchmarking experimentation in WebODE.

Although this application can be only used with WebODE, it can serve as an inspiration for automating the execution of the benchmark suites on other systems.

The tasks that the *rdfsbs* application automates are the following:

- To import RDF(S) files into WebODE.
- To export WebODE ontologies to RDF(S) files.
- To create text files describing all the components of ontologies for comparing ontologies, so as to avoid the manual inspection of the ontologies using WebODE.
- To create in WebODE the ontologies described in the benchmarks using WebODE's ontology management API.

The *rdfsbs* application is specific to WebODE, as it uses WebODE's ontology management API for creating ontologies and ontology descriptions and for importing and exporting ontologies.

[1] http://webode.dia.fi.upm.es/

3.1 Software requirements

The requirements for executing the *rdfsbs* application are those required when the WebODE system is running, and these are the following:

- Windows 2000/XP
- Java 1.4.2
- Oracle 8.1.7
- Minerva version 1 build 15
- WebODE version 2 build 110

3.2 Running the *rdfsbs* application

The source code of the *rdfsbs* application can be downloaded from the Knowledge Web portal[2]. Using the application does not require a specific installation but just to compile the source code.

rdfsbs is a command-line application. It requires one argument that can be either *-import* or *-export*. If executed with the *-import* argument, it executes the benchmarks of the RDF(S) Import Benchmark Suite, and if executed with the *-export* argument, it executes the benchmarks of the RDF(S) Export Benchmark Suite. In the case of inserting a wrong argument or an incorrect number of arguments, it exits with an error message and shows how to use it:

```
Usage: RunBenchmarks <argument>
       -import : Run RDF(S) import benchmarks
       -export : Run RDF(S) export benchmarks
```

3.3 Using the *rdfsbs* application in the benchmarking

This section describes how the *rdfsbs* application can be used to perform the different experiments required in the interoperability benchmarking.

The steps to follow for executing the RDF(S) Import Benchmark Suite are the following:

[2] http://knowledgeweb.semanticweb.org/semanticportal/rdfsbs_v1.0_source.zip

1. To define in natural language the expected ontologies resulting from importing the RDF(S) files.
2. To execute the *rdfsbs* application with the *-import* argument. The application imports the RDF(S) files into WebODE and, for each imported ontology, it creates a text file with the description of this imported ontology.
3. To manually compare the description of the imported ontologies with the expected ontologies defined in natural language and to check whether they are the same.

The steps to follow for executing the RDF(S) Export Benchmark Suite are the following:

1. To define in RDF(S) the expected ontologies resulting from exporting the ontologies defined in the benchmarks.
2. To execute the *rdfsbs* application with the *-export* argument. The application creates the ontologies described in the benchmarks into WebODE and exports these ontologies to RDF(S) files.
3. To manually compare the exported RDF(S) ontologies with the expected RDF(S) ontologies to check whether they are the same.

The steps to follow for executing the RDF(S) Interoperability Benchmark Suite are the same that for executing the RDF(S) Import Benchmark Suite. In the RDF(S) Interoperability Benchmark Suite the RDF(S) files to import are those that were previously exported to RDF(S) by the other tools.

Appendix A

Definition of the benchmark suites

A.1 RDF(S) Import Benchmark Suite

The 82 benchmarks that compose the RDF(S) Import Benchmark Suite are defined in Table A.1.

Table A.1: Definition of the import benchmarks

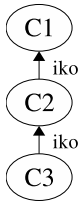
Id	**Description**	**Graphical representation**
Class benchmarks		
I01	Import just one class	
I02	Import several classes with no properties between them	
Metaclass benchmarks		
I03	Import one class that is instance of another class, being this last class instance of a third one	

Table A.1 – continued from previous page

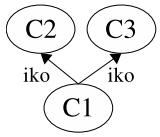
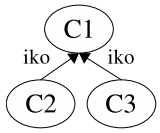
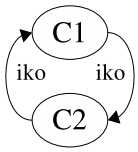
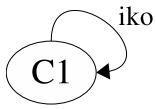
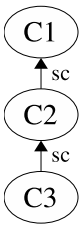
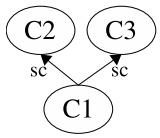
Id	Description	Graphical representation
I04	Import one class that is instance of several classes	
I05	Import several classes that are instance of the same class	
I06	Import one class that is instance of another class and viceversa	
I07	Import just one class that is instance of himself	
Subclass benchmarks		
I08	Import one class that is subclass of another class, being this last class subclass of a third one	
I09	Import one class that is subclass of several classes	

Table A.1 – continued from previous page

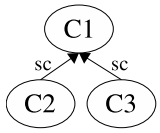
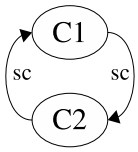
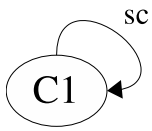
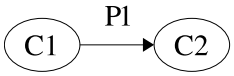
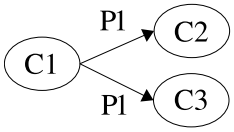
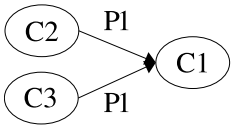
Id	Description	Graphical representation
I10	Import several classes that are subclass of the same class	
I11	Import one class that is subclass of another class and viceversa, forming a cycle	
I12	Import just one class that is subclass of himself, forming a cycle	
Class and property benchmarks		
I13	Import one class that has a property with another class. The property is supposed to be defined with a domain and a range of some metaclass of the classes (such as rdfs:Class)	
I14	Import one class that has the same property with several classes. The property is supposed to be defined with a domain and a range of some metaclass of the classes (such as rdfs:Class)	
I15	Import several classes that have the same property with the same class. The property is supposed to be defined with a domain and a range of some metaclass of the classes (such as rdfs:Class)	

Table A.1 – continued from previous page

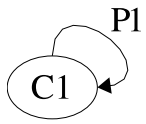
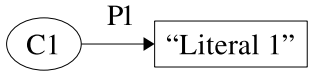
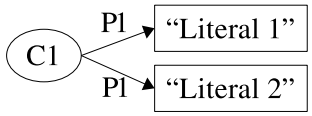
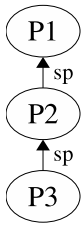
Id	Description	Graphical representation
I16	Import just one class that has a property with itself. The property is supposed to be defined with a domain and a range of some metaclass of the classes (such as rdfs:Class)	
I17	Import just one class that has a property with a literal. The property is supposed to be defined with a domain and a range of some metaclass of the classes (such as rdfs:Class)	
I18	Import just one class that has the same property with several literals. The property is supposed to be defined with a domain and a range of some metaclass of the classes (such as rdfs:Class)	
Single property benchmarks		
I19	Import just one property	
I20	Import several properties	
Subproperty benchmarks		
I21	Import one property that is subproperty of another property that is subproperty of a third one	

Table A.1 – continued from previous page

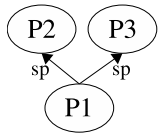
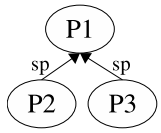
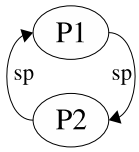
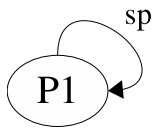
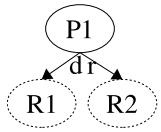
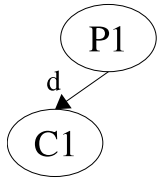
Id	Description	Graphical representation
I22	Import one property that is subproperty of several properties	
I23	Import several properties that are subproperty of the same property	
I24	Import one property that is subproperty of another property and viceversa	
I25	Import just one property that is subproperty of himself	
Property with domain and range benchmarks		
I26	Import just one property that has as domain a resource and as range another resource, without the resource definitions	
I27	Import just one property that has as domain a class, with the class defined in the ontology	

Table A.1 – continued from previous page

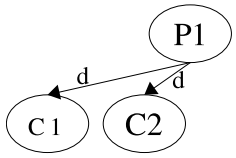
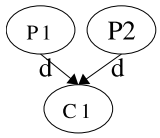
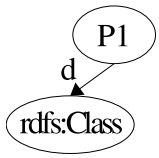
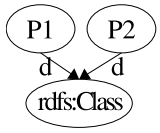
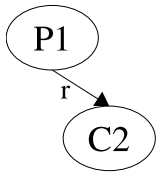
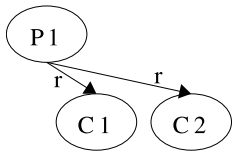
Id	Description	Graphical representation
I28	Import just one property that has as domain several classes, with the classes defined in the ontology	
I29	Import several properties that have as domain the same class, with the class defined in the ontology	
I30	Import just one property that has as domain rdfs:Class	
I31	Import several properties that have as domain rdfs:Class	
I32	Import just one property that has as range a class, with the class defined in the ontology	
I33	Import just one property that has as range several classes, with the classes defined in the ontology	

Table A.1 – continued from previous page

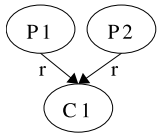
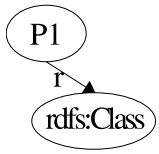
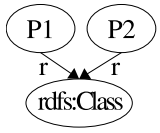
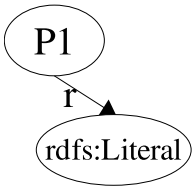
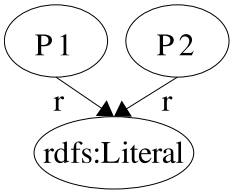
Id	Description	Graphical representation
I34	Import several properties that have as range the same class, with the class defined in the ontology	
I35	Import just one property that has as range rdfs:Class	
I36	Import several properties that have as range rdfs:Class	
I37	Import just one property that has as range rdfs:Literal	
I38	Import several properties that have as range rdfs:Literal	

Table A.1 – continued from previous page

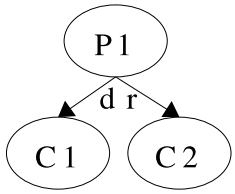
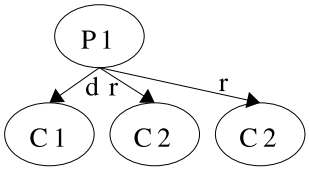
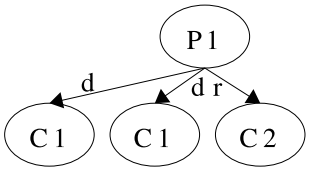
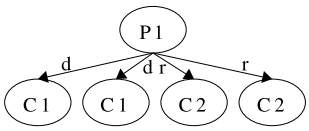
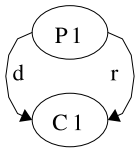
Id	Description	Graphical representation
I39	Import just one property that has as domain a class and as range another class, with the classes defined in the ontology	
I40	Import just one property that has as domain a class and as range several classes, with the classes defined in the ontology	
I41	Import just one property that has as domain several classes and as range a class, with the classes defined in the ontology	
I42	Import just one property that has as domain several classes and as range several classes, with the classes defined in the ontology	
I43	Import just one property that has as domain and range the same class, with the class defined in the ontology	

Table A.1 – continued from previous page

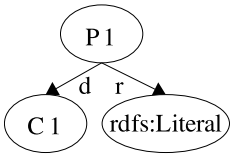
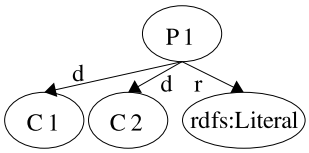
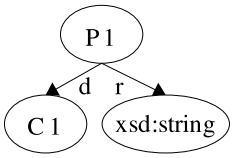
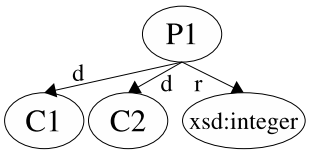
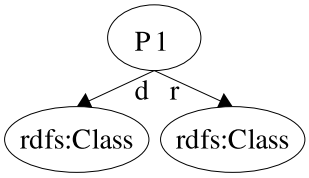
Id	Description	Graphical representation
I44	Import just one property that has as domain a class and as range rdfs:Literal, with the class defined in the ontology	
I45	Import just one property that has as domain several classes and as range rdfs:Literal, with the classes defined in the ontology	
I46	Import just one property that has as domain a class and as range the XML Schema datatype ”string”, with the class defined in the ontology	
I47	Import just one property that has as domain several classes and as range the XML Schema datatype ”integer”, with the classes defined in the ontology	
I48	Import just one property that has as domain rdfs:Class and as range rdfs:Class	

Table A.1 – continued from previous page

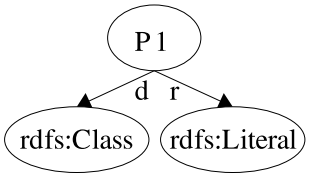
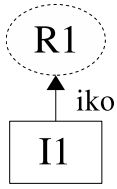
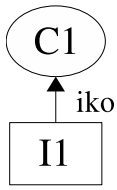
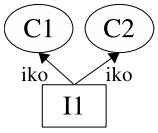
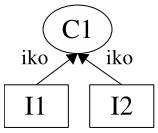
Id	Description	Graphical representation
I49	Import just one property that has as domain rdfs:Class and as range rdfs:Literal	
Instance benchmarks		
I50	Import just one instance of a resource, without the resource definition	
I51	Import one class and one instance of the class	
I52	Import several classes and one instance of all of them	
I53	Import one class and several instances of the class	
Instance and property benchmarks		

Table A.1 – continued from previous page

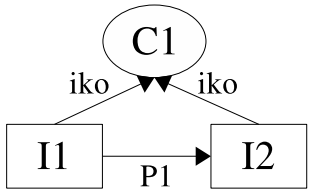
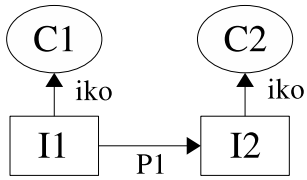
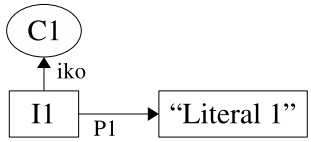
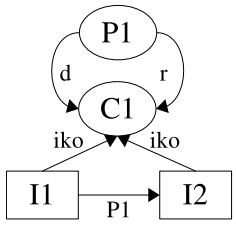
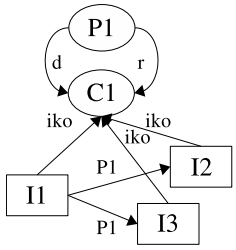
Id	Description	Graphical representation
I54	Import one class and one instance of the class that has a property with another instance of the same class, without the property definition	
I55	Import two classes and one instance of one class that has a property with an instance of the other class, without the property definition	
I56	Import one class and one instance of the class that has a property with a literal, without the property definition	
I57	Import one class, one property with domain and range the class, and one instance of the class that has the property with another instance of the same class	
I58	Import one class, one property with domain and range the class, and one instance of the class that has the property with several instances of the class	

Table A.1 – continued from previous page

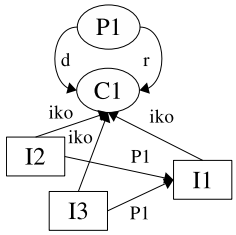
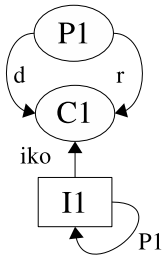
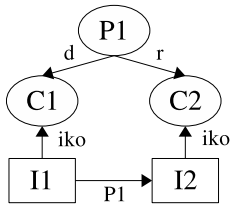
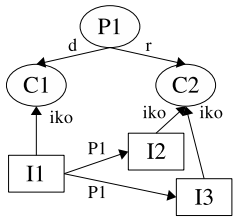
Id	Description	Graphical representation
I59	Import one class, one property with domain and range the class, and several instances of the class that have the property with the same instance of the class	
I60	Import one class, one property with domain and range the class, and one instance of the class that has the property with himself	
I61	Import two classes, one property with domain one class and range the other class, and one instance of one class that has the property with an instance of the other class	
I62	Import two classes, one property with domain one class and range the other class, and one instance of one class that has the property with several instances of the other class	

Table A.1 – continued from previous page

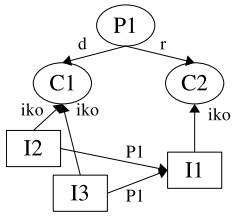
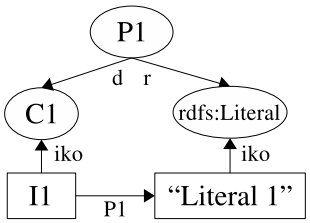
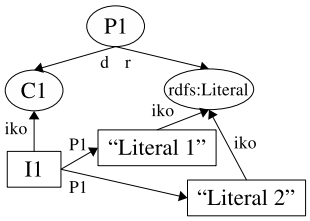
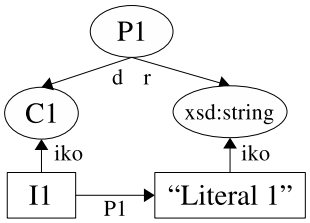
Id	Description	Graphical representation
I63	Import two classes, one property with domain one class and range the other class, and several instances of one class that have the property with the same instance of the other class	
I64	Import one class, one property with domain the class and range rdfs:Literal, and one instance of the class that has the property with a literal	
I65	Import one class, one property with domain the class and range rdfs:Literal, and one instance of the class that has the property with several literals	
I66	Import one class, one property with domain the class and range the XML Schema datatype ”string”, and one instance of the class that has the property with a value	

Table A.1 – continued from previous page

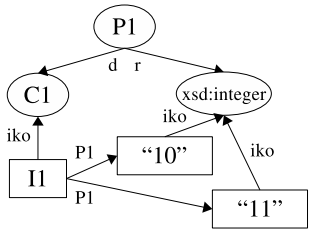
Id	Description	Graphical representation
I67	Import one class, one property with domain the class and range the XML Schema datatype ”integer”, and one instance of the class that has the property with several integer values	
Syntax and abbreviation benchmarks		
URI reference benchmarks		
I68	Import several resources with absolute URI references	
I69	Import several resources with URI references relative to a base URI	
I70	Import several resources with URI references transformed from rdf:ID attribute values	
I71	Import several resources with URI references relative to an ENTITY declaration	
Empty node benchmarks		
I72	Import several resources with empty nodes	
I73	Import several resources with empty nodes shortened	
Multiple properties benchmarks		
I74	Import several resources with multiple properties	
I75	Import several resources with multiple properties shortened	
Typed node benchmarks		
I76	Import several resources with typed nodes	
I77	Import several resources with typed nodes shortened	
String literal benchmarks		
I78	Import several resources with properties with string literals	
I79	Import several resources with properties with string literals as XML attributes	
Blank node benchmarks		
I80	Import several resources with blank nodes with identifier	

Table A.1 – continued from previous page

Id	Description	Graphical representation
I81	Import several resources with blank nodes shortened	
Language identification benchmarks		
I82	Import several resources with properties with xml:lang attributes	

A.2 RDF(S) Export and Interoperability Benchmark Suites

The 66 benchmarks that compose the RDF(S) Export Benchmark Suite and the RDF(S) Interoperability Benchmark Suite are defined in Table A.2.

Table A.2: Definition of the export benchmarks

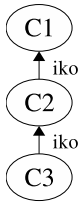
Id	Description	Graphical representation
Class benchmarks		
E01	Export just one class	
E02	Export several classes	
Metaclass benchmarks		
E03	Export one class that is instance of another class that is instance of a third one	

Table A.2 – continued from previous page

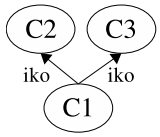
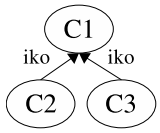
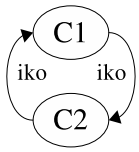
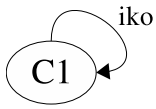
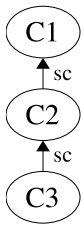
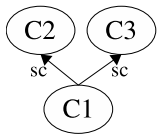
Id	Description	Graphical representation
E04	Export one class that is instance of several classes	
E05	Export several classes that are instance of the same class	
E06	Export one class that is instance of another class and viceversa	
E07	Export just one class that is instance of himself	
Subclass benchmarks		
E08	Export one class that is subclass of another class that is subclass of a third one	
E09	Export one class that is subclass of several classes	

Table A.2 – continued from previous page

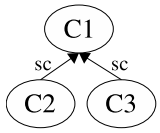
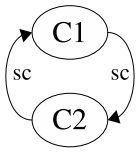
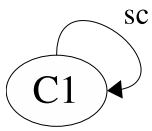
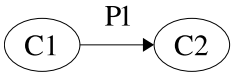
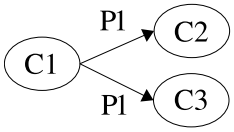
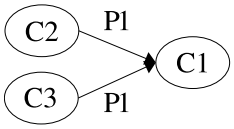
Id	Description	Graphical representation
E10	Several classes that are subclass of the same class	
E11	Export one class that is subclass of another class and viceversa, forming a cycle	
E12	Export just one class that is subclass of himself, forming a cycle	
Class and object property benchmarks		
E13	Export one class that has an object property with another class. The property is supposed to be defined with a domain and a range of some metaclass of the classes	
E14	Export one class that has the same object property with several classes. The property is supposed to be defined with a domain and a range of some metaclass of the classes	
E15	Export several classes that have the same object property with the same class. The property is supposed to be defined with a domain and a range of some metaclass of the classes	

Table A.2 – continued from previous page

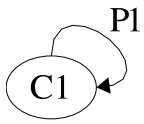
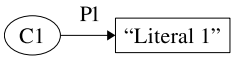
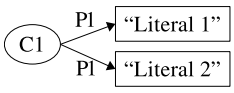
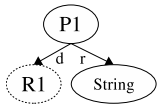
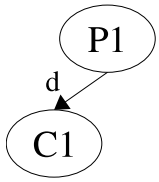
Id	Description	Graphical representation
E16	Export just one class that has an object property with itself. The property is supposed to be defined with a domain and a range of some metaclass of the class	
Class and datatype property benchmarks		
E17	Export just one class that has a datatype property with a literal. The property is supposed to be defined with a domain and a range of some metaclass of the class	
E18	Export just one class that has the same datatype property with several literals. The property is supposed to be defined with a domain and a range of some metaclass of the class	
Datatype property benchmarks		
E19	Export just one datatype property	
E20	Export several datatype properties	
E21	Export just one datatype property that has as domain a resource and as range "String", without the resource definition	
E22	Export just one datatype property that has as domain a class, with the class defined in the ontology	

Table A.2 – continued from previous page

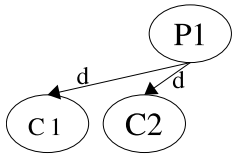
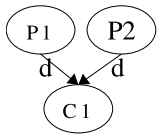
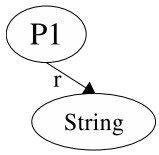
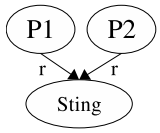
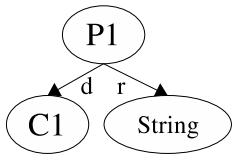
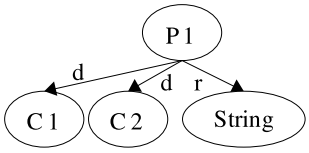
Id	Description	Graphical representation
E23	Export just one datatype property that has as domain several classes, with the classes defined in the ontology	
E24	Export several datatype properties that have as domain the same class, with the class defined in the ontology	
E25	Export just one datatype property that has as range ”String”	
E26	Export several datatype properties that have as range ”String”	
E27	Export one datatype property that has as domain a class and as range ”String”, with the class defined in the ontology	
E28	Export one datatype property that has as domain several classes and as range ”String”, with the classes defined in the ontology	

Table A.2 – continued from previous page

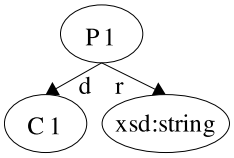
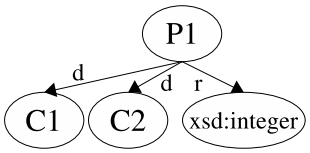
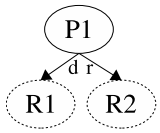
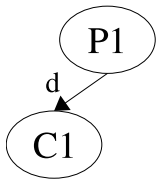
Id	Description	Graphical representation
E29	Export one datatype property that has as domain a class and as range the XML Schema datatype ”string”, with the class defined in the ontology	
E30	Export one datatype property that has as domain several classes and as range the XML Schema datatype ”integer”, with the classes defined in the ontology	
Object property benchmarks		
E31	Export just one object property	
E32	Export several object properties	
E33	Export just one object property that has as domain a resource and as range another resource, without the resource definitions	
E34	Export just one object property that has as domain a class, with the class defined in the ontology	

Table A.2 – continued from previous page

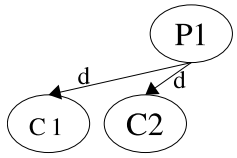
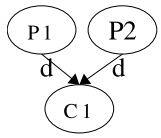
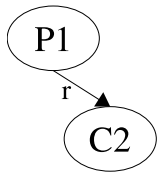
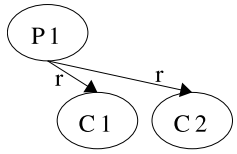
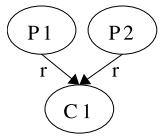
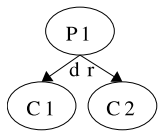
Id	Description	Graphical representation
E35	Export just one object property that has as domain several classes, with the classes defined in the ontology	
E36	Export several object properties that have as domain the same class, with the class defined in the ontology	
E37	Export just one object property that has as range a class, with the class defined in the ontology	
E38	Export just one object property that has as range several classes, with the classes defined in the ontology	
E39	Export several object properties that have as range the same class, with the class defined in the ontology	
E40	Export just one object property that has as domain a class and as range another class, with the classes defined in the ontology	

Table A.2 – continued from previous page

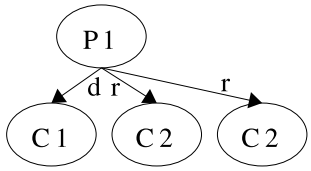
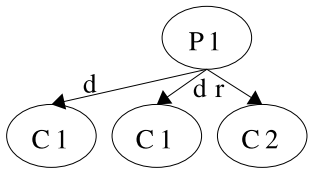
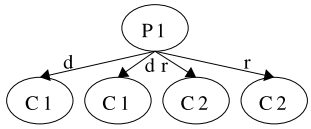
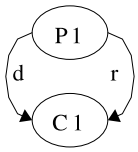
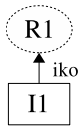
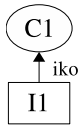
Id	Description	Graphical representation
E41	Export just one object property that has as domain a class and as range several classes, with the classes defined in the ontology	
E42	Export just one object property that has as domain several classes and as range a class, with the classes defined in the ontology	
E43	Export just one object property that has as domain several classes and as range several classes, with the classes defined in the ontology	
E44	Export just one object property that has as domain and range the same class, with the class defined in the ontology	
Instance benchmarks		
E45	Export just one instance of a resource, without the resource definition	
E46	Export one class and one instance of the class	

Table A.2 – continued from previous page

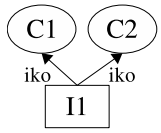
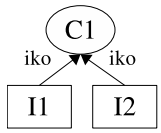
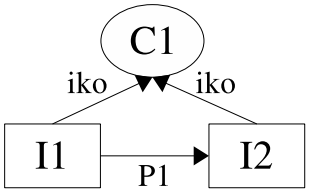
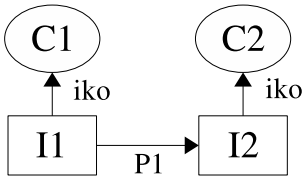
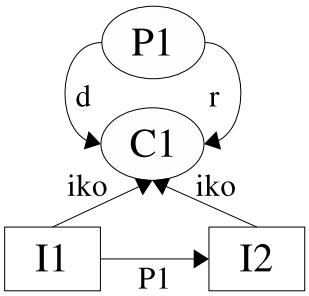
Id	Description	Graphical representation
E47	Export several classes and one instance of all of them	
E48	Export one class and several instances of the class	
Instance and object property benchmarks		
E49	Export one class and one instance of the class that has an object property with another instance of the same class, without the property definition	
E50	Export two classes and one instance of one class that has an object property with an instance of the other class, without the property definition	
E51	Export one class, one object property with domain and range the class, and one instance of the class that has the property with another instance of the same class	

Table A.2 – continued from previous page

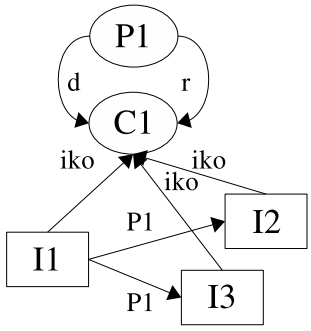
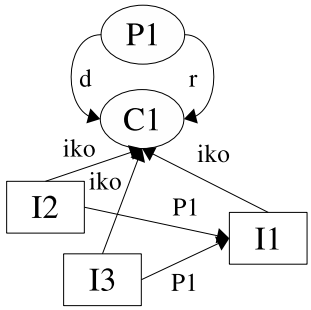
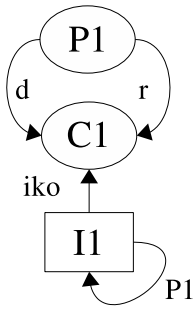
Id	Description	Graphical representation
E52	Export one class, one object property with domain and range the class, and one instance of the class that has the property with several instances of the class	
E53	Export one class, one object property with domain and range the class, and several instances of the class that have the property with the same instance of the class	
E54	Export one class, one object property with domain and range the class, and one instance of the class that has the property with himself	

Table A.2 – continued from previous page

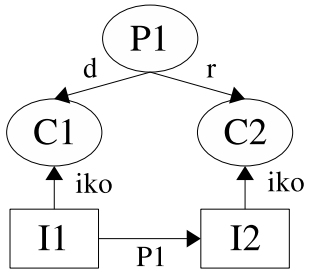
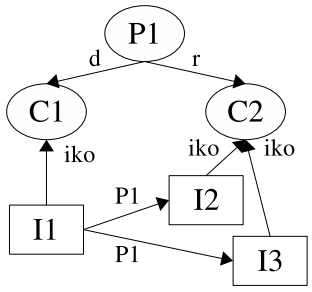
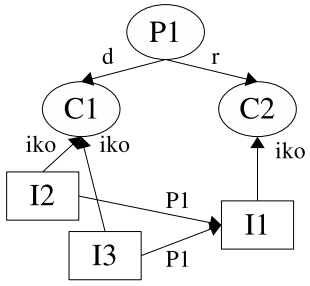
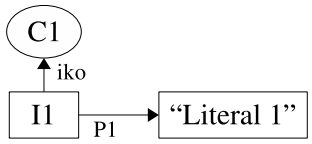
Id	Description	Graphical representation
E55	Export two classes, one object property with domain one class and range the other class, and one instance of one class that has the property with an instance of the other class	
E56	Export two classes, one object property with domain one class and range the other class, and one instance of one class that has the property with several instances of the other class	
E57	Export two classes, one object property with domain one class and range the other class, and several instances of one class that have the property with the same instance of the other class	
Instance and datatype property benchmarks		
E58	Export one class and one instance of the class that has a datatype property with a literal, without the property definition	

Table A.2 – continued from previous page

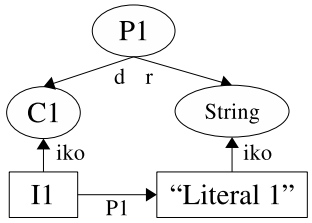
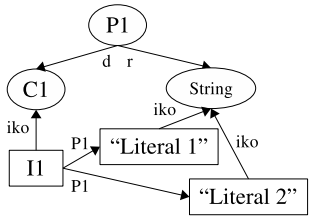
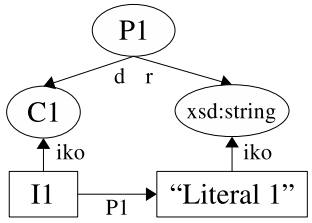
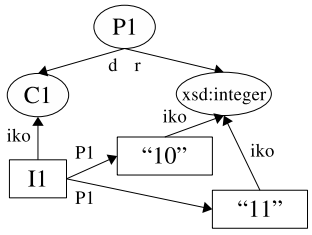
Id	Description	Graphical representation
E59	Export one class, one datatype property with domain the class and range ”String”, and one instance of the class that has the property with a literal	
E60	Export one class, one datatype property with domain the class and range ”String”, and one instance of the class that has the property with several literals	
E61	Export one class, one datatype property with domain the class and range the XML Schema datatype ”string”, and one instance of the class that has the property with a value	
E62	Export one class, one datatype property with domain the class and range the XML Schema datatype ”integer”, and one instance of the class that has the property with several integer values	
URI character restrictions		
Concepts and properties whose names start with a character that is not a letter or ’_’		

Table A.2 – continued from previous page

Id	Description	Graphical representation
E63	Export an ontology containing two classes named ”1class” and ”2class”, each with one datatype property of type String named ”-datatypeProperty1” and ”-datatypeProperty2” respectively, and an object property between the classes named ”.objectProperty”	
Concepts and properties with spaces in their names		
E64	Export an ontology containing two classes named ”class 1” and ”class 2”, each with one datatype property of type String named ”datatype property 1” and ”datatype property 2” respectively, and an object property between the classes named ”object property”	
Concepts and properties with URI reserved characters in their names (’;’, ’/’, ’?’, ’:’, ’@’, ’&’, ’=’, ’+’, ’$’, ’,’)		
E65	Export an ontology containing two classes named ”concept/1” and ”concept:1”, each with one datatype property of type String named ”datatype/property/1” and ”datatype=property=2” respectively, and an object property between the classes named ”object$property”	
Concepts and properties with XML delimiter characters in their names (’¡’, ’¿’, ’#’, ’%’, ””)		
E66	Export an ontology containing two classes named ”class¡1” and ”class¿1”, each with one datatype property of type String named ”datatype#property#1” and ”datatype%property%2” respectively, and an object property between the classes named ”object”property”	

Bibliography

[Arpírez *et al.*, 2003] J.C. Arpírez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez. WebODE in a nutshell. *AI Magazine*, 24(3):37–47, Fall 2003.

[García-Castro and Gómez-Pérez, 2005] R. García-Castro and A. Gómez-Pérez. A method for performing an exhaustive evaluation of RDF(S) importers. In *Proceedings of the Workshop on Scalable Semantic Web Knowledge Based Systems (SSWS2005)*, number 3807 in LNCS, pages 199–206, New York, USA, November 2005. Springer-Verlag.

[García-Castro *et al.*, 2004] R. García-Castro, D. Maynard, H. Wache, D. Foxvog, and R. González-Cabero. D2.1.4 specification of a methodology, general criteria and benchmark suites for benchmarking ontology tools. Technical report, Knowledge Web, December 2004.

[Wache *et al.*, 2004] H. Wache, L. Serafini, and R. García-Castro. D2.1.1 survey of scalability techniques for reasoning with ontologies. Technical report, Knowledge Web, July 2004.

Acknowledgments

Thanks to Rosario Plaza for reviewing the grammar of this deliverable. Thanks to all the people that are participating in the interoperability benchmarking by the time of writing this deliverable: Olivier Corby, York Sure, Moritz Weiten, and Markus Zondler.

Related deliverables

A number of Knowledge web deliverables are clearly related to this one:

Project	Number	**Title** and relationship
KW	D2.1.1	**Survey of scalability techniques for reasoning with ontologies** provided an overview on benchmarking and its main related areas: software experimentation and measurement. It also presented a state of the art on the evaluation of ontology technology.
KW	D2.1.4	**Specification of a methodology, general criteria and benchmark suites for benchmarking ontology tools** presented the benchmarking methodology that is being used for benchmarking the interoperability of ontology development tools.