D2.1.4 Specification of a methodology, general criteria, and benchmark suites for benchmarking ontology tools

Raúl García-Castro (Universidad Politécnica de Madrid)

with contributions from:
Diana Maynard (University of Sheffield)
Holger Wache (Vrije Universiteit Amsterdam)
Doug Foxvog (National University of Ireland Galway)
Rafael González Cabero (Universidad Politécnica de Madrid)

Abstract.
EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB
Deliverable D2.1.4 (WP2.1)

This deliverable proposes a benchmarking framework to be used in the benchmarking activities that will be performed in Knowledge Web. This framework includes a benchmarking methodology, guidelines for building benchmark suites, a list of tools that can be useful when performing benchmarking, and specific considerations for benchmarking the different types of tools that will be considered in workpackage 2.1 (ontology development tools, ontology-based annotation tools, ontology-based reasoning tools, and semantic web service technology).
Keyword list: benchmarking, methodology, benchmark suite, scalability, robustness, interoperability

Document Identifier	KWEB/2004/D2.1.4/v1.5
Project	KWEB EU-IST-2004-507482
Version	v1.5
Date	8. February, 2005
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

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

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

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

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

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

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

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

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

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	08.10.04	Raúl García-Castro	created
0.2	01.11.04	Raúl García-Castro	updated with Diana Maynard's contribution (chapter 6)
0.3	14.11.04	Raúl García-Castro	updated with Raúl García-Castro's contribution (chapters 2, 3, 5)
0.4	16.11.04	Raúl García-Castro	formatted with the style of KW deliverables
0.5	29.11.04	Raúl García-Castro	included the comments from Asunción Gómez Pérez
0.6	06.12.04	Raúl García-Castro	updated with Doug Foxvog's contribution (chapter 8)
0.7	08.12.04	Raúl García-Castro	updated with Holger Wache's contribution (chapter 7)
0.8	10.12.04	Raúl García-Castro	updated with Raúl García-Castro's contribution (chapters 1, 4, 9)
0.9	20.12.04	Raúl García-Castro	included the comments from Asunción Gómez Pérez
1.0	21.12.04	Raúl García-Castro	version 1.0 sent to Quality Assessor (Holger Wache)
1.1	03.01.05	Raúl García-Castro	included the comments from Rosario Plaza
1.2	10.01.05	Diana Maynard	included the comments from Diana Maynard
1.3	14.01.05	Raúl García-Castro	included the comments from Diana Maynard and Holger Wache, and sent to Quality Controller (Denny Vrandecic)
1.4	29.01.05	Raúl García-Castro	included the comments from Denny Vrandecic, and sent to Quality Assurance Coordinator (Francisco Martin-Recuerda)
1.5	08.02.05	Raúl García-Castro	included the comments from Francisco Martin-Recuerda

Executive Summary

Ontology technology, as any other technology, must be thoroughly assessed for its consolidation both in the academic and in the industrial world. In Knowledge Web, this assessment will be performed by means of benchmarking activities that can achieve a continuous improvement of the tools and extract the set of best practices carried out in them, thus obtaining better results than in ordinary evaluations.

This deliverable presents the benchmarking framework that will serve as the grounding for Knowledge Web benchmarking activities. This framework provides Knowledge Web partners with a methodology for benchmarking ontology tools that is conceived as a continuous process with three main phases: *Plan*, *Experiment*, and *Improve*.

In the benchmarking methodology proposed, and in any other ontology tool evaluation, it is quite convenient to use benchmark suites for experimentation. These benchmark suites allow reducing the cost of the experimentation and the standardisation of the results obtained by different groups of people. Therefore, this deliverable presents some guidelines and criteria for developing a benchmark suite. These guidelines can also be used to assess the quality of other already existing benchmark suites.

In order to provide a first overview of Knowledge Web's future benchmarking activities, this deliverable includes information about several kinds of tools that can help when performing these activities, and introduces some considerations with regard to the benchmarking of the different ontology tools that will be taken into account in workpackage 2.1: ontology development tools, ontology-based annotation tools, ontology-based reasoning tools, and semantic web service technology.

Contents

Chapter 1

Introduction

by RAÚL GARCÍA-CASTRO

To improve the quality of an ontology tool different kinds of evaluations over the tool can be performed. These evaluations allow developers to obtain a tool with less errors, to know the tool's weaknesses, to know how the tool performs certain functionalities, or even to compare its performance with any other tools.

In Knowledge Web, instead of performing ontology tool evaluations, ontology tool benchmarking is going to be performed. Besides providing all the benefits that can be obtained from an evaluation, benchmarking also provides a continuous improvement of the quality of the tools and learning from the best practices of other people when developing the 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.

The role of this deliverable is to provide a framework that supports the different benchmarking activities to be performed in workpackage 2.1. Although this deliverable is focused on workpackage 2.1, certain chapters deal with general issues so they can be used by other Knowledge Web partners in the benchmarking activities of other workpackages or even by Knowledge Web's industrial partners.

In the Knowledge Web deliverable 2.1.1, an overview of benchmarking and its main related areas was presented: software experimentation and measurement. Practitioners from these areas have developed through the years different methodologies to carry out these tasks and, grounding in these methodologies, Chapter 2 proposes a new methodology for software benchmarking, specifically for ontology tools.

This methodology proposes a continuous process where each benchmarking iteration comprises three phases: a *Plan* phase for planning the benchmarking, an *Experiment* phase for obtaining data from the tools, and an *Improve* phase for improving the tools. These phases contain the main tasks to perform when benchmarking ontology tools, and specific recommendations for Knowledge Web's benchmarking activities.

To support the benchmarking activities, Chapter 3 also gives some recommendations for developing benchmark suites for ontology tools, as well as the desirable properties that these benchmark suites should have. Furthermore, Chapter 4 describes a number of tools that can help in the different phases of the benchmarking process.

Finally, there is a chapter devoted to each type of tool that will be benchmarked in workpackage 2.1: ontology development tools (Chapter 5), ontology-based annotation tools (Chapter 6), ontology-based reasoning tools (Chapter 7), and semantic web service technology (Chapter 8). These chapters introduce the future benchmarking activities in which these tools will be involved, presenting the different tools that are candidates to be benchmarked in workpackage 2.1, the general evaluation criteria that can be used with these tools according to workpackage 2.1 topics (scalability, robustness, and interoperability), the different benchmark suites that can be used to measure the tools according to these criteria, and the supporting tools that can be used in each specific benchmarking.

Chapter 2

Benchmarking methodology

by RAÚL GARCÍA-CASTRO

This chapter presents a benchmarking methodology that provides a set of guidelines to follow in the benchmarking activities to be performed in Knowledge Web. This methodology has been developed from methodologies from different areas such as business community benchmarking, software experimentation, and software measurement. A sample of these methodologies can be found in Knowledge Web's deliverable 2.1.1 [Wache *et al.*, 2004].

The benchmarking methodology includes the main phases to carry out when benchmarking ontology tools with a definition of the tasks to perform in each of them, considering its inputs, outputs and actors. Therefore, the methodology can be used in a twofold manner: it can be used to assist in carrying out benchmarking activities or it can be used to know, at a certain point of time, which is the actual progress of a benchmarking activity.

This methodology considers that the benchmarking activity is focused on software products, specifically ontology tools. Benchmarking can also deal with methods or processes, but we have limited the scope of the methodology for a better understanding and because benchmarking studies in Knowledge Web will be performed over ontology tools or methods implemented by ontology tools.

In order to cover all the benchmarking activities present in Knowledge Web and to make them useful to other people from outside the Network of Excellence (for instance, industrial partners), this methodology is intended to be as general as possible. Nevertheless, specific comments and recommendations regarding Knowledge Web benchmarking activities have been included preceded by: ***KW***, and framed in a grey rectangle.

2.1 Benchmarking Actors

The tasks of the benchmarking process are carried out by different actors according to the kind of roles that must be performed in each task. In this section we present the different kind of actors involved in the benchmarking process.

Benchmarking initiator The benchmarking initiator is the member (or members) of an organisation who performs the first tasks of the benchmarking process. His work consists in preparing a proposal for carrying out the benchmarking activity in the organisation and in obtaining the management approval to perform it.

Organisation management The organisation management plays a key role in the benchmarking process, as it must approve the benchmarking activity and the changes that result from it. It must also assign resources to the benchmarking and integrate the benchmarking planning into the organisation planning.

Benchmarking team Once the organisation management approves the benchmarking proposal, the benchmarking team are the members of the organisation responsible for performing most of the remaining benchmarking process.

Benchmarking partners The benchmarking partners are the organisations that participate in the benchmarking activity. All the partners must agree on the steps to carry out during the benchmarking and their needs must be taken into account.

Tool developers The developers of the tool considered for the benchmarking are the ones that will implement the necessary changes in the tool to improve it. Some of them may also be part of the benchmarking team and, in this case, care must be taken to minimise bias.

2.2 Benchmarking process

The benchmarking process defined in this methodology is a continuous process that should be performed indefinitely in order to obtain a continuous improvement in the tools.

Figure 2.1 shows the main phases of the benchmarking process, which is composed of a benchmarking iteration that is repeated forever.

Each benchmarking iteration is composed of three phases (*Plan*, *Experiment*, and *Improve*) and ends with a *Recalibration* task. The main goals of these phases are:

Plan phase. It is composed of the set of tasks that must be performed to prepare the proposal for benchmarking, to find other organisations that want to participate in the benchmarking activity, and to plan the benchmarking.

Experiment phase. It is composed of the set of tasks where the experimentation over the different tools considered in the benchmarking activity is performed.

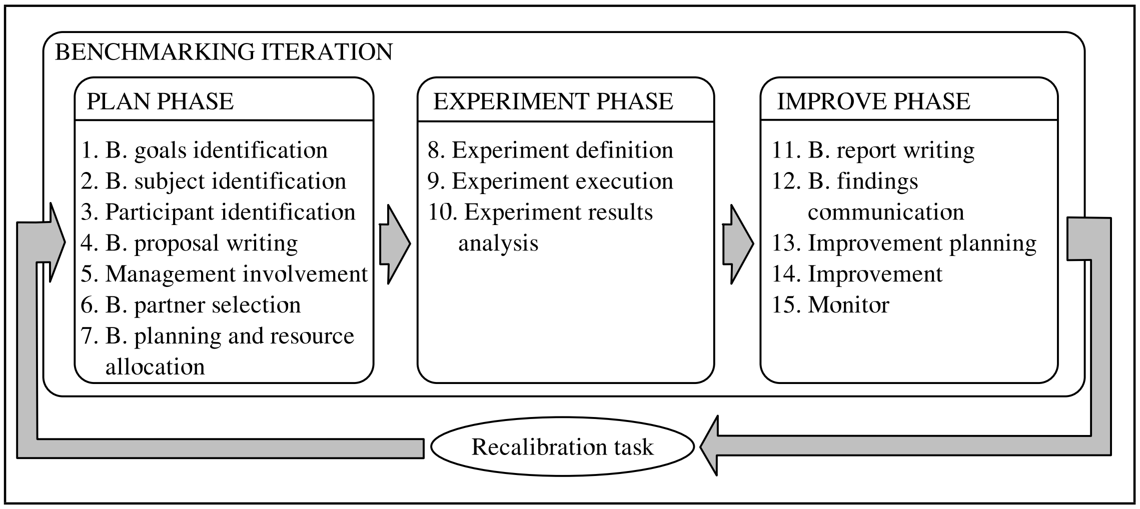


Figure 2.1: The benchmarking process

Improve phase. It is composed of the set of tasks where the results of the benchmarking process are produced and communicated to the benchmarking partners, and where the improvement of the different tools is performed in several improvement cycles.

These three phases of the benchmarking process are described in the following sections, where we provide a definition of the tasks that compose them, the actors that perform these tasks, its inputs, and its outputs.

While the three phases mentioned before are devoted to the tool improvement, the goal of the *Recalibration* task is to improve the benchmarking process itself. This task is described at the end of the chapter.

2.3 Plan phase

The *Plan* phase is composed of the set of tasks that must be performed to prepare the proposal for benchmarking, to find other organisations that want to participate in the benchmarking activity, and to plan the benchmarking. These tasks and its interdependencies are shown in Figure 2.2, and are the following:

- Benchmarking goals identification
- Benchmarking subject identification
- Participant identification
- Benchmarking proposal writing

- Management involvement
- Benchmarking partner selection
- Benchmarking planning and resource allocation

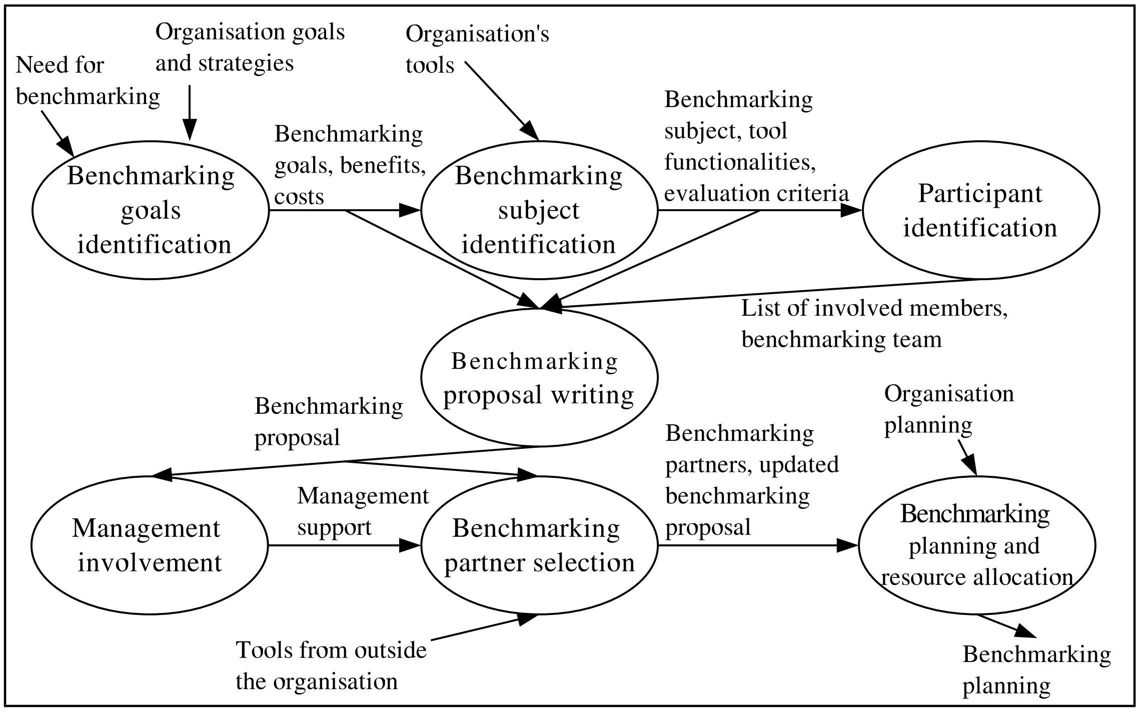

Need for benchmarking
Organisation goals and strategies
Organisation's tools
Benchmarking goals identification
Benchmarking goals, benefits, costs
Benchmarking subject identification
Benchmarking subject, tool functionalities, evaluation criteria
Participant identification
List of involved members, benchmarking team
Benchmarking proposal writing
Benchmarking proposal
Organisation planning
Benchmarking partners, updated benchmarking proposal
Management involvement
Management support
Benchmarking partner selection
Benchmarking planning and resource allocation
Tools from outside the organisation
Benchmarking planning

Figure 2.2: Plan phase of the benchmarking process

2.3.1 Benchmarking goals identification

Actors	
Benchmarking initiator	
Inputs	**Outputs**
Need for benchmarking Organisation goals and strategies	Benchmarking goals Benchmarking benefits Benchmarking costs

The benchmarking process starts in an organisation with one of its members' awareness of the need for benchmarking. This need for benchmarking varies from one organisation to another and is highly related to the desired goals of the benchmarking process.

The main needs/goals when performing benchmarking in an organisation are the following [Sole and Bist, 1995, Wireman, 2003, Kraft, 1997]:

- To assess the performance of the organisation's products and processes over time.
- To improve the quality of the organisation's products and processes.
- To compare the organisation's products and processes against those of the best organisations and to close the performance gap.
- To obtain a deep understanding of the practices that create superior products and processes.
- To establish standards set by the industry or the best organisations, or to create them after analyzing the best organisations.
- To increase the customer's satisfaction over the organisation products.

The members of the organisation who become aware of the need for benchmarking take the role of benchmarking initiators. The benchmarking initiator will usually be just one member of the organisation and the one who carries out the first tasks of the benchmarking process.

During this task, the benchmarking initiator must identify the benchmarking goals according to the organisation goals and strategies. He must also identify the benefits that the benchmarking process will bring the organisation, and the costs of performing benchmarking.

KW In Knowledge Web, the benchmarking initiator/s should be a member of a partner organisation, and the person in charge of the benchmarking activity in the corresponding workpackage.

The global goal of the benchmarking study must be related to the goals of the workpackage where the benchmarking activity will be performed, and will depend on the Knowledge Web area where the activity is being developed.

In the industry area, the goal will be oriented to give recommendations of ontology tools to the industrial community with respect to their utility, while in the research area it will be oriented to improve these tools and to help users to evaluate their suitability according to the relevant topics of this area: scalability, heterogeneity, etc.

2.3.2 Benchmarking subject identification

Actors	
Benchmarking initiator	
Inputs	**Outputs**
Benchmarking goals Benchmarking benefits Benchmarking costs Organisation's tools	Benchmarking subject Tool's relevant functionalities Evaluation criteria

The goal of this task is to identify which of the tools developed in the organisation will be benchmarked. This includes identifying the tool functionalities relevant for the study and the evaluation criteria that will be used to assess these functionalities. The benchmarking subject, functionalities, and criteria should be those whose improvement would significantly benefit the organisation.

In this task, the benchmarking initiator must perform an analysis of the tools developed in the organisation in order to understand and document them. This analysis must provide a description of the tools, along with their weaknesses and functionalities that need improvement.

Then, the benchmarking initiator must select which of these tools will be the benchmarking subject and which functionalities and criteria will be considered according to:

- The analysis of the tools developed in the organisation.
- The benchmarking goals, benefits, and costs identified in the previous task.
- Other factors considered critical in the organisation such as quality requirements, end user needs, etc.

KW In Knowledge Web this task is quite straightforward, as the benchmarking initiator will usually belong to a Knowledge Web partner that develops one of the tools to be considered in the benchmarking study.

The functionalities and criteria to take into account must be related to the issues of the workpackage where the benchmarking activity is being performed.

2.3.3 Participant identification

Actors	
Benchmarking initiator	
Inputs	**Outputs**
Benchmarking subject Tool's relevant functionalities Evaluation criteria	List of involved members Benchmarking team

In this task, the benchmarking initiator must identify and contact the members of the organisation that are involved with the selected tool and functionalities. This group of people can be composed of managers, developers, ontology engineers, end users, etc. Other relevant participants from outside the organisation, such as customers or consultants, can also be included in this group.

The benchmarking initiator must also select the components of the benchmarking team. This team will perform most of the remaining benchmarking tasks.

The benchmarking team must be composed of the organisation members whose work and interest are related to the kind of tools that will be benchmarked. They should have an understanding of these tools and experience of working with them.

The benchmarking team should be small, and its members must be aware of the time that they will spend in the benchmarking activity, which will be considerable. Usually, the benchmarking initiator will be a member of the benchmarking team.

The benchmarking team must have its responsibilities clearly defined and must also be trained in the different tasks to be performed in the remaining of the benchmarking process.

KW In Knowledge Web, the tool that will be benchmarked usually implements or gives support to some method or theory. The creators of these methods or theories should be identified and contacted. It would also be of high value that some of these people be part of the benchmarking team.

Usually all the members of the benchmarking team will belong to the organisation. In some cases, members from other Knowledge Web partners can also be part of the benchmarking team.

As the assigned effort of Knowledge Web partners in workpackages is limited, the benchmarking teams in Knowledge Web's benchmarking activities will be composed of few members. Care must be taken when choosing these members in order to minimise their training in the tasks to perform and in the ontology tool.

2.3.4 Benchmarking proposal writing

Actors	
Benchmarking initiator Benchmarking team	
Inputs	**Outputs**
Benchmarking goals Benchmarking benefits Benchmarking costs Benchmarking subject Tool's relevant functionalities Evaluation criteria List of involved members Benchmarking team	Benchmarking proposal

In this task, the benchmarking team (and the benchmarking initiator if he is not part of the team) must write a document with the benchmarking proposal. This proposal will be used as a reference along the benchmarking process and should include all the relevant information about the process.

The benchmarking proposal must include all the information identified in the previous benchmarking tasks with an approximate description of the benchmarking process, and a more detailed description of the benchmarking costs with an estimation of the resources needed in the benchmarking process: people, equipment, travel, etc.

When writing the benchmarking proposal, the benchmarking team must consider that it will have different intended readers: organisation management, organisation developers, members of partner organisations, and the benchmarking team themselves. The proposal must be useful and understandable by all of them and must include:

- The description of the benchmarking process.
- The benchmarking goals.
- The benchmarking benefits.
- The benchmarking costs.
- The benchmarking subject.
- The tool's relevant functionalities.
- The evaluation criteria for these functionalities.
- The list of members involved.

- The members of the benchmarking team.
- The resources needed in the benchmarking process.

> *KW* Members from other Knowledge Web partners can contribute towards writing the benchmarking proposal.
>
> The benchmarking proposal must be accessible by all Knowledge Web partners, and will be part of the deliverable that compiles the information on the benchmarking activity.

2.3.5 Management involvement

Actors	
Benchmarking initiator Organisation management	
Inputs	**Outputs**
Benchmarking proposal	Management support

In this task, the benchmarking initiator must carry out the benchmarking proposal to the organisation management. The benchmarking initiator must inform the organisation management about the benchmarking process, its goals, its benefits, and its costs. He/she must also inform about which components of the organisation are, and will be, involved in the process: which tool will be benchmarked, who are the organisation members involved in the benchmarking process, who will be part of the benchmarking team, etc.

This task is of great importance because the management approval is needed to continue with the benchmarking process. The organisation management must commit enough resources for carrying out the benchmarking. The management support will also be needed in the future when implementing changes based on the benchmarking, either in the tool or in the organisation processes that affect the tool.

> *KW* The benchmarking proposal must be communicated to the members of the workpackage where the benchmarking activity is being carried out as well as to the organisation management. These two groups must approve the proposal in order to continue with the benchmarking.

2.3.6 Benchmarking partner selection

Actors	
Benchmarking team Benchmarking partners	
Inputs	**Outputs**
Benchmarking proposal Management support Tools developed outside the organisation	Benchmarking partners Updated benchmarking proposal

Once the benchmarking activity has the organisation management support, the benchmarking team must select the different tools to be considered in the benchmarking study.

First of all, the benchmarking team must research and identify the tools that are comparable to the tool that has been chosen as the benchmarking subject. The benchmarking team must collect and analyse information about these tools and about the organisations that develop them.

Then, the benchmarking team must define the criteria to identify which of the tools are suitable for benchmarking. According to these defined criteria, the benchmarking team must select from the comparable tools those that will be considered in the benchmarking study.

These criteria can be: the tool is relevant in the community or in the industry, the tool uses the latest technological tendencies, the tool is widely used, the tool is publicly available, etc. In order to obtain better results with the benchmarking activity, the chosen tools should be those that are considered the best.

When the benchmarking team has selected the tools that will be part of the benchmarking study, they must make contact with someone from the organisations that develop each of these tools to see if these organisations are interested in benchmarking.

If the organisation that develops a tool is not interested in the benchmarking activity, the benchmarking team can make contact with other organisations to see if they want to participate assessing that tool, although not being the tool developers.

The organisations willing to participate in the benchmarking activity become benchmarking partners. These benchmarking partners will also have to establish a benchmarking team and to carry out the benchmarking proposal to their own organisation management for approval.

During the course of this task, the benchmarking proposal will be modified to include the partner's opinions and needs. This will result in an updated benchmarking proposal that will be the one used in the remaining of the benchmarking tasks. Depending on the magnitude of the modifications, the proposal should be presented again to each partner organisation management for approval.

KW The benchmarking proposal must be communicated to all Knowledge Web partners, along with a call for participation. All the partners interested should participate in the benchmarking activity, including those that develop tools as the ones that will be benchmarked.

The benchmarking partner organisations do not have to belong to the Knowledge Web, as there are some 'best in class' tools whose developer organisations are outside the Network of Excellence and it may not be possible to benchmark these tools without their support.

2.3.7 Benchmarking planning and resource allocation

Actors	
Benchmarking teams Management of the organisations	
Inputs	**Outputs**
Benchmarking partners Updated benchmarking proposal Organisation planning	Benchmarking planning

In this task, the benchmarking teams and the managers from each partner organisation must define the planning of the rest of the benchmarking process and must reach a consensus on it. This planning must be considered and integrated into each organisation planning.

The benchmarking planning must consider the effort that will be spent in the benchmarking activities and what organisation resources will be devoted to them: people, computers, travel, etc.

KW In Knowledge Web the benchmarking planning must be agreed by each of the benchmarking partners and by the members of the workpackage where the benchmarking activity is being developed.

2.4 Experiment phase

The *Experiment* phase is composed of the set of tasks where the experimentation of the different tools that are considered in the benchmarking activity is performed. These tasks and their interdependencies are shown in Figure 2.3, and are the following:

- Experiment definition
- Experiment execution
- Experiment results analysis

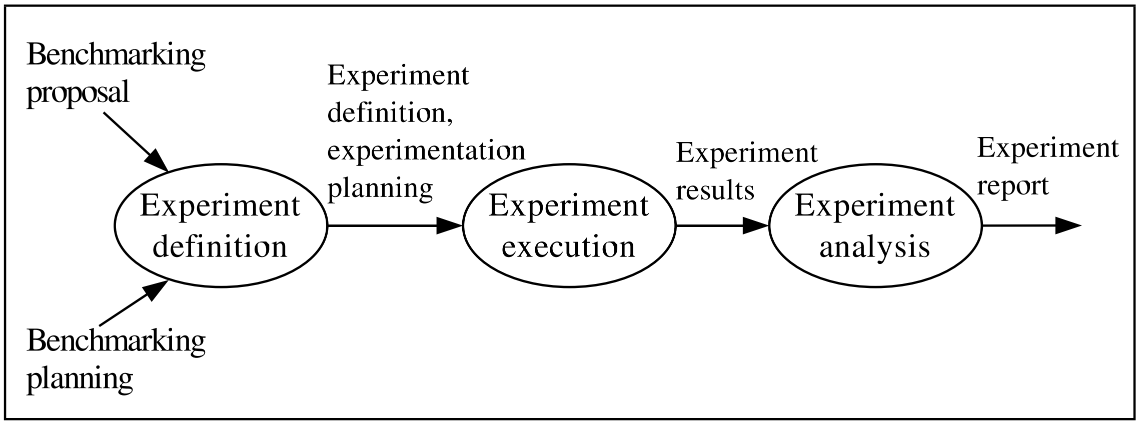


Figure 2.3: Experiment phase of the benchmarking process

2.4.1 Experiment definition

Actors	
Benchmarking teams	
Inputs	**Outputs**
Updated benchmarking proposal Benchmarking planning	Experiment definition Experimentation planning

In this task, the benchmarking teams from each partner organisation must define the experiment that will be performed in each of the tools involved in the benchmarking process.

The experiment must be defined according to the intended benchmarking goals, and must evaluate the selected functionalities of the tools according to their corresponding criteria as stated in the benchmarking proposal.

The experiment must provide objective and reliable data on the tools, not just on its performance but also on the reasons of its performance, and must be defined taking into account its future reuse.

The benchmarking teams must also define the planning that will be followed during the experimentation. The experimentation planning must be defined according to the benchmarking planning defined in the previous task.

The experiment definition and its planning must be communicated to all the partners and they must agree on it.

KW The experiments to perform in Knowledge Web must be related to the goals of the workpackage where the benchmarking activity will be performed.

2.4.2 Experiment execution

Actors	
Benchmarking teams	
Inputs	**Outputs**
Experiment definition Experimentation planning	Experiment results

According to the experimentation planning defined in the previous task, the benchmarking teams must perform the defined experiments on their tools.

The data obtained from all the experiments must be compiled, documented, and expressed in a common format in order to facilitate its future analysis.

KW The results obtained from the experiments must be available to all Knowledge Web partners.

2.4.3 Experiment results analysis

Actors	
Benchmarking teams	
Inputs	**Outputs**
Experiment results	Experiment report

In this task, the benchmarking teams must analyse the results obtained in the experiments. This analysis involves comparing the results of the experiments performed on each tool and the practices that lead to these results.

During the analysis, the benchmarking teams must identify and document any significant differences observed in the results of the tools, determining the practices that lead to these different results. The benchmarking teams must also attempt to identify if, between the practices found, there are some of them that can be considered as best practices.

When the analysis of the experimentation results ends, the benchmarking teams must write a report with all the findings obtained during the experimentation: results of the experimentation, differences in the results, practices and best practices found, etc.

KW The experiment report must be available to all Knowledge Web partners.

2.5 Improve phase

The *Improve* phase is composed of the set of tasks where the results of the benchmarking process are produced and communicated to the benchmarking partners, and the improvement of the different tools is performed in several improvement cycles. These tasks and their interdependencies are shown in Figure 2.4, and are the following:

- Benchmarking report writing
- Benchmarking findings communication
- Improvement planning
- Improvement
- Monitor

2.5.1 Benchmarking report writing

Actors	
Benchmarking teams	
Inputs	**Outputs**
Updated benchmarking proposal Experiment report	Benchmarking report

In this task, the benchmarking teams must write the report of the benchmarking activity. While the experiment report provides technical details and results on the experimentation, the benchmarking report is intended to provide an understandable summary of the benchmarking activity carried out. The benchmarking report must be written taking into account that the document will have different audiences: managers, benchmarking teams, developers, etc. from all the partner organisations.

The benchmarking report must include an explanation of the benchmarking process followed, with all the relevant information from the updated version of the benchmarking

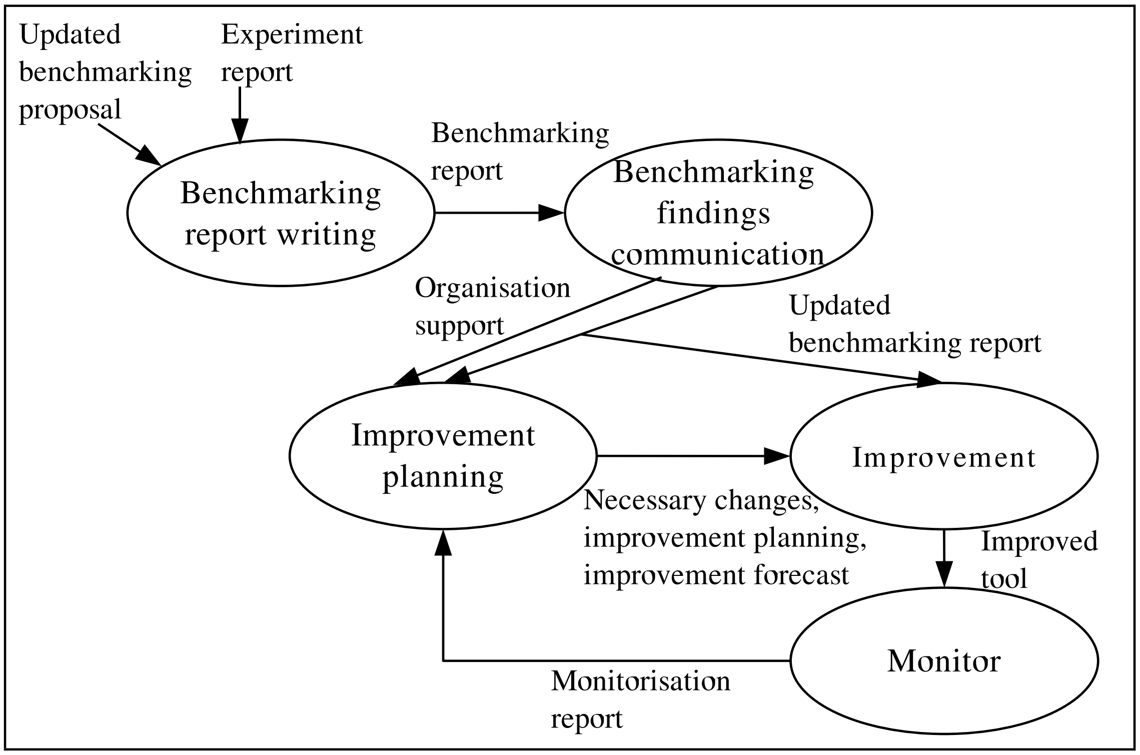

Updated benchmarking proposal
Experiment report
Benchmarking report writing
Benchmarking report
Benchmarking findings communication
Organisation support
Updated benchmarking report
Improvement planning
Improvement
Necessary changes, improvement planning, improvement forecast
Improved tool
Monitor
Monitorisation report

Figure 2.4: Improve phase of the benchmarking process

proposal and the results and conclusions of the experiments present in the experiment report, highlighting the best practices found during the experimentation and including any best practices found in the community.

The benchmarking report must also include the recommendations of the benchmarking team for improving the tools according to the experiment results, to the practices found, and to the community best practices.

The goal of the benchmarking report is not to provide a ranking of the tools, but to provide the practices and the best practices found in the benchmarking study and give improvement recommendations.

KW In Knowledge Web, the benchmarking report is the deliverable that compiles all the information regarding the benchmarking activity carried out in a workpackage.

The benchmarking report must be accessible to all Knowledge Web partners and to other people from outside the Network of Excellence, so they can take advantage of the lessons learnt when benchmarking the tools.

2.5.2 Benchmarking findings communication

Actors	
Benchmarking teams	
Inputs	**Outputs**
Benchmarking report	Updated benchmarking report Organisation support

In this task, the benchmarking teams must communicate the results of the benchmarking study to their organisations, and particularly to all the members involved and identified when planning the benchmarking activity.

This communication should be in the form of meetings that imply one or more partner organisations. The goals of a benchmarking team in these meetings are twofold:

- To obtain feedback from the involved members about the benchmarking process, results, and improvement recommendations.
- To obtain support and commitment from the organisation members for implementing the improvement recommendations in the tool.

Any feedback received during these findings communications must be collected, documented, and analysed. This analysis can result in having to review the work done and to update the benchmarking report.

> ***KW*** In Knowledge Web, the communication of the benchmarking findings must be performed by the distribution of the deliverable and by presentations of the benchmarking activity carried out in different meetings and events.

2.5.3 Improvement planning

Actors	
Benchmarking teams Management of the organisations	
Inputs	**Outputs**
Updated benchmarking report Monitoring report Organisation support	Necessary changes Improvement planning Improvement forecast

The last three tasks of the *Improve* phase (*Improvement planning*, *Improvement*, and *Monitor*) form a cycle that must be performed separately in each organisation. It is in these tasks where each organisation benefits from the results obtained in the benchmarking.

In this task, the benchmarking team and the managers from each partner organisation must identify, from the benchmarking report and the monitoring reports, which are the necessary changes to obtain an improved tool. They must also forecast which will be the improvement obtained after performing these changes.

Both the organisation management and the benchmarking team must provide the organisation with mechanisms that ensure the accomplishment of the improvements.

The benchmarking team must also provide the tool developers with mechanisms for measuring the tool functionalities. These mechanisms can be obtained from the experiments performed in the *Experiment* phase of the benchmarking activity.

Then, they must define the planning for improving the benchmarked tool and reach a consensus on it. This planning must be considered and integrated into each organisation planning.

The improvement planning must consider the time that will be spent in improvement activities and what organisation resources will be devoted to them: people, computers, travel, etc.

KW The last three tasks of the *Improve* phase (*Improvement planning*, *Improvement*, and *Monitor*) are not related to Knowledge Web in terms of work to be done and results to produce. Nevertheless, the partners that participate in the benchmarking activities should carry them out in order to obtain marked improvements in their tools.

2.5.4 Improvement

Actors	
Tool developers	
Inputs	**Outputs**
Updated benchmarking report Necessary changes Improvement planning Improvement forecast	Improved tool

In this task, according to the improvement planning, the developers of each of the tools benchmarked must implement the necessary changes in order to achieve the desired results.

Before implementing any changes, the tool developers must measure the actual state of the tool using the measurement mechanisms provided by the benchmarking team in the *Improvement planning* task. Then, after having implemented the necessary changes, the developers must measure the tool again and compare the resulting measurements with the ones obtained before implementing the changes and with the improvement forecast.

KW Knowledge Web partners that attain improvement successes should communicate them to the benchmarking partners in order to provide further feedback about the benchmarking outcomes.

2.5.5 Monitor

Actors	
Benchmarking team Tool developers	
Inputs	**Outputs**
Improved tool	Monitoring report

In each organisation, the benchmarking team must provide the tool developers with means for monitoring the organisation's tool: monitoring tools, the benchmark suites developed, etc.

The tool developers must periodically monitor the tool and write a report with the results of this monitoring.

These monitoring results can cause a need for new improvements in the tool and the beginning of a new improvement cycle, having to perform again the two previously mentioned tasks: *Improvement Planning* and *Improvement*.

2.6 Recalibration task

Actors	
Benchmarking team	
Inputs	**Outputs**
Benchmarking process Lessons learnt	Improved benchmarking process

The recalibration task is performed at the end of each benchmarking iteration. In this task, the benchmarking team must recalibrate the benchmarking process using the lessons learnt while performing it. This way, the organisation gets improvement not just in the tools, but also in the benchmarking process. This recalibration is needed because both the tools and the organisations evolve over time.

> ***KW*** The lessons learnt in the different benchmarking activities that will be performed in Knowledge Web must be collected so as to develop an improved version of this methodology.

Chapter 3

Building benchmark suites for ontology tools

by RAÚL GARCÍA-CASTRO

The use of benchmark suites is frequent when carrying out a benchmarking activity or when evaluating software systems. As in the framework of Knowledge Web some benchmarking activities have to be conducted, it is quite probable that a number of benchmark suites will be used or developed when needed. This chapter presents the different types of benchmarks that can be developed and the desirable properties that a benchmark suite should have, along with some guidelines on how to develop a benchmark suite.

3.1 Types of benchmarks

Stefani et al. [2003] described the four following types of benchmarks that can be used in the evaluation of software systems:

Application benchmarks These benchmarks use real applications and workload conditions.

Synthetic benchmarks These benchmarks emulate the functionalities of significant applications, while cutting out additional or less important features.

Kernel benchmarks These benchmarks use simple functions designed to represent key portions of real applications.

Technology-specific benchmarks These benchmarks are designed to point out the main differences of devices belonging to the same technological family.

3.2 Desirable properties for a benchmark suite

The following properties, extracted from the work of different authors [Bull *et al.*, 1999, Shirazi *et al.*, 1999, Sim *et al.*, 2003, Stefani *et al.*, 2003], will help the people involved in benchmarking activities either to develop new benchmark suites or to assess the quality of different benchmark suites before using them.

Accessibility A benchmark suite must be accessible to anyone interested in it. This includes the necessary software to execute the benchmark suite, its documentation, and its source code in order to increase transparency.

The results obtained when executing the benchmark suite should also be public, so anyone can apply the benchmark suite and compare their results with the ones available.

Affordability Using a benchmark suite entails a number of costs, commonly human, software, and hardware resources. The costs of using a benchmark suite must be lower than those of having to define, to implement, and to carry out any other experiments that fulfil the same goal.

Some ways of reducing the resources consumed in the execution of a benchmark suite are: automate the execution of the benchmark suite, provide components for data collection and analysis, or facilitate its use for different heterogeneous systems.

Simplicity The benchmark suite must be simple and interpretable. It must be documented and anyone who wants to use it must be able to understand how it works and the results that it produces. If the benchmark suite is not transparent enough, its results will be questioned and it could be interpreted incorrectly.

To make this easier, the elements of the benchmark suite should have a common structure, use, inputs, and outputs. Measurements should have the same meaning across the benchmark suite.

Representativity The actions that perform the benchmarks that compose the benchmark suite must be representative of the actions that are usually performed on the system.

Portability The benchmark suite should be executed on as wide a variety of environments as possible, and should be applicable to as much systems as possible.

It should also be specified at a high enough level of abstraction to ensure that it is portable to different tools and techniques and that it is not biased against other technologies.

Scalability The benchmark suite should be parameterised to allow to scale the benchmarks with varying input rates.

It should also scale to work with tools or techniques at different levels of maturity. It should be applicable to research prototypes and commercial products.

Robustness The benchmark suite must consider unpredictable environment behaviours, and should not be sensitive to factors not relevant to the study. When running the same benchmark suite several times on a given system under the same conditions, the results obtained should not change considerably.

Consensus The benchmark suite must be developed by experts that provide their knowledge about the domain, and are able to identify the key problems. It should also be assessed and agreed on by the whole community.

3.3 Building benchmark suites in Knowledge Web

Performing experiments with ontology tools is one of the main tasks in the different benchmarking activities that will take place in workpackage 2.1 and in other Knowledge Web workpackages (1.2 and 2.2). Using benchmark suites for experimentation in Knowledge Web provides a number of benefits such as saving the resources of developing different experiments for each tool, or facilitating the collection and analysis of experiment data.

As the benchmark suites developed in Knowledge Web will be built by experts in the different domains and will take into account different tools, they will be valuable and useful for all Knowledge Web partners and also for other people from outside the Network of Excellence.

The benchmark suites that will be built in Knowledge Web will vary from one benchmarking to another. Depending on the objectives of the experimentation some kind of benchmarks will be more useful than other. Technology-specific benchmarks are the best option for characterising the tool according to some specific domain tasks. If the interest resides in knowing the behavior of the functionalities of the tools according to specific situations, then synthetic or kernel benchmarks must be considered. Finally, application benchmarks are the choice for knowing the response of the tools to real-life cases.

These benchmark suites that will be developed in Knowledge Web should consider the desirable properties of a benchmark suite described in the previous section. Although a good benchmark suite should have most of these properties, each experiment will require that some of them are considered before others: in some cases obtaining a scalable benchmark suite can be a priority, in other cases this priority can be having a benchmark suite portable between several platforms.

It must also be considered that achieving a high degree of all these properties in a benchmark suite for ontology tools is not possible, as increasing some of them has a negative influence in others. For example, a benchmark suite able to execute in different tools and parameterised in order to accept different workloads will not be very simple or affordable, given the Knowledge Web resources allocated to this task.

Chapter 4

General supporting tools for benchmarking

by RAÚL GARCÍA-CASTRO

When people involved in benchmarking activities need to perform some kind of tasks, like monitoring or analysing data, they can either use an existing general tool or develop a new one that provides them with the specific functionalities they need. We will just consider the case of using an existing tool, although in some cases developing a specific tool is more convenient.

This chapter presents different types of tools that can be useful when performing either benchmarking activities or any other evaluation activity. Each section provides some examples of each kind of tool with references to the web pages where they can be obtained. This is neither an exhaustive listing nor does it necessarily include the most relevant tools for each kind of task. The main criteria considered were that the tools were open source or freely available.

The different types of tools considered are:

- Testing frameworks
- Workload generators
- Monitoring tools
- Statistical packages

4.1 Testing frameworks

Testing frameworks provide mechanisms for writing and running repeatable tests. The most popular testing frameworks belong to the xUnit family [Hamill, 2004], and are free and open source software. These frameworks are devoted to a certain kind of programming language and the most representative are:

JUnit[1] is implemented and used with Java and is the most widely used.

CppUnit[2] is the C++ port of JUnit.

NUnit[3] is written in C# and can be used to test any .NET language.

PyUnit[4] is the Python version of xUnit.

SUnit[5] is written in and used in the Smalltalk language. It is the original xUnit and the basis of the xUnit architecture.

Some add-on tools of the xUnit family can be applied to specialised domains like:

XMLUnit[6] supports XML testing and has versions for JUnit and NUnit.

JUnitPerf[7] supports writing code performance and scalability tests in Java.

HTTPUnit[8] tests web-based applications.

jfcUnit[9] supports testing of Java Swing based applications.

4.2 Workload generators

Workload generators are used to simulate different loads on computers, networks, or servers. Some workload generators can be used to generate different types of loads such as **JMeter**[10] that can simulate loads on a server, a network, or a Java object. Other workload generators focus on a single domain like network traffic generators (**Netperf**[11]), web server load generators (**Hammerhead 2**[12]), or database load generators (**dbMonster**[13]).

[1] http://www.junit.org/
[2] http://cppunit.sourceforge.net/
[3] http://www.nunit.org/
[4] http://pyunit.sourceforge.net/
[5] http://sunit.sourceforge.net/
[6] http://xmlunit.sourceforge.net/
[7] http://www.clarkware.com/software/JUnitPerf.html
[8] http://httpunit.sourceforge.net/
[9] http://jfcunit.sourceforge.net/
[10] http://jakarta.apache.org/jmeter/
[11] http://www.netperf.org/
[12] http://hammerhead.sourceforge.net/
[13] http://dbmonster.kernelpanic.pl/

4.3 Monitoring tools

Monitoring tools can be used to monitor different parts of the environment where a system is running.

Some tools monitor the computer where the software is running on. These tools can be those provided by the operating system or more complex ones such as **Valgrind**[14] that can monitor the computer memory, CPU caches, or multithreads.

Other tools can help to track the performance and generate statistics of other software programs. These tools are also called profilers and two examples of them are **JRat**[15] and the **Extensible Java Profiler**[16].

There are also tools for monitoring network performance. Two examples of these tools are **Argus**[17] that tracks and reports on the status and performance of all network transactions, and **Netlog**[18] which is a C library that can be linked to an existing network application to provide some instrumentation of network performance.

4.4 Statistical packages

Analysing the results of experiments usually involves having to perform some kind of statistical analysis over these results. Sometimes this analysis is quite simple and there is no need of any tool that supports it (or it can be done with other tools that provide light statistical facilities). But when the statistical analysis that must be performed is complex, statistical packages become of great importance.

Some of these packages are:

R[19] is a highly extensible language and environment for statistical computing and graphics that provides a wide variety of statistical and graphical techniques.

PSPP[20] interprets commands in the SPSS language and produces tabular output in different formats.

MacAnova[21] is an interactive, programmable, multi-platform system for matrix manipulation and statistical analysis.

[14]http://valgrind.kde.org/
[15]http://jrat.sourceforge.net/
[16]http://ejp.sourceforge.net/
[17]http://www.qosient.com/argus/
[18]http://dast.nlanr.net/Projects/Netlog/
[19]http://www.r-project.org/
[20]http://www.gnu.org/software/pspp/
[21]http://www.stat.umn.edu/macanova/

OpenStat[22] is a general-purpose statistics package that contains procedures for univariate and multivariate statistical analysis, nonparametric statistics, statistical process control, and simulation as well as data manipulation tools.

[22]http://www.statpages.org/miller/openstat/

Chapter 5

Benchmarking ontology development tools and tool suites

by RAÚL GARCÍA-CASTRO

This chapter presents part of the framework for benchmarking ontology development tools, in order to provide a first overview of the future benchmarking activities that will involve these tools in Knowledge Web. It enumerates the most relevant tools that will be considered as possible benchmarking subjects. This chapter also presents the general evaluation criteria, benchmark suites, and supporting tools that can be used when benchmarking ontology development tools according to the topics of interest in Knowledge Web workpackage 2.1 (scalability, robustness, and interoperability).

5.1 Candidate tools

This section presents a list with the ontology development tools and tool suites candidate to be benchmarked. The descriptions of these tools have been extracted from [OntoWeb, 2002] and from [Gómez-Pérez *et al.*, 2003]. The tools to be analysed when benchmarking ontology development tools in Knowledge Web will be selected from this list. The reasons for choosing these tools have been:

- The tool is of great relevance in the community or in the industry.
- The tool uses the last technological tendencies.
- The tool is widely used.
- The tool is publicly available.
- The tool has its source code available.

The tools proposed for consideration in the Knowledge Web benchmarking activities are:

- KAON
- OilEd
- OntoEdit
- Ontolingua Server
- OntoSaurus
- Protégé-2000
- WebODE
- WebOnto

5.1.1 KAON

The KAON tool suite[1] [Maedche *et al.*, 2003] is an open source extensible ontology engineering environment. The core of this tool suite is the ontology API, which defines its underlying knowledge model based on an extension of RDF(S). The OI-modeler is the ontology editor of the tool suite that provides capabilities for ontology evolution, ontology mapping, ontology generation from databases, etc.

5.1.2 OilEd

OilEd[2] [Bechhofer *et al.*, 2001] is a graphical ontology editor developed by the University of Manchester that allows the user to build ontologies using DAML+OIL.

The knowledge model of OiLEd is based on that of DAML+OIL, although it is extended by the use of a frame-like presentation for modelling. Thus OiLEd offers a familiar frame-like paradigm for modelling while still supporting the rich expressiveness of DAML+OIL where required.

OilEd provides consistency checking functions and automatic concept classifications by means of the FaCT inference engine [Horrocks *et al.*, 1999], though other DL inference engines such as RACER can also be used.

Although OilEd is primarily intended as an editor for OIL ontologies, the tool will export ontologies to a number of formats. These include OIL Standard (the human-readable presentation format for OIL), OIL-RDFS (OIL's standard RDFS serialisation),

[1]http://kaon.semanticweb.org/
[2]http://oiled.man.ac.uk

DAML+OIL (also RDFS), HTML (for viewing the ontology without the tool), the SHIQ and SHOQ(D) languages used by the FaCT reasoner, the DIG language, the OWL web ontology language, and class hierarchies generated by the classifier that can be exported as graphs for viewing with AT&T's Dotty[3] application.

5.1.3 OntoEdit

OntoEdit[4] [Sure *et al.*, 2002] is an Ontology Engineering Environment supporting the development and maintenance of ontologies by using graphical means. OntoEdit is built on top of a powerful internal ontology model. This paradigm supports representation-language neutral modelling as much as possible for concepts, relations and axioms. Several graphical views of the structures contained in the ontology support modelling the different phases of the ontology engineering cycle.

The tool is based on a flexible plug-in framework. This easily allows to extend functionality in a modularised way. The plug-in interface is open to third parties, which enables users to extend OntoEdit easily by adding new functionalities. Then, having a set of plug-ins available like a domain lexicon, an inferencing plug-in and several export and import plug-ins to and from different formats (XML, FLogic, RDF(S), and DAML+OIL) allow user-friendly customisation to adapt the tool to different usage scenarios.

5.1.4 Ontolingua Server

The Ontolingua Server[5] [Farquhar *et al.*, 1997] is a set of tools and services that support the building of shared ontologies between distributed groups, and has been developed by the Knowledge Systems Laboratory (KSL) at Stanford University. The ontology server architecture provides access to a library of ontologies, translators to languages (Prolog, CORBA IDL, CLIPS, Loom, etc.) and an editor to create and browse ontologies. Remote editors can browse and edit ontologies, and remote or local applications can access any of the ontologies in the ontology library using the OKBC (Open Knowledge Based Connectivity) protocol.

5.1.5 OntoSaurus

Ontosaurus [6] [Swartout *et al.*, 1997] has been developed by the Information Sciences Institute (ISI) at the University of South California. It consists of two modules: an ontology server, which uses Loom as its knowledge representation system, and an ontology browser

[3]http://www.research.att.com/sw/tools/graphviz/

[4]http://www.ontoprise.de/products/ontoedit_en/

[5]http://ontolingua.stanford.edu/

[6]http://www.isi.edu/isd/ontosaurus.html

server that dynamically creates html pages (including image and textual documentation) to display the ontology hierarchy. The ontology can be edited by HTML forms, and there are tools for translating LOOM into Ontolingua, KIF, KRSS, and C++.

5.1.6 Protégé-2000

Protégé-2000[7] [Noy *et al.*, 2000] is the latest tool in an established line of tools developed at Stanford University for knowledge acquisition and is freely available for download under the Mozilla open-source license.

The knowledge model of Protégé-2000 is OKBC-compatible. It provides a graphical and interactive ontology-design and knowledge-base development environment, a database backend to store and query the data, and a caching mechanism to enable the loading of large knowledge bases.

One of the major advantages of the Protégé-2000 architecture is that the system is constructed in an open, modular fashion. Its component-based architecture enables system builders to add new functionalities by creating appropriate plug-ins. The Protégé Plug-in Library[8] contains contributions from developers all over the world.

Most plug-ins fall into one of the three categories: (1) backends that enable users to store and import knowledge bases in various formats (RDF(S), XML, OIL, DAML+OIL, OWL, etc.)); (2) slot widgets, which are used to display and edit slot values or their combination in domain-specific and task-specific ways, and (3) tab plug-ins, which are knowledge-based applications usually tightly linked with Protégé knowledge bases (visualisation, ontology merging, version management, inferencing, etc.).

5.1.7 WebODE

WebODE[9] [Arpírez *et al.*, 2003] is an ontological engineering workbench that provides varied ontology related services and covers and gives support to most of the activities involved in the ontology development process and in the ontology usage. It is built on an application server basis, which provides high extensibility and usability by allowing the easy addition of new services and the use of existing ones.

WebODE ontologies are represented with a very expressive knowledge model, based on the reference set of intermediate representations of the METHONTOLOGY methodology [Fernández-López *et al.*, 1999], and are stored in a relational database. Moreover, WebODE provides a well-defined service-oriented API for ontology access that makes easy the integration with other systems. Ontologies built with WebODE can be easily

[7]http://protege.stanford.edu/
[8]http://protege.stanford.edu/plug-ins.html
[9]http://webode.dia.fi.upm.es/

integrated with other systems by using its automatic exportation and importation services from varied ontology specification languages and systems (XML, RDF(S), OIL, DAML+OIL, OWL, CARIN, FLogic, Jess, Prolog).

Ontology edition in the WebODE ontology editor is aided by form based and graphical user interfaces, a user-defined-views manager, a consistency checker, an inference engine, an axiom builder and the documentation service. WebODE also provides an inference service that has been developed in Ciao Prolog. A subset of the OKBC primitives has been defined in Prolog for their use in this inference engine. Additionally, the WebODE Axiom Builder transforms first-order logic axioms and rules into Prolog, if possible, so that they can be used in it as well.

5.1.8 WebOnto

WebOnto[10] [Domingue, 1998] is a tool developed by the Knowledge Media Institute (KMi) of the Open University. It supports the collaborative browsing, creation and editing of ontologies, which are represented in the knowledge modelling language OCML.

Its main features are: management of ontologies using a graphical interface; the automatic generation of instance editing forms from class definitions, support for PSMs and tasks modelling; inspection of elements, taking into account the inheritance of properties and consistency checking; a full tell&ask interface, and support for collaborative work, by means of broadcast/receive and making annotations (using Tadzebao).

The WebOnto server is a freely available service provided to the ontology engineering community. A library with over 100 ontologies is accessible through WebOnto and can be browsed with no restrictions on access.

5.2 General evaluation criteria

This section presents the different functionalities that ontology development tools and tool suites have, along with the general criteria that can be used when evaluating or benchmarking these functionalities. The criteria presented are related to workpackage 2.1 topics (scalability, robustness, and interoperability).

5.2.1 Scalability

The general scalability evaluation criteria regarding ontology development tools are related to the ability of these tools to deal with different types of workloads which can be present in variable quantities. Workloads present in large amounts may significantly af-

[10]http://webonto.open.ac.uk

fect the operation of an ontology development tool. The workloads that may affect an ontology development tool can be of four different types:

Access load This workload is composed of the access requests that the tool receives or generates. The petitions that a tool receives can be originated either by users or by other systems. The petitions that a tool generates can be addressed to the operating system where it resides or to another system (either local or remote).

Tool load This workload is composed of the information that the tools store in them. Ontology development tools can store different internal information such as: ontologies, user data, permission data, versioning data, etc.

Computer load This workload corresponds to the load of the computer where the tool is running. This workload is composed of different types of loads: operating system load, CPU load, memory load, etc.

Network load This workload corresponds to the load of the network where the tool is located. This workload is composed of the different network traffic present in the network and can be identified by protocol or by application.

In order to evaluate the scalability of ontology development tools, experiments must be performed to measure the behaviour of the functionalities of these tools when facing high workloads. Different evaluation criteria can be used depending on the functionality under evaluation:

- In the case of ontology management functionalities (creation, import, export, browse, edition, population, etc.), some criteria could be the response time of the tool when importing or exporting large ontologies or their instances, or the capability of graphically editing large ontologies with the tool.
- In the case of collaborative working with the tool, a criterion could be the number of users that can work concurrently with the tool.
- In the case of tool interoperation, some criteria could be the size of the ontologies that can be exchanged between two tools, or the time spent in this exchange.
- There can also be other scalability evaluation criteria not related to the functionalities of the tools like the cost of deploying a certain tool in one organisation, where installing from *n* to *2n* tools doubles computer resources, maintenance tasks, etc. but it does not necessarily double the profit obtained through the tool.

The benchmarking activities that will be performed in workpackage 2.1 will assess the scalability of ontology development tools. This scalability will be measured in terms of the response time of the tools when performing ontology management tasks (insert, update, remove, and query ontology data; import and export ontologies; etc.). This measurement will be carried out with the help of benchmark suites, either adapting some of the existing benchmark suites or developing new ones.

5.2.2 Robustness

The general robustness evaluation criteria regarding ontology development tools are related to the ability of these tools to function correctly in the presence of invalid inputs or stressful environmental conditions.

Invalid inputs can arrive to the tool from several sources. The main input source comes from human user interaction and, regarding robustness, multiuser interaction is a key issue to consider. Other computer systems, remote services, or the modules attached to the tool can also provide invalid inputs that should be taken into account. Importing ontologies is one of the main ontology management functionalities. Therefore, it is also relevant for ontology development tools to be able to deal with incorrect import ontologies or instances.

In order to evaluate the robustness of ontology development tools regarding invalid inputs, experiments must be performed in order to measure the behavior of these tools' functionalities when facing invalid inputs. Some evaluation criteria that can be used in these experiments are the ratios between invalid inputs and system errors, between invalid inputs and invalid inputs detected by the tool, or between invalid inputs and invalid inputs corrected by the tool.

Stressful environmental conditions can be present in any of the workload environments defined in the scalability criteria section. The stressing workloads can come from the different users' accesses, from the same tool, from the computer where the tool is running, or from the network where it is located.

In order to evaluate the robustness of ontology development tools regarding stressful environmental conditions, the experiments that must be performed are similar to those required for measuring the scalability of the tools, but are focused on stressing these tools with high workloads and checking the workload-error ratios.

The benchmarking activities that will be performed in workpackage 2.1 will assess the robustness of ontology development tools. This robustness will be measured in terms of the ability of the tools to handle invalid inputs and stressful environmental conditions. This measurement will be carried out with the help of benchmark suites, either adapting some of the existing benchmark suites or developing new ones.

5.2.3 Interoperability

The general interoperability evaluation criteria regarding ontology development tools are related to the ability of these tools to exchange information (ontologies, instances, user permissions, etc.) and to use the information that has been exchanged.

Two ontology development tools can exchange information either by communicating between themselves or by means of a common resource (a RDF(S) or OWL file, a DBMS, etc.). For performing a direct information exchange the tools must have communication modules to deal with it. Also, a tool can provide other tools with programming interfaces or services for accessing the information present in the tool. For exchanging information through a common resource, the tools must be able to import and export information from and to the certain resource.

In order to evaluate the interoperability of ontology development tools, experiments must be performed in order to measure the ability of these tools to exchange information and to use the information that has been exchanged. Different evaluation criteria can be used depending on how the exchange is performed:

- In the case of direct information exchange, some criteria could be the ratio of a tool's functionalities that are accessible to other tools, the compatibility between two tools' communication modules, or the ratio of information that can be accessed from other tools.

- In the case of information exchange using a common resource, some criteria could be the ability of the tool to export data to the common format, the ability of other tools to import the data exported by the tool, or the tool's ability to import data in the common format.

- A criteria that can be used in both cases is the amount of knowledge lost when exchanging information between the tools, or between a tool and the common resource.

The benchmarking activities that will be performed in workpackage 2.1 will assess the interoperability of ontology development tools. This interoperability will be measured in terms of the ability of the tools to import and export data from and to two of the most widely used ontology languages: RDF(S) and OWL. Other criteria will be the loss and transformation of knowledge between exports and imports. This measurement will be carried out with the help of benchmark suites, either adapting some of the existing benchmark suites or developing new ones.

5.3 Benchmark suites

This section presents a list of the existing benchmark suites that can be used when evaluating or when benchmarking ontology development tools and tool suites. The benchmark suites presented are related to WP 2.1 topics (scalability, robustness, and interoperability).

5.3.1 RDF and OWL Test Cases

In the scope of the W3C, the RDF Test Cases[11] [Grant and Beckett, 2004] and the OWL Test Cases[12] [Carroll and Roo, 2004] were created by the W3C RDF Core Working Group and the W3C Web Ontology Working Group, respectively. These tests check the correct usage of the tools that implement RDF and OWL knowledge bases and illustrate the resolution of different issues considered by the Working Groups.

The RDF and OWL Test Cases are intended to provide examples for, and clarification of, the normative definition of the languages, and also to be suitable for use by developers in test harnesses, possibly as part of a test driven development process. Therefore, these tests can be used to evaluate the robustness of ontology development tools that manage RDF and OWL ontologies and RDF instances.

The tests, which can be downloaded from the respective web pages, consist of a machine-readable description of the test (in order to assist in machine-processing the tests), one (or more) input documents in the corresponding language, and the intended result.

5.3.2 Lehigh University benchmark

The Lehigh University benchmark [Guo *et al.*, 2003, Guo *et al.*, 2004] provides a set of queries for evaluating DAML+OIL and OWL repositories. These queries cover a range of properties like input size, selectivity, complexity, assumed hierarchy information, and assumed inference. The queries are described in a KIF-like language and have been written in other languages such as RQL or Jess.

These queries have been designed with the following evaluation criteria in mind: the load time for storing data in the repository, the repository size after loading data in it, the response time of the repository when answering queries, and the completeness and soundness of the repository regarding the queries.

5.3.3 WebODE Performance Benchmark Suite

The WebODE Performance Benchmark Suite[13] [García-Castro and Gómez-Pérez, 2004] has been developed by the Ontology Engineering Group of the Universidad Politécnica de Madrid. Although the long-term goal of the benchmark suite is to achieve a continuous improvement in the WebODE platform, there are other short-term goals such as: assessing the platform performance, monitor the platform, and diagnose problems in the platform.

The WebODE Performance Benchmark Suite can be downloaded from its web page.

[11]http://www.w3.org/TR/rdf-testcases/
[12]http://www.w3.org/TR/owl-test/
[13]http://kw.dia.fi.upm.es/wpbs/

It is composed of a set of benchmarks that perform ontology management operations in the WebODE platform, storing their execution times. These operations are performed through the methods of the WebODE Ontology Management API.

This benchmark suite can be used to evaluate the scalability of WebODE, and could be adapted to also work with other ontology development tools. As it executes all the methods in the WebODE Ontology Management API, it can also be used to evaluate the robustness of the tool.

5.4 Supporting tools

This section presents a list of the different types of tools that can be useful when performing benchmarking activities on ontology development tools and tool suites.

5.4.1 Workload generators

Workload generation is one of the main issues to automate ontology tools evaluation. It consists in producing input ontologies for experiments in an automatic way and according to some parameters. Chapter 4 presents some general workload generators that could be used with ontology development tools. In this section, we present three different workload generators specific for ontology development tools.

5.4.1.1 OntoGenerator

OntoGenerator[14] [OntoWeb, 2002] is an OntoEdit plug-in that creates, according to a set of parameters, synthetic ontologies for performance tests of ontology-based tools. These ontologies are not intended to represent a domain of interest, but rather to fulfil certain technical parameters, for example, a certain number of concepts and instances or certain kinds of rules.

5.4.1.2 Lehigh University Benchmark Data Generator

Guo et al. [2003, 2004] developed the Univ-Bench Artificial data generator[15] in order to facilitate the evaluation of Semantic Web repositories. It generates synthetic OWL or DAML+OIL instances over a ontology in the university domain. These instances are repeatable and customisable, by allowing user to specify several creation parameters.

[14]http://www.ontoprise.de/products/ontoedit_plug-ins_en

[15]http://www.lehigh.edu/~yug2/Research/SemanticWeb/LUBM/LUBM.htm

5.4.1.3 WebODE Workload Generator

García-Castro and Gómez-Pérez [2004] presented the workload generator of the WebODE Performance Benchmark Suite[16], which generates synthetic ontologies in the WebODE knowledge model according to a set of load factors that define the load the tool will have when performing the experiments. These ontologies can be later exported to several languages (RDF(S), OIL, DAML+OIL, OWL, etc.) with the WebODE's export services.

5.5 Conclusion

This chapter has presented part of the framework for benchmarking ontology development tools. Specifically, it has enumerated the most relevant tools to consider for benchmarking and, according to the topics of interest in Knowledge Web workpackage 2.1 (scalability, robustness, and interoperability); it has also presented the general evaluation criteria, benchmark suites, and supporting tools that could be used when benchmarking ontology development tools.

There are just a few benchmark suites for evaluating ontology development tools. New benchmark suites will have to be developed through Knowledge Web's benchmarking activities.

In order to automate part of the experimentation in the future benchmarking activities, workload generators can be used. These are publicly available and can be easily adapted for using with many ontology development tools.

[16]http://kw.dia.fi.upm.es/wpbs/WPBS_workload_generation.html

Chapter 6

Benchmarking ontology-based annotation tools

by DIANA MAYNARD

This chapter presents part of the framework for benchmarking ontology-based annotation tools, in order to provide a first overview of the future benchmarking activities that will involve these tools in Knowledge Web.

6.1 Candidate tools

This section presents a list of the candidate ontology-based annotation tools to be benchmarked and a brief description of the tool and the reasons for its inclusion in the list. The following tools were chosen on the basis that they are available for free, ready-to-use, XML-based, and do not require extensive training of the user, so that evaluation can be as easy and quick to perform as possible.

- KIM
- OntoMat
- MnM
- Melita
- GATE

6.1.1 KIM

KIM [Popov *et al.*, 2004] provides a Knowledge and Information Management (KIM) infrastructure and services for automatic semantic annotation, indexing, and retrieval of unstructured and semi-structured content. Within the process of annotation, KIM also performs ontology population. As a baseline, KIM analyzes texts and recognizes entities (Persons, Organizations, Locations, Dates). Then it tries to match the reference with a known entity, having a unique URI and a description in the knowledge base. Alternatively, a new URI and entity description are automatically generated. Finally, the reference in the document gets annotated with the URI of the entity. KIM is a platform which offers a server, a web user interface, and an Internet Explorer plug-in. KIM is equipped with an upper-level ontology (KIMO) of about 250 classes and 100 properties. Further, a knowledge base (KIM KB), pre-populated with up to 200,000 entity descriptions, is bundled with KIM. In terms of underlying technology, KIM uses GATE, Sesame, and Lucene. KIM is used in the SEKT and SWAN projects.

6.1.2 OntoMat Annotizer

OntoMat Annotizer [Handschuh *et al.*, 2002] is a user-friendly interactive annotation tool for web pages. It supports the user in the task of creating and maintaining ontology-based OWL markups, i.e. creating instances, attributes and relationships. It includes an ontology browser for the exploration of the ontology and instances, and an HTML browser that displays the annotated text. It is Java-based and provides a plug-in Interface for extensions. The intended user is the individual annotator, i.e. somebody who wants to enrich their web pages with OWL metadata. Instead of manually annotating the page with a text editor, OntoMat allows the annotator to highlight relevant parts of the web page and create new instances via drag and drop interactions. It supports the metadata creation phase of the lifecycle. It is used in the OntoAgent project.

6.1.3 MnM

MnM [Motta *et al.*, 2002] is an annotation tool which provides both automated and semi-automated support for annotating web pages with semantic contents. MnM integrates a web browser with an ontology editor and provides open APIs to link this editor to ontology servers and to integrate information extraction tools.

6.1.4 Melita

Melita [Ciravegna *et al.*, 2002] is a semi-automatic ontology-based text annotation tool. It implements a methodology to manage the whole annotation process for the users. Several steps in the annotation process which are usually performed manually are automated and handled by the system, thereby saving time and cost. The main competences of Melita can be summarised into four groups: the Managing task, the Extraction, the Learning and the Autonomous Information Tagging. These are performed thanks to the use of a smart interface together with a powerful Information Extraction algorithm. Melita uses GATE and Amilcare as underlying technology.

6.1.5 GATE

GATE [Cunningham *et al.*, 2002] is a tool for: scientists performing experiments that involve processing human language; companies developing applications with language processing components; teachers and students of courses about language and language computation. GATE comprises an architecture, framework (or SDK) and development environment, and has been in development since 1995 in the Sheffield NLP group. The system has been used for language processing projects; in particular for Information Extraction in many languages, is used in SEKT and SWAN, and its IE technology is used in other annotation tools and IE systems such as Melita, KIM, Amilcare and Magpie. GATE is funded by the EPSRC and the EU.

6.2 General evaluation criteria

This section presents the different functionalities that ontology-based annotation tools have, along with the general criteria that can be used when evaluating or when benchmarking these functionalities. The criteria presented are related to workpackage 2.1 topics (scalability, robustness, and interoperability).

Dipper et al. [2004] provide (amongst others) the following set of functionality criteria for annotation tools:

- diversity of data: support of different character sets, modalities etc. (e.g. spoken vs. written text);
- diversity of annotation: support of different data types of annotation, e.g. attribute-value pairs, set relations etc.;
- simplicity: using annotation tools should not require extensive training or prior knowledge of the tools;
- customisability: support for creating new tagsets etc.;
- convertibility: support for conversion of data into different formats, either by providing a standardised input and output format, or by providing converters from/to other tools.

We can translate these criteria into a set of evaluation criteria which includes (but is not limited to) the following:

- performance;
- scalability;
- robustness;
- usability;
- interoperability.

6.2.1 Performance

Tools that perform automatic or semi-automatic semantic annotation should be evaluated on their performance, i.e. on their ability to associate the text with the correct concepts in the ontology by means of associating mentions in the text with instances in that ontology. Semantic annotation may also require a certain amount of disambiguation of the entities in the text with respect to instances in the ontology, for example, correctly disambiguating ”Cambridge” in the text to the correct entity ”Cambridge, UK” vs ”Cambridge, MA”. Some semantic annotation tools may also make decisions about when a new instance needs to be added to the ontology, because the text contains a new instance that does not already exist in the ontology: this needs to be evaluated according to a given gold standard ontology.

The evaluation performance task aims to discover all mentions of instances from the ontology in the text. The gold standard is a set of texts annotated with instances and concepts from the ontology. The metric needs to measure how good the IE system is at discovering all the mentions, and whether the correct class and instance have been assigned to each mention. The standard metric used for this type of evaluation is based on Precision and Recall, although this has limitations in that it provides a binary kind of measure whereby something is considered either correct or incorrect. Evaluations involving ontologies should preferably have a more scalar aspect, so that while two solutions can both be wrong, one could be ”less wrong” than the other. An alternative to Precision and Recall is a cost-based metric, where different kinds of error are weighted differently. There are no current standards for this kind of evaluation widely accepted, since it is a new area of research. A detailed account of evaluation criteria for natural language applications (including ontology-based annotation tools) is given, however, in D1.2.3.

6.2.2 Scalability

Scalability is an important factor in the evaluation of annotation tools. Some tools are designed for high performance on a small number of documents, while others are designed mostly for large-scale annotation. In general, there is a trade-off between correctness, size and speed – tools designed for large-scale annotation can generally perform very fast annotation, but have less reliable results, while those designed for very accurate results are often slower and work best on small sets of domain-specific documents.

Tests for scalability typically involve measuring both the accuracy and speed of annotation over different numbers of concepts and instances: for example, testing over 20/200/2000 concepts, and 100/1000/10000 instances. Note that speed of annotation may also be an independent factor in the evaluation, regardless of the corpus size, and can also be evaluated as a separate issue.

6.2.3 Robustness

As mentioned in Section 6.2.2, systems designed for high performance on small datasets tend to be designed for use on domain-specific documents. One of the problems with this is that it may be difficult to adapt them to new domains and applications. Systems designed for large scale annotation tend to be more robust in that they can deal with any kind of domain; however, as mentioned earlier, for this they must generally sacrifice a degree of performance. The robustness of a system may be an important factor when deciding on the most appropriate system to use. If the system is going to be used on a particular type of text or application this may not be an issue but, if it may be used for a wide variety of purposes it is clearly important.

Robustness can be measured by testing the performance of the systems on different kinds of domains, for example business news, sport, foreign news, culture, or on even wider variations of text such as meeting announcements, emails, news, dialogue, etc.

6.2.4 Usability

In contrast with performance and functionality, usability considers how easy the tool is to use, regardless of how well it performs on the actual annotation task. Usability covers issues such as:

- familiarity: how long it takes to familiarise oneself with the tool
- documentation: how extensive and relevant the documentation is
- aesthetics: how visually pleasing the interface is
- compliance: how well it complies with existing usability standards (e.g. whether it uses commonly understood icons such as a question mark for help)
- accessibility: how accessible the tool is for people with different needs (colour blind, RSI sufferers, dyslexics, visually impaired etc), e.g. tool tips for icons, keyboard shortcuts as an alternative to the mouse, minimisation of the number of mouse clicks or keys to be pressed, use of appropriate colour schemes, ability to change the font size, etc.

6.2.5 Interoperability

Interoperability considers how well the tool interacts with other tools and systems. Annotation is a task that is often combined with other applications, such as browsing, search and retrieval, indexing, etc., so it is important that annotation tools can easily interact with other systems. This is best achieved by complying with existing standards, such as data format. Interoperability evaluation covers issues such as:

- data format: what kinds of text format can be processed, e.g. xml, html, sgml, txt, etc.;
- annotation format: whether standoff markup is used (generally more flexible than inline annotation – see discussion in Section 6.3.2);
- annotation schemes: whether annotation schemes can be imported/exported from other tools;
- plug-ins: if it is possible to plug in other tools;
- converters: if converters to/from other formats are provided if non-standard formats are used.

6.3 Benchmark suites

This section presents the different benchmark suites that can be used when evaluating or benchmarking ontology-based annotation tools. In the context of annotation tools, a benchmark suite should be a corpus of text which has been annotated according to a gold standard. This annotation is either carried out manually, or semi-automatically (i.e. it is first annotated automatically and then corrected by hand). Because the creation of annotated corpora as testbeds involves trained linguists and is a time-consuming and expensive task, such corpora tend to be small but richly annotated, because the data used is restricted to that which is highly relevant.

6.3.1 Inter-annotator agreement

Typically some portion of the corpus will be double annotated (annotated by more than one person) in order to ensure consistency. Inter-annotator agreement (IAA) can be measured to give an idea of (a) how difficult the task is for humans and (b) how much error is likely to be present. If IAA is low, then steps need to be taken to resolve this problem. It may mean that the guidelines for annotation are unclear, or it may be that they need to be modified because they do not make sense with respect to the task and/or corpus in question. If IAA is low simply because the task is hard, and nothing can be done about this, it may be that a single-answer model is not the best solution. The higher the IAA, the more reliable we can consider the gold standard.

6.3.2 Format of data

Annotated corpora should be available in some generic format, typically sgml, xml or html. There are two formats in which annotation is typically stored: standoff markup and inline markup. Standoff markup is generally more versatile, since the original document is preserved untouched, while inline markup is often preferred because it is more generic, simpler, and easily read.

Figure 6.1 shows a small sample of text “John bought 6 books.” marked with inline POS annotations. Figure 6.2 shows part of the same text marked with standoff POS annotations.

```
<Token  length="4" category="NNP"
 kind="word" string="John">John</Token>
<Token  length="6" category="VBD"
 kind="word" string="bought">bought</Token>
<Token length="1" category="CD"
 kind="number" string="6">6</Token>
<Token  length="1" category="NN"
  kind="word" string="books">books</Token>
<Token length="1" category="."
  kind="punctuation" string=".">.</Token>
```

Figure 6.1: Example of inline POS annotations

6.3.3 Corpora available

There are many corpora annotated with standard Named Entities (such as Person, Location, Organization etc.), which have been used for official evaluations such as MUC and ACE. These can be used to evaluate the performance of systems on such entity types, which will give an idea of their quality at a basic annotation task.

6.3.3.1 MUC7

The MUC-7 corpus consists of a set of 100 newswires in English, as SGML files, annotated with standoff markup for the types Person, Location, Organization, Money, Percent,

```
<Node id="106"/>John<Node id="110"/>
<Node id="111"/>bought<Node id="117"/>
<Node id="118"/>6<Node id="119"/>
<Node id="121"/>books<Node id="128"/>
<Annotation Type="Token" StartNode="106" EndNode="110">
<Feature>
  <Name className="java.lang.String">orth</Name>
  <Value className="java.lang.String">upperInitial</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">category</Name>
  <Value className="java.lang.String">NNP</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">length</Name>
  <Value className="java.lang.String">4</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">kind</Name>
  <Value className="java.lang.String">word</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">string</Name>
  <Value className="java.lang.String">John</Value>
</Feature>
</Annotation>
```

Figure 6.2: Example of standoff POS annotations

Date, and Time. There is extensive documentation and guidelines available, and the corpus has been divided into training and test sets. Inter-annotator agreement is very high (around 97%) so this can be regarded as having a high standard.

An example of the MUC-annotated data is shown below:

```
<p>Endeavour, with an international crew of six, was set to
blast off from the <ENAMEX TYPE="ORGANIZATION|LOCATION">
Kennedy Space Center</ENAMEX> on <TIMEX TYPE="DATE">
Thursday</TIMEX> at <TIMEX TYPE="TIME">4:18 a.m. EST
</TIMEX>, the start of a 49-minute launching period. The
<TIMEX TYPE="DATE"> nine day</TIMEX> shuttle flight was to
be the 12th launched in darkness.
```

6.3.3.2 ACE

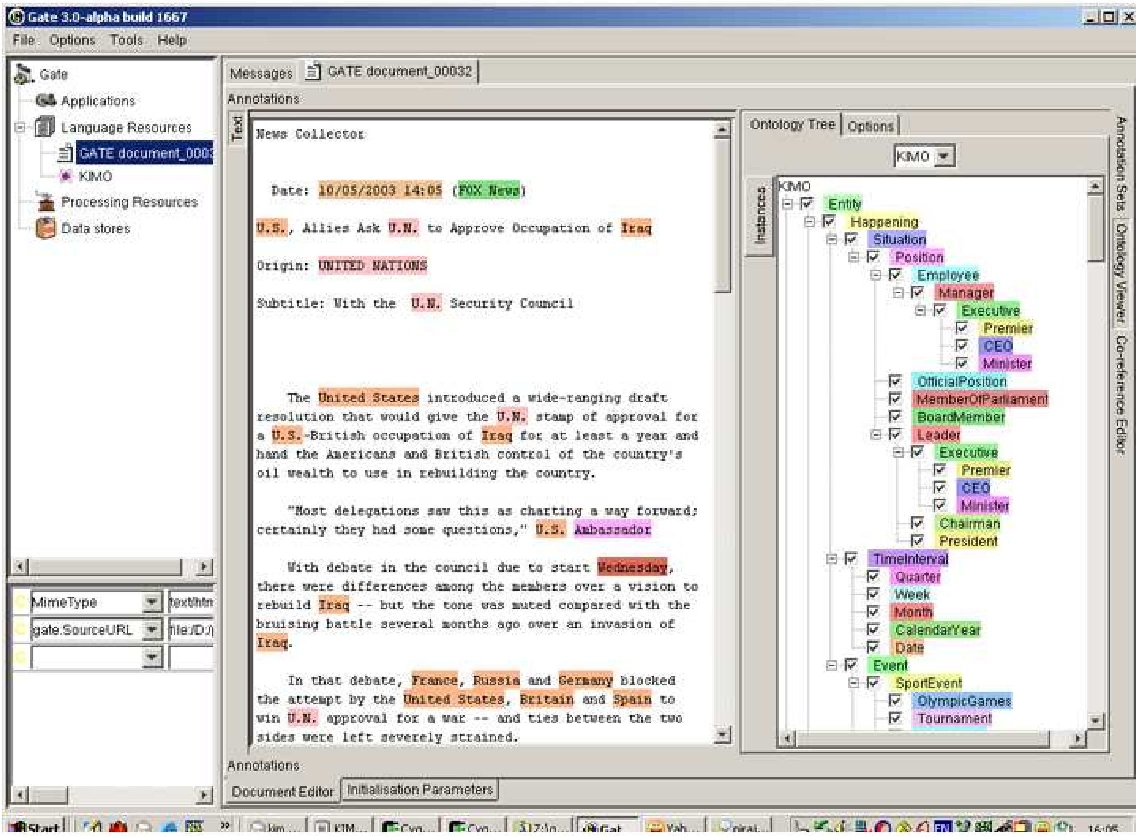
The ACE corpus is similar to the MUC-7 corpus, but more oriented towards ontology-based annotation, in that it contains a two-level ontology of most entities. For example, locations are additionally annotated with their location type (city, country, etc). ACE extends the annotation method in MUC-7 towards a more semantic kind of interpretation, for example marking examples of metonymous usage of instances (where their intended meaning is a different semantic category from their literal meaning). The ACE corpus also measures system robustness (see Section 6.2.3) as it contains texts from different domains and genres, including both degraded texts (output of ASR and OCR systems) and their ”cleaned” versions, and texts from newspapers, newswires and broadcast news on a variety of different topics. Also unlike MUC-7, which deals only with proper names (for the NE task), the ACE corpus is annotated with proper names (e.g. England, Mr. Smith, IBM), pronouns (e.g., he, she, it) and nominal mentions (e.g. the company, the spokesman), where these refer to entities. The idea is that systems should identify which mentions in the text refer to which entities, e.g. (Tony Blair, Mr. Blair, he, the prime minister, he) might all refer to the same entity.

An example of a fragment of the ACE markup is shown below. It describes various related instances of the entity ”National Air Traffic Services” which has the type ”Organization”: for example, the name mentions (proper nouns) ”National Air Traffic Services”, ”NATS” and ”Nats”, and the pronominal mention ”its”.

```
ACE example:
<entity ID="ft-airlines-27-jul-2001-2"
     GENERIC="FALSE"
     entity_type = "ORGANIZATION">
     <entity_mention ID="M003"
                     TYPE = "NAME"
                     string = "National Air Traffic Services">
     </entity_mention>
     <entity_mention ID="M004"
                     TYPE = "NAME"
                     string = "NATS">
     </entity_mention>
     <entity_mention ID="M005"
                     TYPE = "PRO"
                     string = "its">
     </entity_mention>
     <entity_mention ID="M006"
                     TYPE = "NAME"
                     string = "Nats">
     </entity_mention>
  </entity>
```

6.3.3.3 OntoNews

The OntoNews corpus was created for the purpose of the SEKT project[1], and contains news documents annotated according to the KIMO ontology [Popov *et al.*, 2003b]. KIMO is a simplistic upper-level ontology containing 250 general entity types and 100 entity relations. Typical entity types such as Person and Location are subdivided into categories such as Mountain, River, etc. Figure 6.3 shows a piece of text annotated with the KIMO ontology in GATE, with different annotations highlighted in different colours.



News Collector

Date: 10/05/2003 14:05 (FOX News)

U.S., Allies Ask U.N. to Approve Occupation of Iraq

Origin: UNITED NATIONS

Subtitle: With the U.N. Security Council

The United States introduced a wide-ranging draft resolution that would give the U.N. stamp of approval for a U.S.-British occupation of Iraq for at least a year and hand the Americans and British control of the country's oil wealth to use in rebuilding the country.

"Most delegations saw this as charting a way forward; certainly they had some questions," U.S. Ambassador

With debate in the council due to start Wednesday, there were differences among the members over a vision to rebuild Iraq -- but the tone was muted compared with the bruising battle several months ago over an invasion of Iraq.

In that debate, France, Russia and Germany blocked the attempt by the United States, Britain and Spain to win U.N. approval for a war -- and ties between the two sides were left severely strained.

Figure 6.3: Text annotated with the KIMO ontology

6.4 Supporting tools

The GATE corpus benchmark tool (CBT) enables two versions of an annotated corpus to be compared, i.e. the annotations produced by the system to be evaluated, and the gold standard (or ”key”) annotations. For each annotation type, figures are generated for precision, recall, F-measure and false positives, and other metrics can be generated as necessary. The output of the tool is written in an HTML file in tabular form.

[1]Semantically Enabled Technologies, EU-IST Project IST-2003-506826

6.5 Conclusion

This section has described a methodology for benchmarking ontology-based annotation tools, including a set of proposed tools to be evaluated. It is important to note that because ontology-based annotation is a relatively new task, metrics for their evaluation (and specifically, that of performance evaluation) are not well established and development of such metrics is an ongoing research task. It is also important to stress that evaluation of such tools should always be on an application-specific basis, depending on the particular requirements of the end user, and it would be inappropriate to evaluate such tools on an overall ranking basis.

Chapter 7

Benchmarking ontology-based reasoning tools

by HOLGER WACHE

The Semantic Web comes with a series of languages for representing ontological knowledge: RDF/RDFS and the three flavours of OWL: OWL Lite, OWL DL and OWL Full. The main characteristic of the OWL sequence of languages is its growing representation power together with its growing reasoning complexity. RDF/RDFS may have a strong connection to database technologies but OWL Lite and OWL DL relate to description logics. Or, in other words, the reasoning tools depend on the ontology representation language.

Expressive description logic is more complex and might prevent someone to use it and prefer RDF/RDFS. However, for practical and useful cases we strongly believe that simple expressive power of RDF/RDFS is not enough for the Semantic Web and we favour reasoning about ontologies in expressive languages.

Expressive languages based on description logics show how they can be applied in real practical situations with practical problems. Therefore, we would like to focus on benchmarking reasoning tools with OWL Lite, OWL DL, and OWL Full in terms of scalability and robustness. In order to categorise the results we will compare them with the selected RDF/RDFS systems.

This chapter presents part of the framework for benchmarking ontology-based reasoning tools, in order to provide a first overview of the future benchmarking activities that will involve these tools in Knowledge Web.

7.1 Candidate tools

The different ontology-based reasoning tools that will be considered in the benchmarking activities that will be performed in the context of workpackage 2.1 can be classified in three groups: logical reasoners, developments of Knowledge Web partners, and RDF/RDFS systems.

7.1.1 Logical reasoners

In order to reason with expressive description logics languages only a few reasoning systems exist. The most popular logical reasoners are FaCT [Horrocks, 1998] and RACER [Haarslev and Möller, 2001]. Both systems are mature description logics reasoners. There are other description logics reasoners available, such as different variants of FaCT (iFaCT [Horrocks, 1999] or FaCT++) or DLP [Patel-Schneider and Horrocks, 1999] but they are under development and will not be considered during benchmarking until they become mature. Furthermore, some specialised reasoners will be benchmarked. PELLET is a reasoner for OWL and logical entailment. InstanceStore [Horrocks *et al.*, 2004] is for reasoning with large sets of instances and uses database methods for this purpose.

7.1.2 Own developments

Apart from the existing systems we intend to benchmark our own developments. The prototypes are designed to overcome the expected drawbacks of the existing systems, e.g. approximate reasoning for introducing anytime behaviour into the reasoning process to achieve reasonable results faster. An example is a distributed description logics reasoner [Serafini and Tamilin, 2004] which uses distributed processing for seeping up the inferences.

7.1.3 RDF/RDFS Systems

All these systems and prototypes will be compared to RDF/RDFS systems. We do not want to evaluate all existing RDF/RDFS systems but only a representative collection that includes Sesame [Broekstra *et al.*, 2002] as a reference to the other systems in terms of expressiveness and the criteria described in the next section.

7.2 General evaluation criteria

The general evaluation criteria for reasoning services are obvious. In order for the Semantic Web to become a success, the reasoning services must be efficient and can not consume many resources. Especially for the Semantic Web, the investigated systems should get on well with the growing and dynamic nature of the (semantic) web. Even the latter point requires scalable and robust methods.

7.2.1 Performance

The most important criterion to evaluate reasoning is their performance, i.e. the amount of inference steps, the time needed and the space requirements. This criterion will also serve as an evaluation base for the following criteria.

7.2.2 Scalability

Scalability evaluates the behavior of the reasoning services when the ontology and/or instance base is growing. For example, how does the performance change if an ontology grows about a quarter or a half of its original size?

7.2.3 Robustness

Even for large ontologies, reasoning services must support the user and/or the application. For example, they must be robust with respect to some inconsistencies in the ontology. In the case where the system cannot find logical correct answers to a query, it should relax the query and return the most appropriate answers.

7.3 Benchmark suites

The benchmark suites are mainly defined by the data sets which should be benchmarked. The data sets should be designed to reflect the most typical use-cases in the Semantic Web. For this purpose the data set should be real world examples and not artificial generated ones.

In general, we see two potential use-cases for the Semantic Web concerning ontology-based reasoning. A third case would be a combination of the first two cases:

1. **Concept reasoning**
   The user wants to understand a given ontology. Therefore concept definitions in the ontology are compared and the subsumption hierarchy is navigated. The reasoning is the normal T-Box reasoning. In terms of scalability and robustness even very large ontologies are of interest. The benchmark suite should consist of large ontologies with perhaps thousands of concept definitions. The concept definitions themselves should be as complex as possible in order to demonstrate and use the expressive power of OWL. This may be difficult to obtain.

2. **Instance retrieval**
   All instances according to a query are selected. This will be the most prominent use case. The ontology itself might be small but the set of instances must be large.

Besides the ontology and the set of instances a representative collection of queries is also important for a meaningful benchmark suite.

3. **Instance retrieval with large ontologies**
   This use case is an combination of concept reasoning and instance retrieval.

Some of these use-cases are already defined by the workpackage 2.4 which provides typical semantic web service scenarios. But also the benchmark suites for the description logic reasoner challenge in the year 1998 [Franconi *et al.*, 1998] may serve as benchmark suites.

7.4 Supporting tools

For benchmarking ontology-based reasoning tools no specific supporting tools are available. The Vrije Universiteit Amsterdam implements its own monitoring tool for approximate reasoning on top of the DIG interface [Bechhofer *et al.*, 2003]. The analysis of the results is done by normal spreadsheets such as MS Excel, etc. Perhaps some statistical package may help us to analyse large protocols.

7.5 Conclusion

This section gives an overview of which requirements must be fulfilled for benchmarking ontology-based reasoning. The core candidate tools are introduced and classified into three groups. These systems will be evaluated according performance, scalability and robustness. Furthermore some candidates for benchmark suites are determined but further search according to the proposed requirements is needed.

Chapter 8

Benchmarking semantic web service technology

by DOUG FOXVOG AND RAFAEL GONZÁLEZ-CABERO

This chapter presents part of the framework for benchmarking semantic web service technology, in order to provide a first overview of the future benchmarking activities that will involve these tools in Knowledge Web.

8.1 Candidate tools

This section presents a list of the candidate semantic web service tools to be benchmarked along with a brief description of the tool and the reasons for its inclusion in the list. The following tools were chosen on the basis that they are readily available and appear to have a running version.

- WSMX
- SWWS Studio
- wsml4j
- ODE SWS
- OWL-S 2 UDDI Matchmaker
- Internet Reasoning Service (IRS III)
- Semantic Web Service Language Editors
  - WSMO Editor
  - OWL-S Editor

- Language Translators
  - WSDL2OWL-S
  - Java2OWL-S
  - OWL-S 2 UDDI
- Validation Services
  - WSML Validator
  - OWL-S Validator

8.1.1 WSMX

WSMX [Moran and Mocan, 2004] is an execution environment for the dynamic discovery, selection, mediation, invocation and inter-operation of Semantic Web Services whose semantics have been formally described. The WSMX manager controls the operational flow and event management for a semantic web service process. WSMX Matchmaker attempts to provide a service to fulfil a certain goal. Its Selector selects the service discovered that best matches the goal based on preferences of the service requested. An XML Converter is included in case the language of the service needs to be translated from a logical language syntax into XML. WSMX handles mediation of data, processes, and business protocols between (or among) web services that have modeled these aspects in different ways. The mediator provides and uses mapping rules in this conversion. WSMX has a graphical user interface that allows the user to view ontologies and the status of execution.

8.1.2 SWWS Studio

SWWS Studio [Dimitrov *et al.*, 2004] is a WSMO compliant Semantic Web Services editor. It includes a Service Designer - a graphical editor for modelling WSMO descriptions, a GUI for modelling web service compositions, and a registry service for storing, browsing, and finding service descriptions.

8.1.3 WSMO4J

WSMO4J is an API for the Java programming language to build semantic web services applications compliant with the Web Service Modeling Ontology. Currently it is compliant with the WSMO v1.0 specification.

8.1.4 ODESWS

ODESWS [Gómez-Pérez *et al.*, 2004] is an environment for the design and composition of semantic web services at knowledge level. It proposes to use Problem-Solving Method, to describe semantic web services and once they are created in this language independent way, they can be translated into a concrete semantic web service language. The architecture of the ODESWS is composed of several modules. The most important that have been released up to now are: SWSDesigner, a graphical editor based on the assumption that the design and development of a service is performed by creating graphically the different views of the Problem-Solving Method; and SWSTranslator, a module that generates from these language independent descriptions of semantic web services a description written in an implementation language (currently OWL-S).

8.1.5 OWL-S 2 UDDI Matchmaker

Matchmaker [Kawamura *et al.*, 2003] helps to establish connections between service requesters and providers. This is described as the ”yellow pages” of service capabilities. Matchmaker establishes a mechanism for registering service capabilities and enabling users and/or software agents to find each other through semantic queries. Registration information is stored as advertisements. When Matchmaker receives a request for a service, it searches its repository of advertisements for agents that can fulfill that request.

8.1.6 Semantic Web Service Composer

IBM's Semantic Web Service Composer[1] is a semantic matching and composition engine that can help service requesters to find and compose suitable Web Services. If a single service cannot be found to meet the given requirements, the engine will use AI planning algorithms to attempt to find a composition of services to meet the requirements.

8.1.7 Internet Reasoning Service (IRS III)

The Internet Reasoning Service III (IRS-III) [Domingue *et al.*, 2004] is a platform and infrastructure for creating WSMO-based Semantic Web Services, building upon the previous implementation, IRS-II [Motta *et al.*, 2003]. The specific extensions in IRS-III are: (1) UPML based types of knowledge models: domain, task, problem solving method and application have been extended to include goal models, mediator models and web service models; (2) the WSMO ontology has been implemented and slightly extended in OCML; (3) a WSMO specific Java API has been created and (4) the IRS browser has been customised to reflect WSMO.

[1] http://www.daml.org/services/owl-s/tools.html

8.1.8 OWL-S Editor

The OWL-S Editor [Scicluna, 2004] is comprised of three main parts: Creator, Validator, and Visualizer. Creator enables the creation of a new OWL-S description based on either a template or a WSDL description. The conversion process is based on the WSDL2OWL-S tool. The Validator validates the syntax of OWL-S ontologies and checks if URIs used in the OWL-S description are valid. The Visualizer produces graphical depictions of the Web Service descriptions either on a computer screen or as hard copy.

8.1.9 WSMO Editor

The WSMO Editor[2] consists of a graphical user environment coded in Java. The main screen is divided into four areas, one for editing a WSMO file, one for showing a hierarchy of compiled concepts, a third for displaying control messages, and a fourth for browsing the file system.

8.1.10 Language Translators

A set of language translators [Popov *et al.*, 2003a] is provided by the Intelligent Software Agents Lab of the Robotics Institute at Carnegie Mellon University for languages used for the Semantic Web and Semantic Web Services. These include translators from Java to WSDL, WSDL to OWL-S, and OWL-S to UDDI.

8.1.11 Validation Services

Validation services accept a code (or the URL for a file that holds the code) in a given language and says whether the code is syntactically valid for the given language or not.

8.1.11.1 WSML Validator

WSML Validator [Lausen and Felderer, 2004] was created using a tool, SableCC, that takes in a syntax in a variant of the Extended Backus Naur Form, and produces a program that will verify whether input text corresponds to that form or not. If it does not correspond to the form, the first faulty line is returned with an explanation. There are two versions of the WSML Validator for two different versions of the WSML language.

[2]http://www.wsmo.org/2004/d9/v01/

8.1.11.2 OWL-S Validator

OWL-S Validator[3] performs a syntactic and structural validation for the OWL-S files to catch the most common errors. It is not designed as a replacement of OWL validators but built to be used in conjunction with them so that OWL-S specific problems can be found.

8.2 General evaluation criteria

This section presents the different evaluation criteria that can be used when evaluating or when benchmarking semantic web service technology. The criteria presented are related to the scalability, robustness, interoperability, and usability of these tools.

While some of the tools are traditional types of tools used in a new area, other tools are special-purpose tools or tool sets that need individual attention. Tools like editors, language translators, and verifiers should be benchmarked in the same way as any other editor, language translator, or verifier.

8.2.1 Scalability

Different scalability criteria can be used when benchmarking semantic web technology. In editors, some criteria could be the ability to view the structure of large ontologies, to operate simultaneously on multiple service repositories, or to work with large services. In translators and verifiers, some criteria could be the ability to work with large files or the speed of the translations/verifications. In semantic web service repositories, some criteria could be the number of simultaneous service requests that can be processed, the average response time to petitions, or the ability to distribute queries among different repositories.

8.2.2 Robustness

Different robustness criteria can be used when benchmarking semantic web technology. Some examples are the ability of validation within an editor, the verification of the language generated by a translator, or the ability to handle different types of errors in verifiers. In semantic web service repositories, some criteria could be the ability to deal with non-standard service petitions. In the case of distributed repositories a criteria could be the ability to work when some of the nodes crash.

[3]http://www.mindswap.org/2004/owl-s/index.shtml

8.2.3 Interoperability

Different interoperability criteria can be used when benchmarking semantic web technology. Some criteria are the ability to handle different input formats (such as text, files, or URLs), the ability to handle different language variants, or the ability to plug the tool into other tools. These criteria can be applied to editors, translators, and verifiers.

8.2.4 Usability

Usability considers how easy a tool is to use, regardless of how well it performs. This covers issues such as:

Familiarity: how long it takes to familiarise oneself with the tool

Documentation: how extensive and relevant the documentation is

Aesthetics: how visually pleasing the interface is

Compliance: how well it complies with existing usability standards

Accessibility: how accessible the tool is for people with different needs (colour blind, RSI sufferers, dyslexics, visually impaired etc), e.g. tool tips for icons, keyboard shortcuts as an alternative to the mouse, minimisation of the number of mouse clicks or keys to be pressed, use of appropriate colour schemes, ability to change the font size, etc.

8.3 Benchmark suites

To the best of the authors' knowledge, there are no specific benchmark suites for semantic web technology benchmarking. In Knowledge Web, different benchmark suites will be developed for benchmarking semantic web technology in order to assess the performance of these tools and the interoperability between the different types of tools.

8.4 Supporting tools

To the best of the authors' knowledge, there are no specific supporting tools for semantic web technology benchmarking. The tools commonly used in evaluation of semantic web service technology are general ones.

8.5 Conclusion

This section has described a framework for the evaluation of semantic web service tools, including a set of proposed tools to be evaluated. It is important to note that because the field of semantic web services is a relatively new one and the range of tools is so varied, metrics and benchmark suites for their evaluation (and in particular, that of performance evaluation) in general are not well established and development of such metrics and benchmark suites is an ongoing research task. It is also important to stress that evaluation of such tools should always be on an application-specific basis, depending on the particular requirements of the end user, and it would be inappropriate to evaluate such tools on an overall ranking basis.

Chapter 9

Conclusion

by RAÚL GARCÍA-CASTRO

This deliverable proposes a framework for the benchmarking activities that must be performed in Knowledge Web. Although some chapters of the deliverable are focused on workpackage 2.1 topics, its contents can also be helpful in other workpackages (1.2 and 2.2).

The proposed framework is composed of a benchmarking methodology, a set of guidelines for developing benchmark suites, and information about several types of tools that can be used to support benchmarking.

Regarding workpackage 2.1, there are different chapters that introduce the benchmarking activities that must be performed. These chapters give an overview of the different tools that are candidates to be benchmarked, the general evaluation criteria that can be used with these tools according to workpackage 2.1 topics (scalability, robustness, and interoperability), the different benchmark suites that can be used to measure the tools according to these criteria, and the supporting tools that can be used when benchmarking each kind of tool: ontology development tools, ontology-based annotation tools, ontology-based reasoning tools, and semantic web service technology.

In workpackage 2.1, the benchmarking activities will start on January 2005 and the expected dates for delivering the benchmarking results of each type of tool are the following:

June 2006	Benchmarking of ontology development tools.
December 2006	Benchmarking of ontology-based annotation tools.
June 2007	Benchmarking of ontology-based reasoning tools.
December 2007	Benchmarking of semantic web service technology.

In general, there are few specific instruments to help evaluate ontology tools (benchmark suites, workload generators, etc.). Therefore, benchmarking activities in workpackage 2.1 will involve the development of these kind of instruments to help with the benchmarking and to automate data collection and analysis.

Moreover, as ontology technology is relatively new, in some cases evaluation criteria for ontology tools are not well defined. Nevertheless, most of the evaluation criteria are adopted from software evaluation, which provides widely used definitions and understanding of these criteria.

Chapter 10

Glossary

This chapter presents definitions of different terms used in the deliverable:

Benchmark A benchmark is a test that measures the performance of a system or subsystem on a well-defined task or set of tasks [Sill, 1996].

Benchmarking Benchmarking is a continuous, systematic process for evaluating the products, services, and work processes of organisations that are recognised as representing the best practices for the purpose of organisational improvement [Spendolini, 1992].

Benchmarking partners The benchmarking partners are the organisations that participate in the benchmarking activity.

Benchmarking subject The benchmarking subject is the ontology tool that will be chosen for benchmarking and whose improvement would significantly benefit the organisation.

Interoperability The ability of two or more systems or components to exchange information and to use the information exchanged [IEEE, 1991].

Performance The degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage [IEEE, 1991].

Robustness The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions [IEEE, 1991].

Scalability Ability of a software system to meet its performance objectives when the workload is increased significantly [Weyuker and Avritzer, 2002].

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.

[Bechhofer *et al.*, 2001] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: A reason-able ontology editor for the semantic web. *Lecture Notes in Computer Science*, 2174:396–408, 2001.

[Bechhofer *et al.*, 2003] S. Bechhofer, R. Möller, and P. Crowther. The dig description logic interface. In *Proceedings of International Workshop on Description Logics (DL 2003)*, 2003.

[Broekstra *et al.*, 2002] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: An architecture for storing and querying rdf and rdf schema. In *roceedings of the First International Semantic Web Conference (ISWC 2002)*, number 2342 in Lecture Notes in Computer Science (LNCS), pages 54–68. Springer-Verlag, 2002.

[Bull *et al.*, 1999] J. M. Bull, L. A. Smith, M. D. Westhead, D. S. Henty, and R. A. Davey. A methodology for benchmarking java grande applications. In *Proceedings of the ACM 1999 conference on Java Grande*, pages 81–88, 1999.

[Carroll and Roo, 2004] J.J. Carroll and J. De Roo. OWL web ontology language test cases. Technical report, W3C, February 2004.

[Ciravegna *et al.*, 2002] F. Ciravegna, A. Dingli, D. Petrelli, and Y. Wilks. User-System Cooperation in Document Annotation Based on Information Extraction. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, pages 122–137, Siguenza, Spain, 2002.

[Cunningham *et al.*, 2002] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[Dimitrov *et al.*, 2004] Marin Dimitrov, Zlatina Marinova, and Peter Radkov. SWWS Studio - a WSMO compliant editor. In *Proceedings of the WSMO Implementation Workshop 2004 (WIW2004)*, 2004.

[Dipper *et al.*, 2004] S. Dipper, M. Götze, and M. Stede. Simple Annotation Tools for Complex Annotation Tasks: an Evaluation. In *Proceedings of the LREC Workshop on XML-based Richly Annotated Corpora*, pages 54–62, Lisbon, Portugal, 2004.

[Domingue *et al.*, 2004] J. Domingue, L. Cabral, F. Hakimpour, D. Sell, and E. Motta. IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services. Proceedings of the Workshop on WSMO Implementations. In *Proceedings of the WSMO Implementation Workshop 2004 (WIW2004)*, 2004.

[Domingue, 1998] J. Domingue. Tadzebao and webonto: Discussing, browsing, and editing ontologies on the web. In B.R. Gaines and M.A. Musen, editors, *11th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, pages 1–20, Banff, Canada, 1998.

[Farquhar *et al.*, 1997] A. Farquhar, R. Fikes, and J. Rice. The ontolingua server: A tool for collaborative ontology construction. *International Journal of Human Computer Studies*, 46(6):707–727, 1997.

[Fernández-López *et al.*, 1999] M. Fernández-López, A. Gómez-Pérez, J. Pazos-Sierra, and A. Pazos-Sierra. Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems*, 14(1), January-February 1999.

[Franconi *et al.*, 1998] Enrico Franconi, Giuseppe De Giacomo, Robert M. MacGregor, Werner Nutt, and Christopher A. Welty, editors. *Proceedings of the International Workshop on Description Logics (DL'98)*, IRST, Povo – Trento, Italy, June 6 - 8 1998.

[García-Castro and Gómez-Pérez, 2004] R. García-Castro and A. Gómez-Pérez. A benchmark suite for evaluating the performance of the WebODE ontology engineering platform. In *Proceedings of the 3rd International Workshop on Evaluation of Ontology-based Tools (EON2004)*, Hiroshima, Japan, November 2004.

[Gómez-Pérez *et al.*, 2003] A. Gómez-Pérez, O. Corcho, and M. Fernández-López. *Ontological Engineering*. Springer Verlag London Ltd., London, UK, 2003.

[Gómez-Pérez *et al.*, 2004] Gómez-Pérez, Rafael González-Cabero, , and Manuel Lama. A Framework for Description, Composition, and Evaluation of Semantic Web Services. *IEEE Intelligent Systems: Special Issue on Semantic Web Services*, 2004.

[Grant and Beckett, 2004] J. Grant and D. Beckett. RDF test cases. Technical report, W3C, February 2004.

[Guo *et al.*, 2003] Y. Guo, J. Heflin, and Z. Pan. Benchmarking DAML+OIL repositories. In *Proceedings of the 2nd International Semantic Web Conference, (ISWC 2003)*, Florida, USA, October 2003.

[Guo *et al.*, 2004] Y. Guo, Z. Pan, and J. Heflin. An evaluation of knowledge base systems for large OWL datasets. In *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*, pages 274–288, Hiroshima, Japan, November 2004.

[Haarslev and Möller, 2001] Volker Haarslev and Ralf Möller. Description of the racer system and its applications. In *Proceedubgs International Workshop on Description Logics (DL-2001)*, 2001.

[Hamill, 2004] P. Hamill. *Unit Test Frameworks*. O'Reilly, November 2004.

[Handschuh *et al.*, 2002] S. Handschuh, S. Staab, and F. Ciravegna. S-CREAM — Semi-automatic CREAtion of Metadata. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, pages 358–372, Siguenza, Spain, 2002.

[Horrocks *et al.*, 1999] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proceedings of the 6th International Conference on Logic Programming and Automated Reasoning (LPAR'99)*, pages 161–180. LNAI, Springer-Verlag, 1999.

[Horrocks *et al.*, 2004] Ian Horrocks, Lei Li, Daniele Turi, and Sean Bechhofer. The instance store: DL reasoning with large numbers of individuals. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, pages 31–40, 2004.

[Horrocks, 1998] I. Horrocks. The FaCT system. In H. de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98*, number 1397 in Lecture Notes in Artificial Intelligence, pages 307–312. Springer-Verlag, 1998.

[Horrocks, 1999] I. Horrocks. FaCT and iFaCT. In P. Lambrix, A. Borgida, M. Lenzerini, R. Mller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics (DL'99)*, pages 133–135, 1999.

[IEEE, 1991] IEEE. *IEEE-STD-610 ANSI/IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology*. IEEE, February 1991.

[Kawamura *et al.*, 2003] Takahiro Kawamura, Jacques-Albert De Blasio, Tetsuo Hasegawa, Massimo Paolucci, and Katia Sycara. A Preliminary Report of a Public Experiment of a Semantic Service Matchmaker combined with a UDDI Business Registry. In *1st International Conference on Service Oriented Computing (ICSOC 2003)*, Trento, Italy, 2003.

[Kraft, 1997] J. Kraft. The department of the navy benchmarking handbook: A systems view. Technical report, Department of the Navy, 1997.

[Lausen and Felderer, 2004] Holger Lausen and Michael Felderer. Architecture for an Ontology and Web Service Modelling Studio . In *Proceedings of the WSMO Implementation Workshop 2004 (WIW2004)*, 2004.

[Maedche *et al.*, 2003] A. Maedche, B. Motik, L. Stojanovic, R. Studer, and R. Volz. Ontologies for enterprise knowledge management. *IEEE Intelligent Systems*, 18(2):26–33, 2003.

[Moran and Mocan, 2004] Matthew Moran and Adrian Mocan. WSMX Ž013 An Architecture for Semantic Web Service Discovery, Mediation and Invocation. In *3rd International Semantic Web Conference (ISWC2004)*, 2004.

[Motta *et al.*, 2002] E. Motta, M. Vargas-Vera, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, pages 379–391, Siguenza, Spain, 2002.

[Motta *et al.*, 2003] E. Motta, J. Domingue, L. Cabral, and M. Gaspari. IRS-II: A framework and infrastructure for semantic web services. In *Proceedings of the 2nd International Semantic Web Conference (ISWC2003)*, Florida, USA, October 2003.

[Noy *et al.*, 2000] N.F. Noy, R.W. Fergerson, and M.A. Musen. The knowledge model of protégé-2000: Combining interoperability and flexibility. In *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*, pages 17–32. Springer-Verlag, 2000.

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

[Patel-Schneider and Horrocks, 1999] P. F. Patel-Schneider and I. Horrocks. DLP and FaCT. In N. V. Murray, editor, *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'99*, number 1617 in Lecture Notes in Artificial Intelligence, pages 19–23. Springer-Verlag, 1999.

[Popov *et al.*, 2003a] B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, A. Kirilov, and M. Goranov. Towards a Semantic Choreography of Web Services: from WSDL to DAML-S. In *Human Language Technologies Workshop at the 2nd International Semantic Web Conference (ISWC2003)*, Florida, USA, 2003.

[Popov *et al.*, 2003b] B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, A. Kirilov, and M. Goranov. Towards Semantic Web Information Extraction. In *Human Language Technologies Workshop at the 2nd International Semantic Web Conference (ISWC2003)*, Florida, USA, 2003.

[Popov *et al.*, 2004] B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. KIM – Semantic Annotation Platform. *Natural Language Engineering*, 2004.

[Scicluna, 2004] James Scicluna. University of Malta, Malta, 2004.

[Serafini and Tamilin, 2004] L. Serafini and A. Tamilin. Local tableaux for reasoning in distributed description logics. In *Proc. of the 2004 International Workshop on Description Logics (DL'04)*, 2004. To appear electronically as CEUR publication.

[Shirazi *et al.*, 1999] B. Shirazi, L.R. Welch, B. Ravindran, C. Cavanaugh, B. Yanamula, R. Brucks, and E. Huh. Dynbench: A dynamic benchmark suite for distributed real-time systems. In *Proceedings of the 11 IPPS/SPDP'99 Workshops Held in Conjunction with the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing*, pages 1335–1349. Springer-Verlag, 1999.

[Sill, 1996] D. Sill. comp.benchmarks frequently asked questions version 1.0, 1996.

[Sim *et al.*, 2003] S. Sim, S. Easterbrook, and R. Holt. Using benchmarking to advance research: A challenge to software engineering. In *Proceedings of the 25th International Conference on Software Engineering (ICSE'03)*, pages 74–83, Portland, OR, 2003.

[Sole and Bist, 1995] T.D. Sole and G. Bist. Benchmarking in technical information. *IEEE Transactions on Professional Communication*, 38(2):77–82, June 1995.

[Spendolini, 1992] M.J. Spendolini. *The Benchmarking Book*. AMACOM, New York, NY, 1992.

[Stefani *et al.*, 2003] F. Stefani, D. Macii, A. Moschitta, and D. Petri. Fft benchmarking for digital signal processing technologies. In *17th IMEKO World Congress*, Dubrovnik, Croatia, 22-27 June 2003.

[Sure *et al.*, 2002] Y. Sure, M. Erdmann, J. Angele, S. Staab, and D. Wenke. Ontoedit: Collaborative ontology engineering for the semantic web. *Proceedings of the first International Semantic Web Conference 2002 (ISWC 2002)*, LNCS 2342:221–235, 2002.

[Swartout *et al.*, 1997] B. Swartout, P. Ramesh, K. Knight, and T. Russ. Toward distributed use of large- scale ontologies. In A. Farquhar, M. Gruninger, A. Gómez-Pérez, M. Uschold, and P. van der Vet, editors, *AAAI 97 Spring Symposium on Ontological Engineering*, pages 138–148, Stanford University, California, 1997.

[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.

[Weyuker and Avritzer, 2002] E.J. Weyuker and A. Avritzer. A metric to predict software scalability. In *Proceedings of the 8th International Symposium on Software Metrics*, 2002.

[Wireman, 2003] Terry Wireman. *Benchmarking Best Practices in Maintenance Management*. Industrial Press, 2003.

Acknowledgements

Thanks to Rosario Plaza and Diana Maynard for reviewing the English grammar of this deliverable.

Related deliverables

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

Project	Number	**Title** and relationship
KW	D1.2.2	**Report on Semantic Web Framework requirements analysis** identifies the main interoperability requirements for ontology tools. These requirements will have to be taken into account when defining specific evaluation criteria for benchmarking these tools.
KW	D1.2.3	**Methods for ontology evaluation** presents methods and tools that enable the user to select the most appropriate ontology for use in industry, by evaluating the content of different ontologies. Whilst the tools that create or manage the ontologies do affect their content, the purpose of the evaluation in this case is not to evaluate such tools themselves.
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	D2.2.2	**Specification of a benchmarking methodology for alignment techniques** presents a methodology with the different evaluations that can be performed on alignment algorithms.