

Whitepaper: Evaluation of Ontology-based Tools

This *whitepaper* aims at encouraging people to contribute work to the OntoWeb-SIG3 Workshop “**Evaluation of Ontology-based Tools (EON2002)**” at the 13th International Conference on Knowledge Engineering and Knowledge Management EKAW 2002, Siguenza (Spain), 30th September 2002. It is based on a draft for the OntoWeb (cf. <http://www.ontoweb.org/>) deliverable D1.3, chapter 4 on “Ontology Evaluation Tools”.

Call for Papers: <http://km.aifb.uni-karlsruhe.de/eon2002>

Important dates:

- Deadline paper submissions: June 28th, 2002
- Notification of acceptance: August 2nd, 2002
- Camera ready deadline: August 30th, 2002
- Workshop: September 30th, 2002 in Siguenza (Spain)

4 Ontology Evaluation Tools

Authors: Jürgen Angele (ontoprise GmbH), angele@ontoprise.de

York Sure (Institute AIFB, University of Karlsruhe), sure@aifb.uni-karlsruhe.de

4.1 Introduction

Currently the semantic web [1.1] attracts researchers from all around the world. Numerous tools and applications of semantic web technologies are already available [1.2,1.3,1.4] and the number is growing fast. Ontologies play an important role for the semantic web as a source of formally defined terms for communication. They aim at capturing domain knowledge in a generic way and provide a commonly agreed understanding of a domain, which may be reused, shared, and operationalized across applications and groups. However, because of the size of ontologies, their complexity, their formal underpinnings and the necessity to come towards a shared understanding within a group of people, ontologies are still far from being a commodity. Developing and deploying large scale ontology solutions typically involves several separate tasks and requires applying multiple tools. Therefore pragmatic issues such as e.g. interoperability are key requirements if industry is to be encouraged to take up ontology technologies rapidly.

The large visibility of the semantic web, it's tools and applications already attracts industrial partners, e.g. in numerous projects funded by the European Commission. In particular, as they move from academic institutions into commercial environments they have to fulfil stronger requirements and in some cases new requirements (e.g. concerning scalability and multi-user access). Different tools from different sources need to interoperate and therefore are typically not anymore standalone solutions but integrated into frameworks. These frameworks must be open to other commercial environments and provide connectors and interfaces to industrial standards. Larger applications need also larger ontologies and therefore require substantially more performance and scalability.

A systematic evaluation of ontologies and related technologies might lead to a consistent level of quality and thus acceptance by industry. For the future, this effort might also lead to standardized benchmarks and certifications.

In this chapter we propose an evaluation framework for properties of ontologies and technologies for developing and deploying ontologies (cf. Section 4.2). We present example implementations of the framework (cf. Section 3), each addressing different aspects of the evaluation criteria proposed in the framework. We compare the tools against our framework (cf. Section 4). Before we conclude (cf. Section 6) we give a brief discussion of related work (cf. Section 5).

4.2 Evaluation framework

Our evaluation framework consists of two main aspects: (i) the evaluation of properties of ontologies generated by development tools, (ii) the evaluation of the technology properties, i.e. tools and applications which includes the evaluation of the evaluation tool properties it selves. In an overview these aspects are structured as follows in Table 1.

Table 1: Evaluation framework criteria

Ontology Properties	Technology Properties
Language conformity (Syntax)	Interoperability (e.g. Semantics)
Consistency (Semantics)	Turn around ability
	Performance
	Memory allocation
	Scalability
	Integration into frameworks
	Connectors and interfaces

For ontologies generated by development tools the language conformity and consistency may be checked. **Language conformity** means that the syntax of the representation of the ontology in a special language is conform to a standard. Such a standard is either a well-documented standard defined by a standardization gremium or it is an industrial standard mostly given by a reference implementation. So in the first case the outcome of an ontology tool must be checked with respect to the syntax definition and in the second case it must be tested using the reference implementation. Evaluation of **consistency** means to what extend the tools ensure that the resulting ontologies are consistent with respect to their semantics, e.g. that different parts of the ontology representation do not contradict.

Ontology properties may be evaluated using the ontologies only, i.e. without having tools, e.g. for development, it selves available. In contrast to that for the second block of properties the tools it selves are examined. **Interoperability** means how easy it is to exchange ontologies between different tools. This includes such aspects as “is a tool able to interpret the outcome of another tool in the same way?”. This is more than only checking the language conformity because it examines whether different tools interpret the same things in the same way. Often things can be represented in the same language in different ways. **Turn around ability** means that the outcome of a tool is represented to the user in the same way again later on. E.g. a value restriction may be represented as a range restriction or by a constraint. If the tool shows that as a range restriction it should not show it as a constraint the next time it reads the same ontology. **Performance** especially concerns the runtime effort of the tools, e.g. how much time is needed for solving a special inference task, for storing ontologies etc. Benchmark tests must be developed to evaluate these performance issues. For these benchmarks reference ontologies, reference ontology classes, reference tasks and reference task classes may be very helpful. **Memory allocation** means how much memory is needed by the tools to handle ontologies. Similarly to the performance evaluation benchmarks must be available to test memory allocation. For performance evaluation as well as for memory allocation it must be clarified what does the “size” of an ontology or the complexity of a task mean and which parameters influence this size in what way etc. **Scalability** evaluates the performance and memory behaviour of the tools with respect to increasing ontologies and tasks. It examines questions like “how increases a linear growth of ontologies the amount of memory allocated by the tool?”. **Integration into frameworks** means how easy it is to switch between such tools. For instance it is not very convenient that for a switch between tools it is necessary to store the ontology, to transform it afterwards with a different tool into another language which is a precondition to load it with the other tool. Entirely integrated environments similar to well-known programming environments must be the goal for ontology development tools. Last but not least, the **connectivity** to other tools is important. This concerns on the one hand connectors for instance to databases, index servers, to systems like MS exchange, Lotus Notes, etc and on the other hand interfaces to the tool itself to use its functionality within other tools.

4.3 Implementations of the framework

4.3.1 Overview

Table 2 shows the evaluated tools in this section.

Table 2: Overview of evaluation tools

Name of the tool	Provider
OntoAnalyser OntoGenerator	Ontoprise GmbH Contact: Jürgen Angele, angele@ontoprise.de Haid-und-Neu-Str. 7 76131 Karlsruhe http://www.ontoprise.de Institute AIFB Contact: York Sure, sure@aifb.uni-karlsruhe.de University of Karlsruhe Postfach 76128 Karlsruhe Germany http://www.aifb.uni-karlsruhe.de
...	...

4.3.2 OntoAnalyser

Based on the criteria shown in the previous section, we (Institute AIFB and ontoprise GmbH) implemented two tools, i.e. OntoAnalyser and OntoGenerator. Both examples for evaluation tools are realized as plugins for OntoEdit which is a graphically oriented ontology engineering environment [3.1]. The underlying plugin framework [3.2] allows for flexible extensions of OntoEdit's core functionalities (also for third parties). Though OntoEdit supports the full lifecycle of ontology development [3.3], some tasks are only weakly supported by the core functionalities. Specialized plugins enable more fine grained support, e.g. for the evaluation of ontologies through the two plugins here presented. Therefore we tightly integrate the evaluation of ontologies that are (i) created with OntoEdit or (ii) imported from other tools like Protégé [3.4] and WebODE [3.5] into the development process.

Each of the two plugins addresses different aspects of the evaluation criteria presented in the previous section. OntoAnalyser focuses on evaluation of ontology properties, in particular language conformity and consistency. OntoGenerator focuses on evaluation of ontology based tools, in particular performance and scalability. We now illustrate each plugin by presenting a motivational use case scenario and describing the functionalities.

From our own experiences of ontology development and deployment (e.g. the creation of an ontology based corporate history analyser or the development of the ontology based portal of our own institute) we learned that for different purposes ontologies must have different properties. These properties might even be different in different ontology projects or for different target applications. For instance a tool which allows to visualize concept hierarchies might require to allow single inheritance only.

The definition of evaluation methods for such properties must be very flexible and easily maintainable. So it is not convenient to program it into a tool. Logic is a very comfortable and powerful way on a very abstract level to express constraints for an ontology or to examine properties of an ontology. For that purpose the rule or constraint language must be able to access the ontology itself, i.e. to make statements about classes, relations, subclasses etc..

F-logic [3.6] allows to define statements and rules about the ontology (concepts, subconcepts, relations) it selves. E.g. the examination whether in an ontology a concept has at maximum one super concept may be expressed by the following rule:

```
FORALL C
    check("concept has more than one super concept",C)
<- EXISTS S1,S2
    C::S1 AND C::S2 AND NOT equal(S1,S2).
```

In the same way it is easy to define rules to check for instance the disjointness of classes, special conventions for relation names etc. E.g. the regularities for a project say, that relations must be named with lower letters:

```
FORALL R
    check("relation name is not in lower case letters",R)
<- EXISTS C,C1,R1
    C[R=>>C1] AND tolower(R, R1) AND NOT equal(R,R1).
```

OntoAnalyser is a tool which takes such rules, runs the Ontobroker inference engine (cf. [3.7]) with such rules and provides the user the results of this examination. OntoAnalyser is able to load different rule packages, each intended for a different target tool or target project.

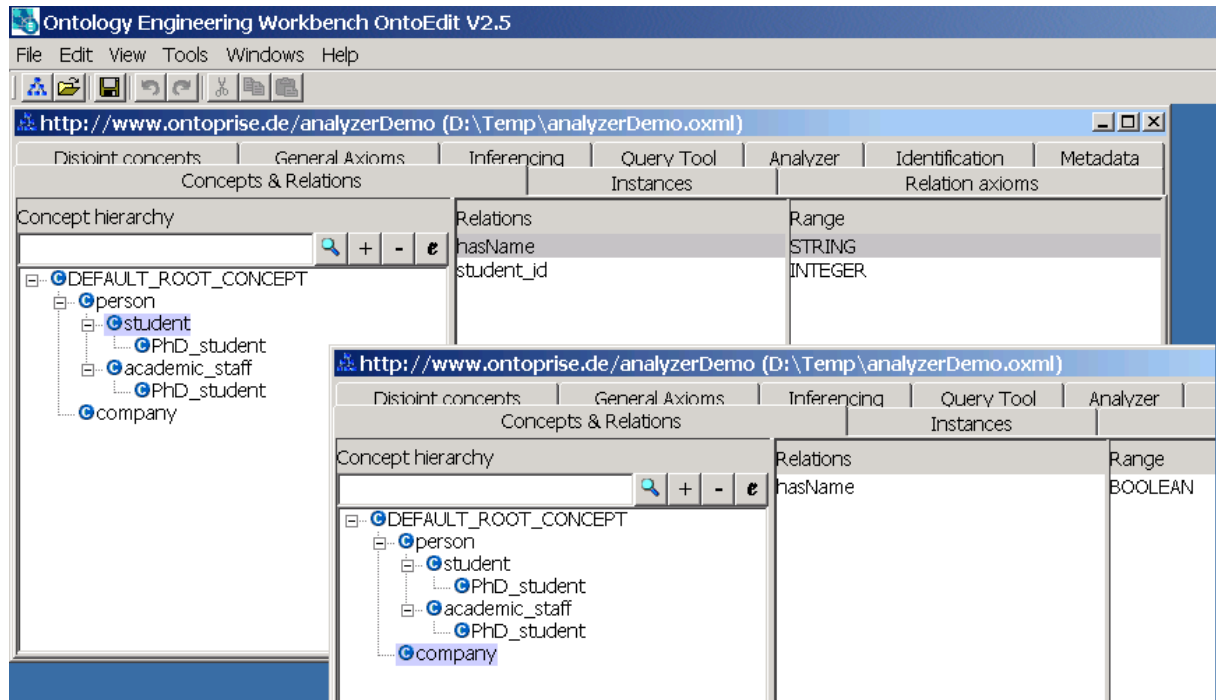


Figure 1: Example ontology in OntoEdit

We illustrate the application of the two rules from above with OntoAnalyser by giving an example. Figure 1 shows a screenshot of OntoEdit with an example ontology. It is noteworthy that (i) the concept `PhD_student` is a subconcept of `student` as well as of `academic_staff`, i.e. there exists multiple inheritance, and (ii) that the two concepts `student` and `company` both have a relation¹ `hasName`, but each one with a different range, viz. `STRING` and `BOOLEAN`.

¹ Sometimes relations with predefined range types like “STRING” and “BOOLEAN” are called “attributes”.

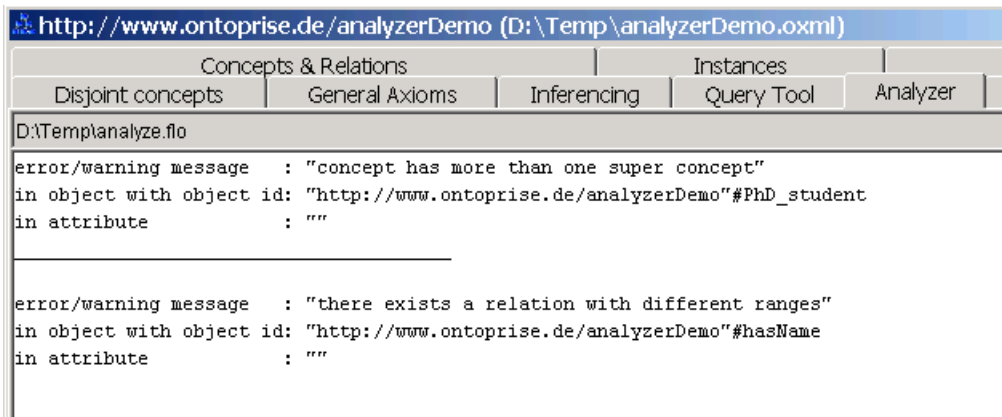


Figure 2: Evaluation results of OntoAnalyser for example ontology

The two rules presented above were now applied to the example ontology shown in Figure 1. The results of the evaluation by OntoAnalyser are shown in Figure 2. Firstly, the multiple inheritance of the concept `PhD_student` resulted in an error message “concept has more than one super concept”. Secondly, the existence of the relation `hasName` with different ranges resulted in an error message “there exists a relation with different ranges”.

OntoAnalyser is realized as a plugin in OntoEdit. Therefore it is well integrated into the ontology engineering environment itself. An ontology may be developed in OntoEdit and during this process it may be on the fly checked using OntoAnalyser. Thus, no export and import and no transformation of the ontology is necessary to perform evaluations. For importing ontologies we would need to check e.g. the interoperability of our tool and the tool that was used for the development – therefore an evaluation of another criteria of our framework would be necessary.

For the future we plan to develop standard rule packages for evaluation of various purposes to support efficient and effective engineering of ontologies and to improve the quality of ontologies at the same time.

4.3.3 OntoGenerator

The second implementation OntoGenerator supports performance tests (“stress tests”) of ontology based tools. It creates “synthetic” ontologies which are not intended to represent a domain of interest, but rather to fulfil certain technical parameters like e.g. a certain number of concepts and instances or certain kinds of rules.

OntoGenerator originally was implemented to support the optimisation of the Ontobroker rule evaluation for F-Logic rules (e.g. the ones presented in the previous section). The performance of the rule evaluation strongly depends on the sequence the rule bodies are evaluated in each rule. An example from our optimisation experiments shows the significance of this sequence. E.g., for one example the **optimal sequence** produced **20.000** intermediate evaluation results while a **bad sequence** produced **38.000.000** intermediate evaluation results. Our optimiser searches for the best sequence by using a genetic algorithm. To test such optimisation strategies it is necessary to test it for numerous different ontologies and different rule sets with special properties. Therefore we implemented OntoGenerator that is able to produce “synthetic ontologies” on the fly according to predefined parameters.

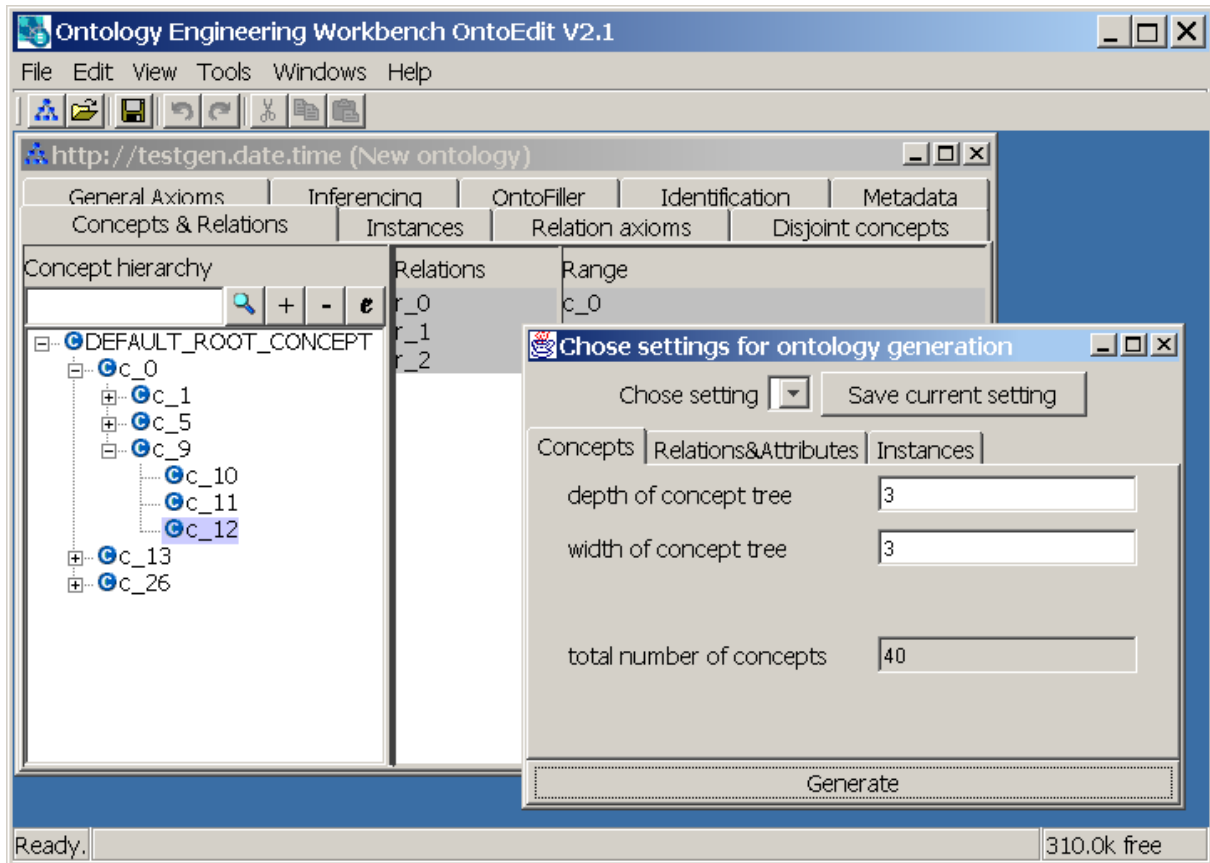


Figure 3: Generating synthetic ontologies with OntoGenerator

Figure 3 shows a generated ontology by OntoGenerator. Currently one might predefine the following parameters for generation of ontologies: (i) depth of the concept tree, (ii) width of the concept tree, (iii) total number of relations, (iv) total number of attributes and (v) total number of instances. In detail the parameters have the following effects during the generation: (i) defines how deep the generated tree will be (here: 3 levels deep), (ii) defines the number of children per generated concept (here 3 children), (iii) and (iv) define the total numbers of relations and attributes that are currently attached randomly to concepts and (v) defines the number of instances created, the concepts are chosen randomly for instantiation.

For the future we plan to implement statistical measures for the selection of concepts, e.g. used during attachment of relations and attributes and during creation of instances, given the fact that it is important at what level a relation is attached. Due to the inheritance mechanism a relation attached at a higher level will result in a higher amount of inherited relations at the lower levels of the concept hierarchy. Furthermore we will extend OntoGenerator by a rule generator, thus exploring e.g. the following characteristics:

- **Depth of the rule tree:** how long is the chain where one rule depends on the other.
- **Cyclicity of rules:** the dependence of rules may be cyclic.
- **Length of rule cycles:** how long are the dependence cycles.
- **Complexity of rule bodies:** how complex are the object formulas.
- **Transitivity:** how many transitive rules are included.

4.4 Comparison of the implementations against the evaluation framework

All described tools cover different aspects in our evaluation framework. In the following Table 3 we evaluate the reference implementations according to the descriptions in the previous section against our evaluation framework. It is shown that each tool targets different aspects of the framework.

Table 3: Evaluation of the implementations against the evaluation framework

	OntoAnalyser	OntoGenerator
Ontology properties		
Language conformity (Syntax)	X	
Consistency (Semantics)	X	
Technology properties		
Interoperability (e.g. Semantics)		
Turn around ability		
Performance		X
Memory allocation		X
Scalability		X
Integration into frameworks		
Connectors and interfaces		

4.5 Related Work

Though there exists seminal research for evaluation of ontologies (e.g. [5.1], [5.2]), a recent position statement clarifies the current state of the art [5.3]: “Evaluation of ontologies is rarely discussed in the literature. ... There is a dearth of detailed reports on evaluation and lack of tools to assist ‘ontology testers’. Evaluation during development could be assisted by guidelines, integrated environments with support for certain types or errors (type-checking, referential integrity, etc.). ... Given the sheer size and ever-increasing number of ontologies there is an urgent need for evaluation upon deployment in applications. This, preferably, has to be executable.”

4.6 Conclusions

The growing interest in the semantic web attracts beside academia more and more industrial partners. Ontologies and related technologies like development tools as a core component for the semantic web need to fulfil strong requirements to meet the demands of industrial partners. We presented a framework for evaluation of ontologies and related technologies that aims at ensuring necessary levels of quality. Our framework consists of two main areas. We distinguish between (i) ontology properties, i.e. language conformity and consistency, and (ii) technology properties, i.e. interoperability, turn around ability, performance, memory allocation, scalability, integration into frameworks and, last but not least, connectors and interfaces. We presented reference implementations of the framework, OntoAnalyser and OntoGenerator. Each addresses different aspects of the evaluation criteria, which was shown by evaluating the implementations against the framework. OntoAnalyser focuses on evaluation of ontology properties, in particular language conformity and consistency. OntoGenerator focuses on evaluation of ontology based tools, in particular performance and scalability.

For the future we envision a growing interest in evaluation of ontologies and in particular in evaluation of technologies related to ontologies. Real life implementations of new technologies require benchmarks and standardization efforts, e.g. like in the database community. Therefore we encourage the semantic web community to enforce their research efforts by developing further standard criteria, e.g. benchmarks and certifications, and tools that implement these criteria to evaluate ontologies and related technologies. Currently we see two encouraging starting points: (i) the EU IST OntoWeb thematic network initiative that creates a roadmap for the future of the ontology and semantic web community and (ii) a planned workshop at the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2002) which is called “EON 2002: Evaluation of Ontology-based Tools”.

Acknowledgements: Research for this paper was partially funded by EU in the project IST-1999-10132 On-To-Knowledge.

4.7 References

- [1.1] T. Berners-Lee, J. Hendler and O. Lassila. A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 2002, cf. <http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html>.
- [1.2] EU IST-1999-10132 project “On-To-Knowledge: Content-driven knowledge management tools through evolving ontologies”, cf. <http://www.ontoknowledge.org>.
- [1.3] US DARPA project “DARPA Agent Markup Language (DAML)”, cf. <http://www.daml.org>.
- [1.4] EU IST-2000-29243 thematic network “OntoWeb: Ontology-based Information Exchange for Knowledge Management and Electronic Commerce”, cf. <http://www.ontoweb.org>.
- [3.1] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer and D. Wenke. OntoEdit: Collaborative Ontology Engineering for the Semantic Web. In *Proceedings of the International Semantic Web Conference 2002 (ISWC 2002)*, June 9-12 2002, Sardinia, Italia.
- [3.2] S. Handschuh. Ontoplugins – a flexible component framework. Technical report, University of Karlsruhe, May 2001.
- [3.3] S. Staab, H.-P. Schnurr, R. Studer, and Y. Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems, Special Issue on Knowledge Management*, 16(1), January/February 2001.
- [3.4] N. Fridman Noy, R. Ferguson, and M. Musen. The knowledge model of Protégé2000: Combining interoperability and flexibility. In *Proceedings of EKAW 2000*, NCS 1937, pages 17–32. Springer, 2000.
- [3.5] J.C. Arprez, O. Corcho, M. Fernandez-Lopez, and A. Gomez-Perez. WebODE: a scalable workbench for ontological engineering. In *Proceedings of the First International Conference on Knowledge Capture (K-CAP) Oct. 21-23, 2001, Victoria, B.C., Canada, 2001*.
- [3.6] M. Kifer, G. Lausen, J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages, *Journal of the ACM*, 42, 1995.
- [3.7] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In R. Meersman et al., editor, *Database Semantics: Semantic Issues in Multimedia Systems*. Kluwer Academic, 1999.
- [5.1] A. Gómez-Pérez. A framework to verify knowledge sharing technology. *Expert Systems with Application*, 11(4):519-529, 1996.
- [5.2] Grüninger, M., Fox, M.S. The Role of Competency Questions in Enterprise Engineering. In *IFIP WG 5.7, Workshop Benchmarking. Theory and Practice*, Trondheim/Norway, 1994.
- [5.3] Y. Kalfoglou. Evaluating ontologies during deployment in applications. Position statement at the OntoWeb 2 meeting 07-12-2002, Amsterdam, The Netherlands, cf. [1.4].