
# D3.1.2 Incremental Ontology Evolution - Evaluation

**Peter Haase and York Sure**
**(Institute AIFB, University of Karlsruhe)**

**Abstract.**
EU-IST Integrated Project (IP) IST-2003-506826 SEKT
Deliverable D3.1.2 (WP3.1)
In this deliverable addresses we present the evaluation of our work performed in the task 'T3.1 Incremental Ontology Evolution'. There are two aspects to this evaluation: First, we introduce the notion of ontology evaluation into our framework of incremental ontology evolution. Second, we present evaluation results of our methods in terms of effectiveness, performance and applicability.
Keyword list: Ontology Evolution, Ontology Evaluation, Ontology Management

| | |
|---|---|
| **Document Id.** | SEKT/2005/D3.1.2/v1.0 |
| **Project** | SEKT EU-IST-2003-506826 |
| **Date** | November 2, 2005 |
| **Distribution** | public |

# SEKT Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2003-506826.

**British Telecommunications plc.**
Orion 5/12, Adastral Park
Ipswich IP5 3RE
UK
Tel: +44 1473 609583, Fax: +44 1473 609832
Contact person: John Davies
E-mail: john.nj.davies@bt.com

**Jozef Stefan Institute**
Jamova 39
1000 Ljubljana
Slovenia
Tel: +386 1 4773 778, Fax: +386 1 4251 038
Contact person: Marko Grobelnik
E-mail: marko.grobelnik@ijs.si

**University of Sheffield**
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Tel: +44 114 222 1891, Fax: +44 114 222 1810
Contact person: Hamish Cunningham
E-mail: hamish@dcs.shef.ac.uk

**Intelligent Software Components S.A.**
Pedro de Valdivia, 10
28006 Madrid
Spain
Tel: +34 913 349 797, Fax: +49 34 913 349 799
Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

**Ontoprise GmbH**
Amalienbadstr. 36
76227 Karlsruhe
Germany
Tel: +49 721 50980912, Fax: +49 721 50980911
Contact person: Hans-Peter Schnurr
E-mail: schnurr@ontoprise.de

**Vrije Universiteit Amsterdam (VUA)**
Department of Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
Tel: +31 20 444 7731, Fax: +31 84 221 4294
Contact person: Frank van Harmelen
E-mail: frank.van.harmelen@cs.vu.nl

**Empolis GmbH**
Europaallee 10
67657 Kaiserslautern
Germany
Tel: +49 631 303 5540, Fax: +49 631 303 5507
Contact person: Ralph Traphöner
E-mail: ralph.traphoener@empolis.com

**University of Karlsruhe**, Institute AIFB
Englerstr. 28
D-76128 Karlsruhe
Germany
Tel: +49 721 608 6592, Fax: +49 721 608 6580
Contact person: York Sure
E-mail: sure@aifb.uni-karlsruhe.de

**University of Innsbruck**
Institute of Computer Science
Technikerstraße 13
6020 Innsbruck
Austria
Tel: +43 512 507 6475, Fax: +43 512 507 9872
Contact person: Jos de Bruijn
E-mail: jos.de-bruijn@deri.ie

**Kea-pro GmbH**
Tal
6464 Springen
Switzerland
Tel: +41 41 879 00, Fax: 41 41 879 00 13
Contact person: Tom Bösser
E-mail: tb@keapro.net

**Sirma AI EAD, Ontotext Lab**
135 Tsarigradsko Shose
Sofia 1784
Bulgaria
Tel: +359 2 9768 303, Fax: +359 2 9768 311
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

**Universitat Autonoma de Barcelona**
Edifici B, Campus de la UAB
08193 Bellaterra (Cerdanyola del Vallès)
Barcelona
Spain
Tel: +34 93 581 22 35, Fax: +34 93 581 29 88
Contact person: Pompeu Casanovas Romeu
E-mail: pompeu.casanovas@uab.es

# Executive Summary

This deliverable is the successor of D3.1.1, where we have presented methods and tools for the ontology management infrastructure to support incremental ontology evolution. This deliverable complements the prior one with an evaluation in two senses: First, we extend our formalisms with support for formal *ontology evaluation*. Second, we apply our methods in a practical setting of the BT Digital Library case study and present *evaluation results* in terms of performance and effectiveness.

In our formalism we introduce the notion of an ontology evaluation function that allows to assess the quality of an ontology for a given context. Based on this ontology evaluation function, the task of ontology evolution can be formalized as a search for an ontology that maximizes the evaluation function. We incoporate the existing methods for change discovery developed in tasks T3.2 and T.3 into the framework, considering the usage-driven and a data-driven approach, respectively. We demonstrate the application using a scenario from the BT DL case study.

The evaluation of the methods covers aspects such as performance of the approach of consistent ontology evolution and the underlying reasoner, extensibility and effectiveness.

# Contents

# Chapter 1

# Introduction

## 1.1 The SEKT Big Picture

This report is part of the work performed in workpackage (WP) 3 on "Ontology and Metadata Management". As shown in Figure 1.1 this work belongs to the central part of the research and development WPs in SEKT. Quite naturally it is closely connected with Ontology Generation and Metadata Generation, in particular we will integrate parts of their technologies. We are focusing on how to manage ontologies (and related metadata) and their evolution over time. As part of WP3.1, we provide a basic infrastructure for ontology management. We extend this in WP3.2 and WP3.3 with functionalities for data-driven change discovery and usage tracking, i.e. with means to adapt ontologies according to underlying domain knowledge in form of documents on the one hand and the usage of ontologies in applications by users on the other hand. As part of WP7 Methodology we closely collaborate with the case study partners to apply our technologies within the case studies (see e.g. [EGH+04b, EGH+04a, STV+05] and following ones).

The aspect of ontology evaluation has also been addressed as part of task T1.6 Ontology Evaluation. The work is currently being aligned and combined as part of a common framework on ontology evaluation [BGH+05]. The efforts in these tasks are complementary in the sense that T1.6 develops specific methods for ontology evaluation, while T3.1 uses such methods for the purpose of ontology evolution.

## 1.2 Motivation

What is a *good* ontology? This question, though quite crucial from a practical perspective for developing ontology-based applications, cannot be answered clearly at the moment due to the fact that there exists no commonly agreed definition of what 'good' means for ontologies. How to make an ontology *better*? Even if you have defined what 'good' means, the question remains whether you can still improve an ontology so that it becomes

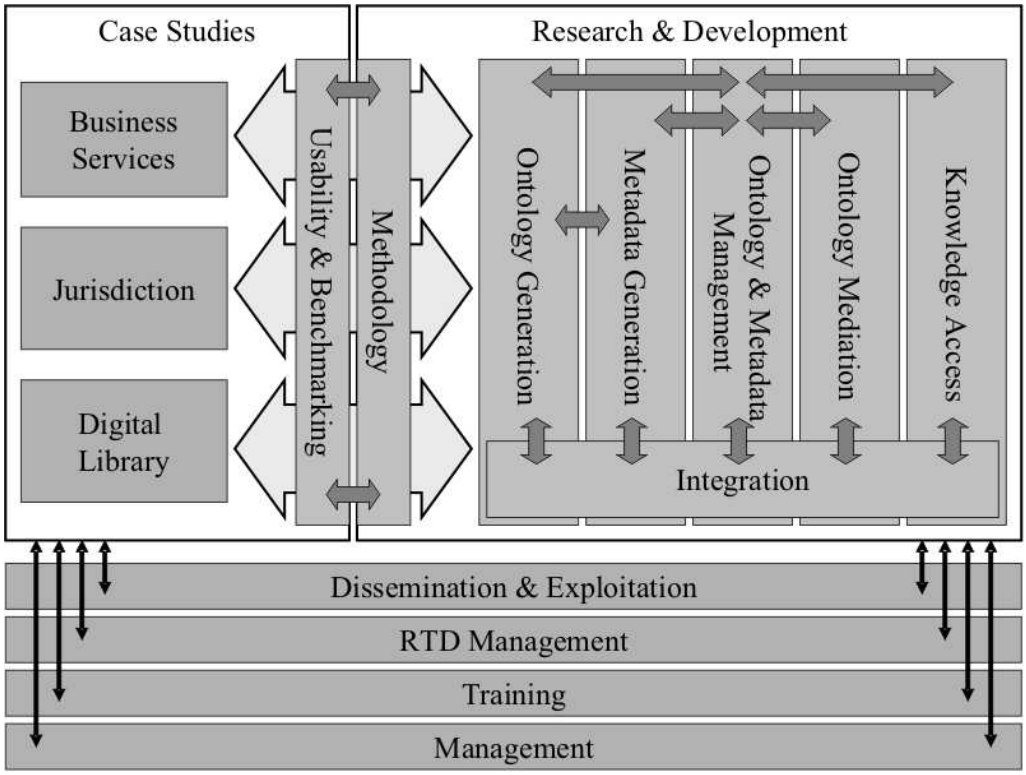



Figure 1.1: The SEKT Big Picture

a 'better' one and what has to be done to make it better.

To approach potential answers for the above questions we briefly re-visit the (currently hot) research topics of ontology evaluation and ontology evolution and how they are embedded into ontology engineering methodologies. Typical criteria to evaluate ontologies are focused on structural aspects and logical consistency [GP04]. E.g., OntoClean [GW04] provides means to evaluate the taxonomic relationships of an ontology based on philosophical notions. Evolution of ontologies deals with the "timely adaptation of an ontology to the arisen changes and the consistent propagation of these changes to dependent artefacts" [Sto04]. Both, ontology evolution and ontology evaluation, are typically seen as parts of an ontology lifecycle. The most well-known methodologies for ontology engineering [FLGPSS, SS02b, TPSS05] include distinct steps for evaluation and evolution which are performed in iterative cycles.

In this work we explore boundaries of current approaches for ontology evaluation and ontology evolution. Particularly we argue that they cannot be seen as separate phases of an ontology lifecycle. In line with current methodologies for ontology engineering we propose iterative cycles during ontology engineering, but in contrast to them we take evaluation considerations into account *during* and even *before* evolution. Our approach facilitates in general the automation of ontology engineering, which seems a crucial step

for making ontology engineering less time-consuming and thus more attractive for industrial users. We would like to emphasize that our approach works for manually engineered as well as for learned ontologies (e.g. from texts).

From our understanding, the question "What is a good ontology?" can only be answered for given contexts. Our main contribution therefore is not to provide a generic answer to the initial question, but rather to provide methods for defining a context for an ontology, methods for defining what 'good ontology' means for a given context, and methods for improving good ontologies. In a nutshell we provide answers to the following questions:

- How can you define the *context* of an ontology?
- How can you define *good ontology* given a certain context?
- How can you make good ontologies *better* given a certain context?

After providing a formalism to answer these questions, we apply the formalism for a specific task of ontology evolution, namely that of incremental ontology learning. Here, the context is given by a text corpus describing a domain of interest. The value of a learned ontology is on the one hand defined by how well it reflects the domain of interest, and on the other hand by its usefulness for question answering, which requires its logical consistency.

## 1.3 Related Work

The work presented here weaves together the fields of contextualisation, evaluation and evolution of ontologies. They have been partially brought together already in ontology lifecyle models like the On-To-Knowledge methodology [SS02b], Methontology [FLGPSS], DILIGENT [TPSS05] and others. All these methodologies and ontology lifecycle models offer rather generic frameworks for locating several ontology techniques in the lifecycle of an ontology. In this paper we focus instead on the explicit and novel combination of context, evaluation and evolution, in order to enhance all three of them.

There has been a lot of work in the respective disciplines. We point to a number of works and describe how they can be related to the approach presented here.

**Context.** Early models for context representation come from Artificial Intelligence, where they are are mostly represented in a logical notation. Functions and predicates are given for each of the aspects (dimensions) of the environment [GMF04]. The goal in our work however is not to incorporate context explicitly into the logic, but instead to model contextual information as annotations of the ontology elements.

There exists a huge body of work on the broader theme of probabilistic and uncertainty aspects of knowledge representation and reasoning [Bac90]. The specifics of ontology languages are a research issue right now, particularly involving Bayesian networks (see e.g. [DP04]).

**Evaluation.** The work on ontology evaluation has increased significantly in the last years. Approaches like [PSK05] or [GP04] deal with fundamental aspects of ontologies like syntactic correctness or logical consistency, that are rather a necessary precondition for the usage of the suggested framework. But many evaluation methodologies deal with the relation between the intended conceptualization and the actual specification that is the ontology. [Gua98] says that most ontologies due to a weak axiomatization allow unintended models, and describes two ways to get closer to a conceptualization: developing a richer axiomatization or adopting a richer domain and/or a richer set of relevant conceptual relations. Based on this model of ontology evaluation the authors later introduce the OntoClean methodology [GW04], which is used within this paper as well, in order to evaluate concept hierarchies based on philosophical notions annotated to the concepts.

In order for an evaluation approach to be amenable to our framework, it has to fulfil two conditions: first, it needs to be able to answer the question which of two given ontologies is the better one. Second, it needs to offer a way to come up automatically with sensible ontology change operations in order to automatically evolve the ontology. Although most evaluation methodologies only answer the first question, many of them can be extended to offer answers for the second as well: in [BADW04] we find a methodology to compare a given ontology with a text and calculate a similarity. Ontology change operations adding further keywords from the text, or removing superfluous ones, may be used in order to use the whole framework with the evaluation methodology in [BADW04]. [MS02] analyses the similarity to a gold standard ontology in order to suggest changes to the evolving ontology. As described in [HHSTS05] the gold standard could also be an ontology aggregated from several personalized ontologies.

Ontology evaluation methodologies that need a high amount of user input, or even input from a community of users, as for example [Sup05], are not applicable within this framework, as the search for the fitting ontology change operation would ultimately involve comparing a high number of ontologies, which can't be sensibly done manually. On the other hand, methodologies that offer a metric-based approach, like [BJSSA04], [FBGL98] or [LTGP04] fit easily into the framework, as long as the metric maps reliably to quality as defined by the user.

**Evolution.** Ontology evolution is a central task in ontology management that has been addressed for example in [KN03] and [SMMS02]. In [SMMS02] the authors identify a possible six-phase evolution process: (1) change capturing, (2) change representation, (3) semantics of change, (4) change implementation, (5) change propagation, and (6) change validation. Our work is based on this evolution process. The link between the evaluation

methodology and the evolution process is that the evaluation provides as with an ontology change operation, thus covering phases (1)–(3) of the given process. The evolution framework then applies the other phases and we are ready to evaluate the resulting ontology anew. One approach for *usage-driven change discovery* in ontology management systems has been explored in [SS02a], where the user's behaviour during the knowledge providing and searching phase is analysed. [SHG03] describes a tool for guiding ontology managers through the modification of an ontology based on the analysis of end-users' interactions with ontology-based applications, which are tracked in a usage-log.

## 1.4 Application Scenario

In this section we present a typical application scenario to motivate and illustrate our approach. Intelligent search over document corpora in Digital Libraries is one application scenario that shows the immediate benefit of the ability to reason over ontologies automatically learned from text. While search in Digital Libraries nowadays is restricted to structured queries against the bibliographic metadata (author, title, etc.) and to unstructured keyword-based queries over the full text documents, complex queries that involve reasoning over the knowledge present in the documents are not possible.

This application scenario is the subject of the BT Digital Library case study. In the BT Digital Library so-called information spaces have been created for domains known to be of interest to people in the company to structure the contents of journals in the library. One of the key elements of the case study is to use ontologies to enhance the knowledge access to the Digital Library.

Interests of people change over time, as does the content of the digital library. The ontologies need to adapt to those changes in order to stay current and useful.

Figure 1.2 offers an overview of the main building blocks to support the evolution of ontologies in the Digital Library, which we explain in more detail in the following.

Users of the Digital Library interact with the *knowledge portal* as the user interface. The knowledge portal allows the user to search the library's contents as it presents the content in an organized way. The knowledge portal may also provide the user with information in a proactive manner, e.g. by alerts, notification, etc. The typical user primarily consumes knowledge from the Digital Library. He/she uses the Digital Library to fulfil a particular information need. However, an advanced user may also contribute to the Digital Library, either by contributing content or by organizing the existing content, providing metadata, etc.

*Ontologies* are the basis for rich, semantic descriptions of the content in the Digital Library. Here we can identify two main modules of the ontology: The *application ontology* describes different generic aspects of bibliographic metadata (such as *author* or *creation data*) and is valid across various bibliographic sources.

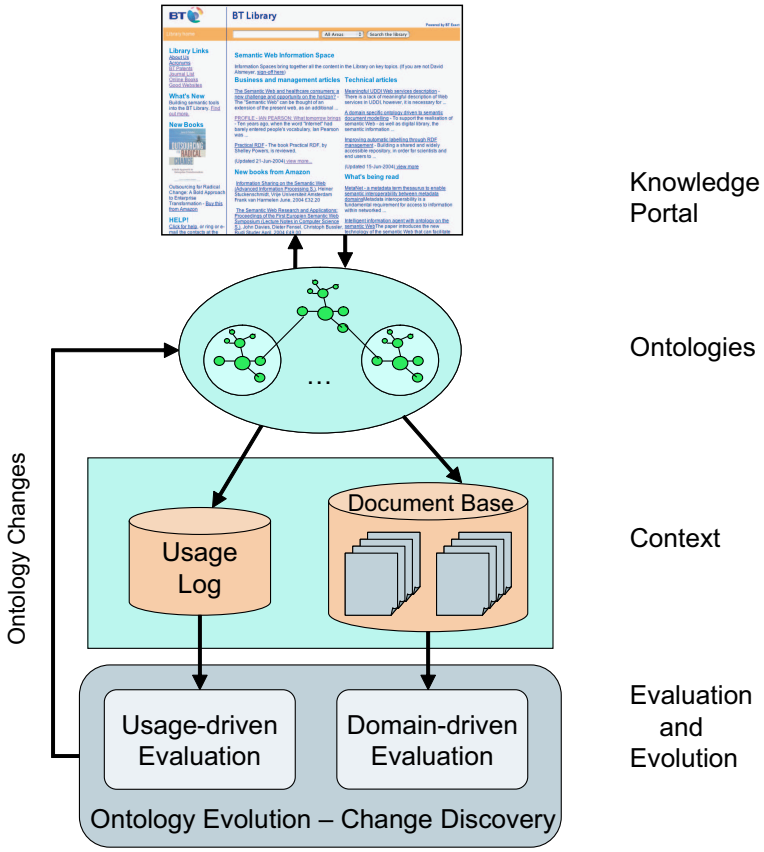



Figure 1.2: Logical Architecture

*Domain ontologies* describe aspects that are specific to particular domains and are used as a conceptual backbone for structuring the domain information provided in the information spaces. Such a domain ontology typically comprises conceptual relations, such as a topic hierarchy, but also richer taxonomic and non-taxonomic relations.

*Personalisation of ontologies* further tailors the interaction of the user with the knowledge portal to a particular user. Thus a view on the domain ontologies that reflect the interests of the user at a certain time can be represented. This allows to better deal with the large size of the data by selecting the relevant parts of the ontology.

The ontologies are used for various purposes: First of all, the documents in the document base are annotated and classified according to the ontology. This ontological metadata can then be exploited for advanced knowledge access, including navigation, browsing, and semantic searches. Of particular interest for this deliverable is the use of learned ontologies to support advanced question answering. An important aspect here is that the learned ontology needs to best reflect the domain of interest as well as it needs to be consistent to return meaningful query results.

Finally, the ontology can be used for the visualization of results, e.g. for displaying the relationships between information objects.

While the application ontology can be assumed to be fairly static and applicable across information spaces, the domain ontologies (including personalized ontologies) must be continuously adapted to the changing domain and user needs.

The *document base* comprises a corpus of documents. The content of the document base typically is not static, but changes over time: New documents come in, but also documents may be removed from the the document base. The document base holds documents from one or more domains. *Information Spaces* are the logical units to organize a collection of documents according to domains.

The interaction of the knowledge worker with the knowledge portal is recorded in a *usage log*. It is of particular interest how the ontology has been used in the interaction, i.e. which elements have been queried, which paths have been navigated, etc. By tracking users' interactions with the application in a log file, it is possible to collect useful information that can be used to assess what the main interests of the users are. In this way, we are able to obtain implicit feedback and to extract needs for changes to the ontology to improve the interaction with the application.

Both the usage log information as well as the document corpus for a particular domain establish a *context* for the ontology. This contextual information is a valuable input for the *evaluation* of the ontology. Specifically, we can define the value of the ontology as how good it reflects the interests of the user and allows the user to obtain the relevant information, and how well it reflects the domain of the document corpus.

Based on the evaluation of ontologies, we can guide the *evolution* of the ontology by discovering and applying potentially useful changes that increase the value of the ontology, i.e. we generate changes to the ontology to improve the interaction with the Digital Library. While the recommendations for ontology changes are generated in an automated manner, they typically will be approved by a knowledge engineer before the actual application.

## 1.5 Overview of the Deliverable

In this Chapter we have motivated the work of this deliverable and situated it within the context of the project. A real-world application scenario from the BT DL case study described in Section 1.4 serves as a base for concrete application examples of our methods in the subsequent chapters.

In Chapter 2 we present our approach to ontology evolution based on ontology evaluation. This chapter is structured as follows. In Section 2.1 we provide the formalization of the context model as well as further foundations including the ontology model underlying the context model. We tackle the notion of 'good ontology' based on the definition of an evaluation function, and show how the evaluation function can be used for making an ontology better. We then apply the approach to dealing with the usage context in Section 2.2, and domain context in Section 2.3.

In Chapter 3 we present evaluation results of our methods. We perform the evaluation by applying our framework to the task of incremental ontology learning and using real-life data from the BT Digital Library case study. We conclude with an outlook to future work in Chapter 4.

# Chapter 2

# Ontology Evaluation for Ontology Evolution

## 2.1 Ontology Evaluation for Ontology Evolution – Overview of the Approach

In this section we provide an overview of our approach to ontology evolution based on ontology evaluation. We first lay the foundations to capture an ontology together with its context. Therefore, we introduce a formal ontology model, based on the OWL ontology language [HPSvH03]. We then formalize how to capture contextual information, i.e. information outside of the ontology itself. We then formalize the notion of an ontology evaluation function, which serves as a basis to define what a good ontology is. Finally, we present the problem of discovering changes for ontology evolution in terms of an optimization of the ontology evaluation function. In the subsequent Sections 2.2 and 2.3 we will fill these rather abstract model with life by applying it to our application scenario.

### 2.1.1 OWL Ontology Model

In continuation of our work in D3.1.1 we base our work on the OWL ontology model. Today, many different ontology languages exist. Although our definitions of context generally would be compatible with any such language, we base our definitions on the W3C standardized OWL ontology language which we briefly summarize here.

The OWL ontology language is based on a family of description logics languages [HPSvH03]. As usual in description logics, an OWL ontology is built over a vocabulary that consists a set of concept names $N_C$, sets of abstract and concrete individual names $N_{I_a}$ and $N_{I_c}$, respectively, and sets of abstract and concrete role names $N_{R_a}$ and $N_{R_c}$, respectively. The set of OWL DL *concepts* is defined by the following syntactic rules, where $A$ is an atomic concept, $R$ is an abstract role, $S$ is an abstract simple role (a role

not having transitive subroles), $T_{(i)}$ are concrete roles, $d$ is a concrete domain predicate, $a_i$ and $c_i$ are abstract and concrete individuals, respectively, and $n$ is a non-negative integer:

$$\begin{aligned} C &\rightarrow A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C \mid \geq n\, S \mid \leq n\, S \mid \{a_1, \ldots, a_n\} \mid \\ &\quad \mid \geq n\, T \mid \leq n\, T \mid \exists T_1, \ldots, T_n.D \mid \forall T_1, \ldots, T_n.D \\ D &\rightarrow d \mid \{c_1, \ldots, c_n\} \end{aligned}$$

An ontology then is a finite set of of the form[1]:

- concept inclusion axioms $C_1 \sqsubseteq C_2$, stating that the concept $C_1$ is a subconcept of the concept $C_2$,
- transitivity axioms $\mathsf{Trans}(R)$, stating that the abstract role $R$ is transitive,
- role inclusion axioms $R \sqsubseteq S$ ($T \sqsubseteq U$) stating that the abstract role $R$ (or concrete role $T$) is a subrole of the abstract role $S$ (or concrete role $U$).
- concept assertions $C(a)$ stating that the abstract individual $a$ is in the extension of the concept $C$,
- abstract role assertions $R(a, b)$ and $T(a, c)$ stating that the abstract individuals $a$, $b$ (or $a$, $c$) are in the extension of the role $R$ ($T$),
- concrete role assertions $T(a, c)$ stating that the abstract individual $a$ and the concrete individual $c$ are in the extension of the concrete role $T$,
- individual (in)equalities $a \approx b$, and $a \not\approx b$, respectively, stating that $a$ and $b$ denote the same (different) individuals.

In the following, we denote the set of all ontology elements, i.e. both axioms and symbol names, with $N$. We denote the set of all possible ontologies with $\mathcal{O}$.

### 2.1.2 Context Model or "How can you define the *context* of an ontology?"

Our ontology model so far describes the actual state of an ontology as an isolated entity. Once we enter the more dynamic scenario of ontology evolution, it makes sense to consider contextual information about the ontology. The term "context" has many different connotations depending on the field it is being used in. In general, the context of an entity includes the circumstances and conditions which "surround" it. [Dey01] has defined context as: *Context is any information that can be used to characterize the situation of*

[1] For the direct model-theoretic semantics we refer the reader to [HST00].

*an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.* Applied to ontologies this means that we consider any information that is external to the ontology itself, but relevant to the interaction between the user and the ontology-based application. Typical examples of contextual information include – as we will later elaborate – usage log information, provenance, information about trust, confidence and certainty, etc.

To capture such contextual information about ontologies, we introduce the notion of *ontology rating annotations*, which are used to relate the context to the elements of the ontology:

**Definition 1** *Let $N$ denote the set of all possible ontology elements and $\mathcal{X}$ be a suitable representation of a context space, then an ontology rating annotation is a partial function $r : N \rightarrow \mathcal{X}$.*

Please note that the ontology elements to be rated – according to the definition of the ontology model – can be either axioms or entities. The context space $\mathcal{X}$ is kept as general as possible in this definition, to allow to attach essentially arbitrary information external to the ontology. We will instantiate $\mathcal{X}$ for the specific contexts we consider.

The user may easily define new rating annotations for whatever need arises. The contextual information is the basis for the following evaluation function, but may also be used outside of evaluations. In Sections 2.2 and 2.3 we will introduce the two particular forms of contexts that will be used to exemplify our approach, *usage context* and *domain context*. These two forms of context are of particular importance, as they model how users interact with the ontology, and how well the ontology reflects a particular domain, respectively. They thus provide useful indications to the value of an ontology in this scenario.

### 2.1.3 Ontology Evaluation or "How can you define a *good ontology* given a certain context?"

Good ontologies are ontologies that serve their purpose. In order to be able to define what a "good" ontology for a particular context is, we need to be able to measure the quality of the ontology with respect to a given set of criteria. For our framework, we do not commit to a certain set of evaluation methodologies, but rather allow the user of this framework to choose the evaluation approach she deems best fitting for her task. We will detail some approaches in this section.

The relationship between intended models and specification is being captured by the context, as described in the former section, and is measured by an *ontology evaluation function*.

**Definition 2** *Let $\mathcal{O}$ be the set of possible ontologies, then an ontology evaluation function $e$ is a function $e : \mathcal{O} \rightarrow [0, 1]$.*

Effectively, the evaluation function provides a total order over the space of possible ontologies and thus allows to compare given ontologies. Here it is important to note that the evaluation function can take the rating annotations into account and thus provides an evaluation measure with regards to a given context. The intuitive reading of the evaluation function is that a value of $1$ indicates the "perfect" ontology, whereas a value of $0$ is the "worst case".

By using the unit interval for the representation of the value of an ontology, we obtain the immediate benefit of being able to combine different quality criteria, e.g. using a weighted average of different ontology evaluation functions.

### 2.1.4 Ontology Evolution or "How can you make good ontologies *better* given a certain context?"

Ontology evolution is timely adaptation of the ontology to changes and the consistent management of these changes. In particular, we need to account for changes in the context of the ontology, i.e. to evolve the ontology with its context. To operationalize this, we first formalize the notion of ontology changes. Based on the ontology evaluation function we are then able to determine whether a particular change leads to an improved ontology. The actual challenge then is the discovery of potentially useful changes. In Sections 2.2 and 2.3 we present such mechanisms for change discovery for the usage-driven and domain-driven evolution of our running scenario.

**Definition 3 (Ontology Change Operation)** *An* ontology change operation $oco$ *is a function*

$$oco : \mathcal{O} \rightarrow \mathcal{O}$$

With OCO we denote the set of all possible ontology change operations. For the ontology model defined above, we allow the atomic change operations of adding and removing axioms, which we denote with $\alpha^+$ and $\alpha^-$, respectively. Complex ontology change operations can be expressed as a sequence of atomic ontology change operations. The semantics of the sequence is the chaining of the corresponding functions: For some atomic change operations $\text{oco}_1, ..., \text{oco}_n$ we can define $\text{oco}_{\text{complex}} = \text{oco}_n \circ ... \circ \text{oco}_1 = \text{oco}_n(...\text{oco}_1)$.

#### Change Discovery

Based on the ontology evaluation function, we can now measure whether a particular change to an ontology leads to an "improvement" of the ontology for the given context. As this context changes over time, we can regard ontology evolution as the adaptation to

the changing context by discovering and applying changes to the ontology. Essentially, the goal is to discover changes that lead to a maximized evaluation function, i.e. the ideal ontology for the particular context:

**Definition 4** *For a given ontology $O$ and an evaluation function $e$, we can define the problem of change discovery as an optimization problem:*

$$max_{oco \in OCO}\ e(oco(O))$$

Having the problem stated as an optimization problem opens the door to applying established optimization techniques to find the "best" ontology with respect to the evaluation function. In general, it will be hard to determine the optimal ontology that maximizes the evaluation function, as one theoretically would need to search the entire space of possible consistent ontologies. However, in most cases it is not necessary to prove the optimality of an obtained solution. Instead it is possible to exploit heuristics-based techniques to obtain a "fairly" optimal ontology.

## 2.2 Usage Context

The first type of context we consider is the *usage context*. The intention of modelling usage context is to capture the users' behavioural patterns, which can in turn be used to assess the effectiveness in the interaction with the ontology, to identify important parts, but also weaknesses of the ontology. We here use the rating annotations to indicate the importance of particular elements. The methods presented in this section have - to a large extent - already been presented in [HS05b]. We here repeat the main ideas and rephrase them in terms of our framework.

Generally, we can distinguish between *explicit* and *implicit* user feedback from usage information. We talk about explicit feedback if we allow that a user (i) can express way how important a certain ontology element is for him, and that he (ii) can explicitly express negative ratings for ontology elements that he does not want to be part of his ontology.

**Example 1 (Explicit Usage Rating)** *We use an explicit rating, called the membership-rating $r^m : N \rightarrow \{-1, 0, +1\}$, for which (i) all symbols and axioms the user actually wants to be part of the ontology have rating $+1$, and (ii) all symbols and axioms not actually part of the ontology can be explicitly marked by the user with a rating $-1$. Finally, $0$ indicates an unrated element.*

Implicit feedback we can obtain from log information that indirectly indicate the importance of ontology elements based on how they have been used.

**Example 2 (Implicit Usage Rating)** *We use an implicit, usage-based rating called $r^u : N \rightarrow \mathbb{N}$, which indicates the relevance of the elements based on how they have been used,*

*e.g. counts the number of queries issued by the user and instances in her knowledge base that reference a given symbol name.*

*We consider two implicit usage rating annotations:*

- $r^u_{queries}$ *annotates the ontology elements with the number of queries that have referenced the particular ontology elements,*
- $r^u_{instances}$ *annotates the ontology elements with the number of instances that are classified under the particular ontology elements.*

The two rating annotations capture two important and typical dimensions of usage, one with respect to the content (how the concepts are used to classify instances), and one with respect to the usage by the end users, i.e. which concepts were actually queried. This information is available in a wide range of application scenarios. Of course, in specific scenarios further information may be available and thus additional rating annotations can be defined.

The focus of the usage-driven evaluation is to evaluate how effectively a particular ontology is used in an ontology-based application. In the following we present two particular forms of evaluation: (1) cost-based evaluation, where we capture the efficiency of using the ontology for a particular task, and (2) collaborative evaluation, where we consider the ontologies of other users and their usage context in a collaborative setting.

### 2.2.1 A Cost-based Approach

#### Cost-based Evaluation

With the evaluation function presented here we capture the intuition that the quality of an ontology built for browsing is determined by how efficiently it allows the users to obtain relevant instances. To measure the efficiency, we introduce a cost model to allow to quantify the user effort to arrive at the desired information. For the case of navigating a concept hierarchy, this cost is determined by the complexity of the hierarchy in terms of its breadth and depth: The *breadth* here means the number of choices (sibling nodes of the correct concept) the user has to consider to decide for the right branch to follow: The broader the hierarchy, the longer it takes to make the correct choice. The *depth* means, how many links does the user need to follow to arrive at the correct concept, under which the desired instance is classified: The deeper the hierarchy, the more "clicks" need to be performed. To minimize the cost, both depth and breadth need to be minimized, i.e. the right balance between them needs to be found.

A very simple, but intuitive cost function, is presented in the following. For a given ontology $O$ with the set of concepts $C$, we can calculate the cost as the weighted sum of the costs of the individual concepts, where the weight is the importance of the concept according to the rating annotations $r^u_{queries}$, i.e. the number of queries:

$$Cost(O) = \sum_{c \in C} r^{u}_{queries}(c) * cost(c)$$

The cost of an individual concept is:

$$cost(c) = cost(parent(c)) + (k_d + k_b * breadth(c))$$

Essentially, the cost is determined by the cost of the parent-concept ($parent(c)$), plus the cost for following the link (the constant $k_d > 1$) and the cost caused by the breadth of $c$ (weighted by a constant $k_b > 1$). Now we can define the ontology evaluation function $e_{cost}(O)$ as the reciprocal of $Cost(O)$. $e_{cost}(O)$ may now be applied as described in section 3 later.

#### Cost-based Heuristics for Evolution

[SSGS03] presents useful heuristics for the generation of changes based on local properties. However, as mentioned before these heuristics do not guarantee that a change leads to a global improvement of the ontology. In the following we describe a selection of two relevant complex change operations[2] and illustrate based on the example from the previous Section 2.2.2 on, how their application can improve the ontology according to the evaluation function.

**Grouping concepts** One possible change is to group concepts with low ratings under a newly introduced concept. This decreases the breadth at the level of the new concept and thus the cost of the sibling concepts.

**Pulling-up concepts** An alternative change is to "pull-up" important concepts that have previously been grouped under a common parent concept. The result is that the depth of these concepts is reduced, resulting in a decreased cost.

### 2.2.2 Collaborative Scenario

#### Collaborative Evaluation

A very typical form of evaluation is based on comparisons with a gold-standard, typically based on similarity measures. The problem here is that such a gold standard is hard to obtain. However, in a multi-user scenario with evolving personal ontologies we have the interesting situation that we can exploit other users' ontologies for evaluation. While each

[2]Of course, further complex change operations are possible. In this sense, the selection of these two change operations has exemplary character.

of them individually may not be gold standard, in their collectivity they may very well be drawn on for evaluation.

The basic idea is as follows: assume that for a target ontology we know similar ontologies called *neighbours* for short. We then would like to define the value of an ontology with respect to how similar it is to the ontology of other users. In particular, we want to take the ontologies of those users into account that have used the ontology in a similar way, i.e. with a similar context. In a next step we then would like to spot patterns in similar ontologies that are absent in our target ontology and recommend them to the target ontology.

To do so, we first need to define what it means for two ontologies to be similar and what it means for their contexts to be similar.

A similarity measure for ontologies can be defined as a function

$$\mathrm{sim}_{ontology} : \mathcal{O} \times \mathcal{O} \to [0, 1]$$

where $\mathrm{sim}_{ontology}(O, P)$ is large for similar ontologies $O$ and $P$ and small for dissimilar ontologies. Typically, these measures are symmetric and maximal for two same arguments. For further properties and examples of similarity functions for ontologies in general, we refer the reader to [EHSH05].

Similarly we can define a similarity measure for the context of two ontologies. Recall that ontologies have additional rating annotations that are valuable information to consider in such a similarity measure. We can for example choose a correlation measures (vector similarity) to compute similarities between the context of the ontologies $O$ and $P$ of two users based on their usage ratings $r^u_O$ and $r^u_P$ of the elements $N$ (i.e. symbol names and axioms) in the ontology:

$$\mathrm{sim}_{context}(O, P) := \frac{\sum_{n \in N} r^u_O(n)\, r^u_P(n)}{\sqrt{\sum_{n \in N} r^u_O(n)^2}\sqrt{\sum_{n \in N} r^u_P(n)^2}} \tag{2.1}$$

For details about such measures, we refer the reader to [HHSTS05].

Based on the similarity measure, we can define the value of ontology to be the average similarity with all its neighbours' neighbours, weighted by the similarity of the contexts of the ontologies:

$$e_{collaborative}(O) := \frac{\sum_{P \in \Omega} (\mathrm{sim}_{context}(O, P) * \mathrm{sim}_{ontology}(O, P))}{\sum_{P \in \Omega} \mathrm{sim}_{context}(O, P)}$$

We now present a recommender function for ontology changes based on collaborative filtering, which recommends changes to increase the value of the ontology according to this evaluation function.

### Collaborative Filtering for Evolution

Change discovery eventually can be improved by taking into account other users' ontologies and thereby establishing a collaborative ontology evolution scenario, where each user keeps her personal ontology but still profits from annotations of other users.

The basic idea is as follows: assume that for a target ontology we know similar ontologies called *neighbours* for short, then we would like to spot patterns in similar ontologies that are absent in our target ontology and recommend them to the target ontology. Another wording of the same idea is that we would like to extract ontology change operations that applied to the target ontology increases the similarity with its neighbours. We do that by applying collaborative filtering techniques to identify these changes. As in standard user-based collaborative filtering, ratings of all neighbours $\Omega$ are aggregated using the similarity-weighted sum of their membership ratings $r^m$:

$$r_{\textit{personalized}}(O, \Omega, c) := \frac{\sum_{P \in \Omega} \mathrm{sim}_{Context}(O, P)\, r_P^m(c)}{\sum_{P \in \Omega} \mathrm{sim}_{Context}(O, P)} \tag{2.2}$$

Let us discuss the intuition of this function: The membership ratings indicate whether the ontology element should be part of the user's ontology or not. With the similarity-weighted average we thus obtain a majority voting for the individual user. The recommendations are obtained directly from the rating: Elements with a positive rating are recommended to be added to the ontology, elements with a negative rating are recommended to be removed.

In [HHSTS05] we have presented experimental results of applying these methods based on collaborative filtering in a case study where users maintain personal ontologies. The results show that (1) the users indeed accept recommendations for ontology changes, and (2) recommendations based on the similarity-based evaluation functions lead to better recommendations than naive, non-personalized measures.

## 2.3 Domain Context

The second type of context we refer to as *domain context*, which considers the relation with the data that the ontology has been engineered from, to allow to assess how well the ontology reflects the underlying corpus of data. In this section we present two examples for data-driven approaches to modelling the domain context of ontology elements based on an analysis of textual data (e.g. domain-specific corpus) or linguistic resources such as WordNet [Mil95].

### Linguistic Properties

With 'Linguistic Properties' we refer to the relatedness between an ontology and a natural language description of the underlying domain. The linguistic properties of an ontology,

in particular the linguistic evidence for a set of ontology elements, can be determined by a detailed linguistic analysis of domain-specific corpora such as the information spaces described in Section 1.4. For this analysis we rely on ontology learning algorithms implemented, for example, in the ontology learning framework of Text2Onto [CV05].

In order to learn subclass-of relations, for instance, we apply a variety of algorithms exploiting the hypernym structure of WordNet [Mil95], matching Hearst patterns [Hea92] in the corpus and applying linguistic heuristics mentioned in [VNCN05]. All ontology learning algorithms provide different kinds of evidences with respect to the correctness and the relevance of ontology elements for the domain in question.

**Example 3 (Ontology learning)** *For example, from the following text fragment we might conclude with a certain confidence that the concept* 'computer scientist' *is a subclass of the concept* 'IT professional'*:*
"Technology and content costs also went up because the company is adding IT professionals such as computer scientists and software engineers to improve the customer experience and processes on its Web sites."
*From the pattern* "... X such as Y and Z..." *we derive with a certain confidence that* Y *and* Z *are instances or subclasses of* X.

Based on these evidences we can compute confidences which model the certainty about whether a particular ontology element holds for a certain domain. Since confidence can be considered as a corpus-based support for ontology elements, it can be used to span the space of possible ontologies, i.e. those ontologies which are (linguistically) supported by the underlying corpus. From this space of possible ontologies we will later choose those which bring us closer to the aim of an optimal ontology (see Section 2.1.4).

In line with the definition in Section 2.1.2 we can represent the confidence with respect to a particular ontology element by means of a special ontology rating annotation being a function

$$r_{conf} : N \rightarrow [0, 1]$$

Moreover, for the formalization of confidence we rely on probability theory, which we consider to be most appropriate for modeling the kind of uncertainty involved in our approach. Thus, we have

- A confidence is a value between 0 and 1. A confidence of 0 means that a proposition certainly does not hold. A confidence value of 1 means that a proposition certainly holds.
- The confidence for propositions (about ontology elements) and their complements add up to 1. When we talk about the confidence for the complement of a proposition we do not refer to the logical negation of the proposition, but to the fact the proposition does not hold.

- The combined confidence of two propositions (analogous to joint probability) is the product of the confidence for one of them and the confidence for the second, conditional on the first.

We believe that linguistic evidence with respect to an ontology can be appropriately measured by ontology learning techniques which try to capture the ontological commitment in human language. Although linguistic evidence is particularly relevant for terminological ontologies, similar data-driven approaches have frequently been applied in the evaluation of domain ontologies (see Section 1.3).

### Formal Properties

As a practical example for establishing the formal context of an ontology we consider the OntoClean methodology. OntoClean is a unique approach towards the formal evaluation of ontologies as it analyses the intensional content of concepts. It defines the four meta-properties rigidity ($R$), unity ($U$), identity ($I$) and dependence ($D$) which can be used for the formal verification of taxonomic (subclass-of) relationships. For a definition of the meta-properties refer to [GW04], or take a look at example 4 for a first idea of how OntoClean is applied.

In order to be able to integrate OntoClean's notion of formal ontology evaluation into our approach, we developed methods for the automatic tagging of concepts with OntoClean meta-properties [VVS05]. In particular, we match lexico-syntactic patterns on the domain corpus and the Web to obtain positive and negative evidence for rigidity, unity, dependence and identity of concepts in an OWL ontology.

For each pattern we collect positive and negative evidence for a concept having a certain meta-property by considering instances of the regarding pattern in the corpus. Given a concept $c$ and the normalized frequencies obtained for all patterns the decision whether or not a meta-property $p$ applies to $c$ is made by a classifier. A set of classifiers – one for each meta-property – can be trained on examples provided by human annotators (shown in [VVS05]). A certainty value for correctness of each classification result is provided by the classifier which has been trained for $p$.

Given the certainty of the classifier we can represent the confidence with respect to the tagging of a particular ontology element with a meta-property $p \in \{R, U, I, D\}$ by means of an ontology rating annotation which is a function

$$r_p : N \rightarrow [0, 1]$$

The approach for automatic tagging of ontological concepts with OntoClean meta-properties which we have chosen to obtain a formal context for the ontology has been successfully evaluated in [VVS05], which we consider as an important step towards automating formal evaluation of ontologies.

### 2.3.1 Domain-driven Evaluation

The focus of the domain-driven evaluation concerns the congruence or "fit" between an ontology and a domain of knowledge.

The goal of the evaluation is to obtain an ontology that is (1) consistent, and (2) captures the most certain information while disregarding the potentially erroneous information. Consistency here is regarded from two points of view: formal consistency, which disallows certain hierarchical relations between concepts based on their taggings as shown in the next example, and logical consistency, which allows meaningful reasoning as it leads to satisfiable ontologies (exemplified afterwards).

We can use the existing OntoClean rules on an ontology tagged with the formal meta-properties in order to check the ontology for consistency. Here we will give only one illustrative example for such a rule. For a full list refer to [GW04]. As shown in [SAS03] such rules can be formalized as logical axioms and validated by an inference engine. Here we give one applied example of OntoClean.

**Example 4 (Consistency of Formal Properties)** *Rigidity of a concept means that every instance of this concept must always be an instance of this concept, in every possible world and at any time as long as it exists, as it is essential to each instance. OntoClean states the following restriction: a rigid concept must not be subsumed by a concept that is not rigid to all of its instances (a so called anti-rigid concept). The classic example is student, an anti-rigid concept (as, contrary to some beliefs, no students are doomed to always remain students), subsuming human, a rigid concept, which is obviously wrong: whereas every student is free to leave the university and stop being a student, humans cannot stop being humans. But as every human would be a student, according to the example, they never could stop being a student, which contradicts the previous sentence.*

Another important form of consistency is that of logical consistency, which refers to the model-theoretic semantics of the ontology. An ontology is logically inconsistent, if it contains contradicting axioms that will allow no possible interpretations. A consequence of a logically inconsistent ontology is that it allows no meaningful reasoning, as any axiom is entailed by an inconsistent ontology. From this perspective, a logically inconsistent ontology has no value.

**Example 5 (Logical Consistency)** *Consider the following ontology:*
$O_1 = \{Employee \sqsubseteq Person, Student \sqsubseteq Person, PhDStudent \sqsubseteq Student, PhDStudent \sqsubseteq Employee, Employee \sqsubseteq \neg Student$ *(Stating that employees cannot be students.) In this ontology, the concept* $PhDStudent$ *does not allow any interpretation, it is thus inconsistent.*

In general, for a particular domain, there may be many different consistent ontologies. For the above example, a consistent ontology could be obtained by removing either the axiom $\alpha = PhDStudent \sqsubseteq Employee$, $PhDStudent \sqsubseteq Student$, or $Employee \sqsubseteq$

$\neg Student$. The difficulty is to select the "best" ontology, i.e. the one that will result in most meaningful reasoning.

For our particular goal to obtain a consistent ontology capturing the most certain information, we can define an evaluation function as follows:

$$e_{certainty}(O) = \begin{cases} \max\left(\frac{\sum_{\alpha \in O} r_{conf}(\alpha) - t}{\|O\|}, 0\right) & \text{if } O \text{ is consistent} \\ 0 & \text{if } O \text{ is inconsistent} \end{cases} \tag{2.3}$$

Let us discuss the intuition behind this function. The basic idea is to maximize the certainty of the ontology based on the confidence of its individual axioms, as given by $r_{conf}(\alpha)$. $t$ is a threshold that is introduced to "filter out" axioms with a confidence level below a minimal value: An axiom with a confidence below $t$ will thus decrease the value of an ontology. An inconsistent ontology is defined to have "no value".

### 2.3.2 Domain-Driven Evolution

The goal of the domain-driven evolution is to adapt the ontology to the changes in the underlying domain corpus. Over time, new information may become relevant that is not yet reflected in the ontology, or existing information may become obsolete. In Section 2.3.1 we have already presented an evaluation function that captures this intuition. We now outline an algorithm that exploits the behaviour of the evaluation function and local characteristics of inconsistencies to maximize the value. It is based on the ideas of consistent ontology evolution as presented in [HS05a]. Consistent ontology evolution ensures the consistency of changing ontologies by mapping consistency conditions to resolution functions that resolve introduced inconsistencies. The task of the resolution function consists of two main steps: (1) localizing the inconsistency and (2) generating additional changes that lead to another consistent state.

Algorithm 1 shows how to optimize an ontology $O$ in the following way: Starting with the current state of the ontology $O$, we incrementally add axioms from the space of possible changes whose confidence is equal to or greater than the threshold $t$. The space of possible ontology changes essentially consists of all axioms that have been rated (e.g. by the ontology learning tool), but that are not actually part of the ontology. If adding the axioms leads to an inconsistent ontology, we localize the inconsistency by identifying a minimal inconsistent subontology. (For the details of this procedure, we refer the reader to [HS05a]). An ontology $O'$ is a minimal inconsistent subontology of $O$, if $O'$ is inconsistent and every subontology of $O'$ is consistent. Within this minimal inconsistent subontology we then identify the axiom that is most uncertain, i.e. has the lowest confidence value. This axiom will be removed from the ontology, thus resolving the inconsistency.

In [HV05] we have applied this approach to ontologies learned from a corpus of documents from the BT Digital Library using the evaluation function and algorithms presented

**Algorithm 1** Algorithm for Ontology Evolution

**Require:** The current ontology $O$, ontology changes $OCO$

**for all** $\alpha^+ \in OCO, \alpha^+ \notin O, r_{conf}(\alpha^+) \geq t$ **do**
  $O' := O \cup \{\alpha^+\}$
  **while** $O'$ is inconsistent **do**
    $O'' :=$ minimal_inconsistent_subontology$(O', \alpha^+)$
    $\alpha^- := \alpha^+$
    **for all** $\alpha' \in O''$ **do**
      **if** $r_{conf}(\alpha') \leq r_{conf}(\alpha^+)$ **then**
        $\alpha^- := \alpha'$
      **end if**
    **end for**
    $O' := O' \setminus \{\alpha^-\}$
  **end while**
  **if** $e(O) < e(O')$ **then**
    $O := O'$
  **end if**
**end for**

in the previous section. Here we performed an analysis of the influence of the threshold of uncertainty on the obtained ontology. The results clearly show the connection between the level of uncertainty and inconsistency introduced. A low threshold $t$ results in more uncertain information being allowed in the target ontology. As a result, the chances for inconsistencies increase. How to choose the "right" threshold $t$ will very much depend on the application scenario, as it essentially means finding a trade-off between the amount of information learned and the confidence in the correctness of the learned information.

# Chapter 3

# Results

In this chapter we present evaluation results for an application of the framework and methods presented in the previous chapter. We first describe the evaluation setting in Section 3.1 and report the results in Section 3.2.

## 3.1 Evaluation Setting

For our evaluation we have chosen the use case of question answering over a knowledge base extracted from a collection of textual resources of the BT DL, in particular the Knowledge Management information space. The ontology changes have been discovered / generated with Text2Onto, using the algorithms presented in [VS05]. The goal of this evaluation was to evaluate (1) the applicability of the approach of incremental ontology evolution for this learning task, (2) the effect of parameters such as the required confidence of information on the evolution process, and (3) the performance of the developed algorithms.

## 3.2 Evaluation Results

We have applied the approach presented in the previous chapter to ontologies learned from a corpus of 1700 abstracts (from documents about knowledge management) of the BT Digital Library. The learned ontology consisted of 938 concepts and 125 instances. For the concepts, 406 subconcept-of relations and 2322 disjoint-concepts relations were identified. For the instances, 143 instance-of relations were obtained (as multiple instantiations are allowed).

For the incremental transformation of the learned ontology to an OWL ontology, we applied the evaluation function and algorithms presented in the previous section. Here we performed an analysis of the influence of the threshold of uncertainty on the transforma-

| Threshold $t$ | # of Inconsistencies | # of Axioms in Result | Time in seconds |
|---|---|---|---|
| 0.1 | 40 | 1686 | 1305 |
| 0.2 | 8 | 896 | 111 |
| 0.4 | 3 | 389 | 11 |
| 0.8 | 0 | 197 | 7 |

Table 3.1: Influence of certainty threshold $t$ on transformation process

tion. The results in Table 3.1 clearly show the connection between the level of uncertainty and inconsistency introduced:

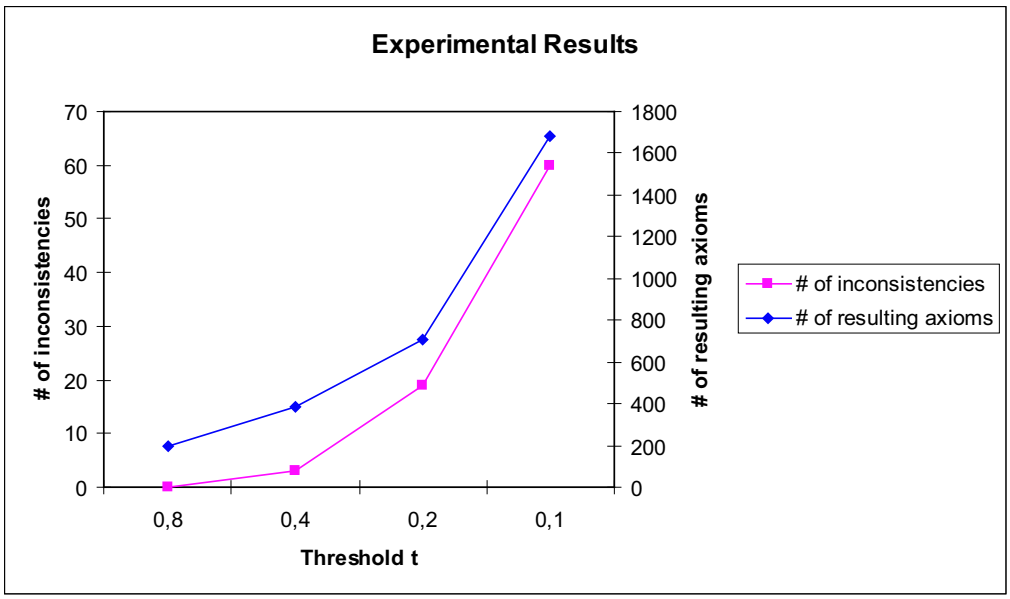



Figure 3.1: Evaluation Results: Threshold vs. Size of KB and Inconsistencies

A low threshold $t$ results in more uncertain information being allowed in the target ontology. As a result, the chances for inconsistencies increase. How to choose the "right" threshold $t$ for the transformation process will very much depend on the application scenario, as it essentially means finding a trade-off between the amount of information learned and the confidence in the correctness of the learned information.

In the following we will discuss typical types of inconsistencies and present examples of such inconsistencies that were detected and resolved. The first type of inconsistency involves unsatisfiable concepts (often called incoherent concepts) in the T-Box of the ontology. This can for example happen if two concepts are identified to be disjoint, but at the same time these concepts are in a subconcept-relation (either explicitly asserted or inferred). Interestingly, this type of inconsistency often occurred for concepts for which

even for a domain expert the correct relationship is hard to identify, as the following example shows:

**Example 6** *The relationship between the concepts* Data, Information, *and* Knowledge *is a very subtle (often philosophical) one, for which one will encounter different definitions depending on the context. The (inconsistent) definitions learned from our data set stated that $Data$ is a subconcept of both $Information$ and $Knowledge$, while $Information$ and $Knowledge$ are disjoint concepts:*

| *Axiom* $t$ | *Confidence* |
|---|---|
| $Data \sqsubseteq Information$ | *1.0* |
| $Data \sqsubseteq Knowledge$ | *1.0* |
| $Information \sqsubseteq \neg Knowledge$ | *0.7* |

*The inconsistency was resolved by removing the disjointness axiom, as its confidence value was lowest.*

The second type of inconsistency involves A-Box assertions. Here, typically instances were asserted to be instances of two concepts that were identified to be disjoint. We again present an example:

**Example 7** *Here $KaViDo$ was identified to be both an instance of $Application$ and a $Tool$ (based on the abstract of [TD03]), however, $Application$ and $Tool$ were learned to be disjoint concepts:*

| *Axiom* $t$ | *Confidence* |
|---|---|
| $Application(kavido)$ | *0.46* |
| $Tool(kavido)$ | *0.46* |
| $Tool \sqsubseteq \neg Application$ | *0.3* |

*This inconsistency was again resolved by removing the disjointness axiom.*

Other types of inconsistencies involving, for example, domain and range restrictions were not considered in our current experiments, thus being left for future work. Nevertheless, this evaluation showed that inconsistency is an important issue in ontology learning.

### 3.2.1 Performance Results

Maintaining consistency during ontology evolution obviously implies a performance overhead. In the case of maintaining logical consistency, the satisfiability needs to be checked after every add-operation (as mentioned before, because of the monotonicity, remove-operations can not cause a logical inconsistency). If logical inconsistencies are found, resolving them (by identifying the cause of the inconsistency) means additional

overhead. Checking satisfiability on the other hand, is computationally expensive in the first place (NEXPTIME for OWL DL, although the ontologies used in these tests are in a lower complexity class).

Figure 3.2 demonstrates this situation for the previous evaluation scenario. It shows the time required for the incremental ontology evolution depending on the threshold of confidence. The relationship between the two is not a direct one: First – as shown in the previous figure – a lower threshold means a larger resulting knowledge base, and second, a larger number of inconsistencies is introduced. The time required for the evolution process depends on both the number of inconsistencies to be resolved *and* the size of the knowledge base. As a result,

The consequences for practical applications are the following:

- Ontology evolution tasks (e.g. based on ontology learning) will be typically done offline, such that results are not required in real-time. Furthermore, ontology learning is already expensive by itself, such that we expect that the costs for consistent ontology evolution will not be a dominant factor.
- For cases where the performance overhead is an issue, we recommend to tune the evolution process to decrease potential inconsistencies, e.g. by requiring a larger threshold for the confidence.

On the other hand, for future work on improving the performance for large knowledge bases, we see the following options:

- The first option concerns the algorithms for reasoning with expressive ontologies (which includes e.g. satisfiability checking). Efficient reasoning algorithms for large ontologies have recently attracted increased attention especially in the context of large A-Boxes. In Appendix A we report on some of the latest results in this area.
- The second option concerns the algorithms for resolving inconsistencies. Again, the field of diagnosis and repair is an active research field. In fact, we see the potential to re-use methods developed in the scope of this project in task 3.4 (Diagnosis and Repair). This synergy has already been promoted with the work in [HvHH+05].

### 3.2.2 Performance of Reasoning with KAON2

As we have seen in the previous results, the performance of consistent ontology evolution is heavily dependent on the complexity and performance of the underlying reasoning algorithms. The development of efficient algorithms for reasoning with ontologies in very expressive description logics (in particular OWL DL) as a main focus of the KAON2 engine. KAON2 is an ontology management infrastructure developed within the European

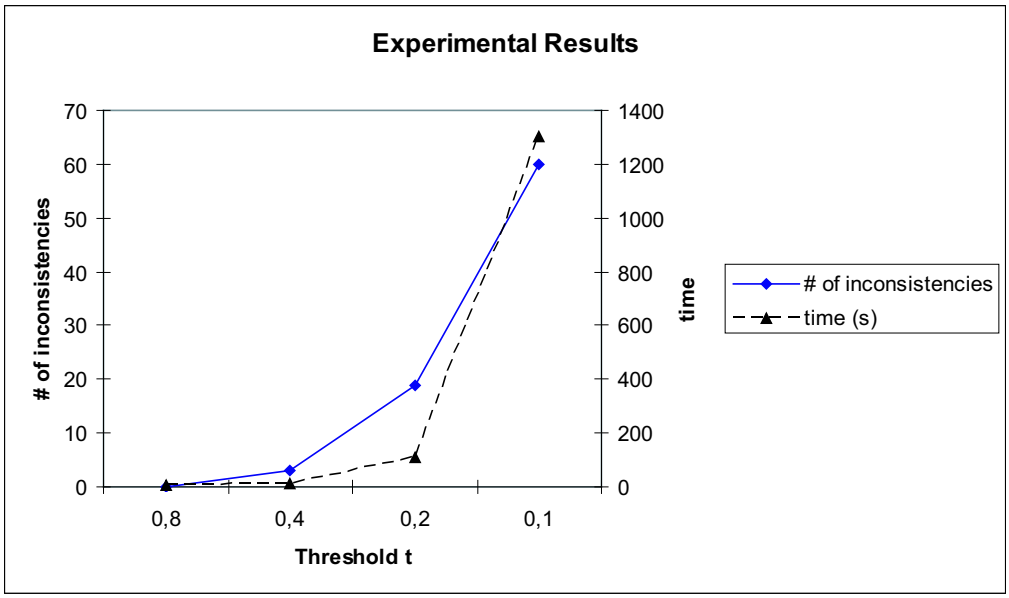



Figure 3.2: Evaluation Results: Threshold vs. Time and Inconsistencies

Semantic Systems initiative (ESSI[1]) cluster. The KAON2 core reasoner has been to a large extent developed in the scope of the DIP[2] project, while extensions e.g. for evolution support have been developed in the scope of SEKT. As such, the performance evaluations of the KAON2 reasoner are not only relevant for the methods developed within this task, but also for all other components and methods relying on the ontology and metadata management developed within this workpackage. We therefore provide recent results of performance evaluations in Appendix A.

[1] `http://www.essi-cluster.org/`
[2] urlhttp://dip.semanticweb.org/

# Chapter 4

# Conclusions and Future Work

## 4.1 Conclusion

In this deliverable we have presented an approach for handling the tasks of ontology evaluation and ontology evolution jointly integrated as part of the ontology lifecycle, instead of treating them as two separate phases.

The core questions we have answered were: How can we define what a good ontology for a particular context is, and how can we perform ontology evolution to actually obtain better ontologies in an automated manner. A central role in this approach is played by the ontology evaluation function, which guides the discovery of changes that lead to an improved ontology.

Considering a practical application scenario as an example, we have shown how to define the context, evaluation functions and methods for evolution. We here selected two particularly important forms of context: the usage-context, and the domain context, which are relevant in many ontology-based applications. In this sense, the methods presented in this deliverable can readily be employed for other application scenarios. On the other hand, the approach provides the flexibility to essentially define arbitrary ontology evaluation functions for a variety of contexts and is open to embed new methods for change discovery leading to improved ontologies.

The key advantage of our presented work is that we allow users a great flexibility in defining for themselves what a good ontology is, and offer them in return a system that provides automatic support for making ontologies even better. In a nutshell the framework needs to be extended only once with an evaluation function, then all kind of different extensions can be applied without having to change the ontology evaluation function.

We have further applied and evaluated the framework in the context of incremental ontology learning. As we have shown, uncertainty and inconsistencies are issues that need to be dealt with in order to allow meaningful reasoning over the learned ontologies. We have presented how uncertainty can be represented in the Learned Ontology Model (LOM) and

how such learned ontologies can be transformed to consistent OWL ontologies using the notion of an ontology evaluation function. Our experiments with ontologies learned from documents of a Digital Library show the feasibility and usefulness of the approach.

## 4.2 Future Work

Future work has to cover such extensions, as well as creating a user-friendly and usable software framework that is able to apply the whole system automatically and create self-contained and self-evolving systems. As extensions we envision evolutionary algorithms to be used to optimize ontologies. Further we could imagine so-called 'pro-active' ontology change operations, meaning methods which analyse the given evaluation criteria, anticipate changes, and act accordingly to update ontologies. Thus, another level of automatization of ontology engineering could be realized on top of our presented work.

In our work we have focused on one particular approach to evaluation based on the confidence as generated by ontology learning algorithms, i.e. a data-driven approach to the evaluation of ontologies. There are many other notions of ontology quality and consistency which could be used for the definition of an ontology evolution function.

# Appendix A

# Performance of Reasoning with KAON2

To test our algorithms on practical problems, we compared the performance of KAON2 with that of other DL reasoners. This comparison, however, is slightly blurred since it compares *implementations*, and not algorithms. DL algorithms are complex, and overheads in maintaining data structures or memory management can easily dominate the run time. The implementation language itself may introduce limitations that become evident when dealing with large data sets. Finally, we had to cope with the usual "chicken-and-egg" problem: powerful ABox reasoners are a rather recent developement, so few knowledge bases with both interesting TBoxes and large ABoxes exist.

Therefore, the results we present in this section should not be taken as a definitive measure of practicability of either algorithm. However, they do show that deductive database techniques can significantly improve the performance of DL reasoning. This seems to hold even if the TBox is complex, as long as it is relatively small. For TBox reasoning, our results show a mixed picture: on certain ontologies, KAON2 performs relatively well, whereas on others it is slower than the tableau systems. This suggests that further optimizations are needed to match the robustness of tableau algorithms on TBox reasoning problems.

## A.1 Test Setting

We compared KAON2 with RACER and Pellet. To the best of our knowledge, these are the only reasoners that provide sound and complete algorithms for $\mathcal{SHIQ}$ with ABoxes.

RACER[1] [HM01] was developed at the Concordia University and the Hamburg University of Technology, and is written in Common Lisp. We used the version 1.8.2, to which we connected using the JRacer library. RACER provides an optimized reasoning mode (so-called nRQL mode 1), which provides significant performance improvements,

[1] `http://www.racer-systems.com/`

but which is complete only for certain types of knowledge bases. At the time of writing, RACER did not automatically recognize whether the optimized mode is applicable to a particular knowledge base, so we used RACER in the mode which guarantees completeness (so-called nRQL mode 3). Namely, determining whether optimizations are applicable is a form of reasoning which, we believe, should be taken into account in a fair comparison.

Pellet[2] [PS04] was developed at the University of Maryland, and it is the first system which fully supports OWL-DL, taking into account all the nuances of the specification. It is implemented in Java, and is freely available with the source code. We used the version 1.3 beta.

For each reasoning task, we started a fresh instance of the reasoner and loaded the test knowledge base. Then, we measured the time required to execute the task. We made sure that all systems return the same answers.

Many optimizations of tableau algorithms involve caching computation results, so the performance of query answering should increase with each subsequent query. Furthermore, both RACER and Pellet check ABox consistency before answering the first query, which typically takes much longer than computing query results. Hence, starting a new instance of the reasoner for each query might seem unfair. We justify our approach as follows.

First, the effectiveness of caching depends on the type of application: if an ABox changes frequently, caching is not very useful. Second, usefulness of caches also depends on the degree of similarity between queries. Third, we did not yet consider caching for KAON2; however, materialized views were extensively studied in deductive databases, and [Vol04] applied them to ontology reasoning. Finally, KAON2 does not perform a separate ABox consistency test because ABox inconsistency is discovered automatically during query evaluation. Hence, we decided to measure only the performance of the actual reasoning algorithm, and to leave a study of possible materialization and caching strategies for future work. Since ABox consistency check is a significant source of overhead for tableau systems, we measured the time required to execute it separately. Hence, in our tables, we distinguish one-time *setup* time (S) and query processing time (Q) for Pellet and Racer. The time for computing the datalog program in KAON2 was not significant, so we include it into the query processing time.

All tests were performed on a laptop computer with a 2 GHz Intel processor, 1 GB of RAM, running Windows XP Service Pack 2. For Java-based tools, we used Sun's Java 1.5.0 Update 5. The virtual memory of the Java virtual machine was limited to 800 MB, and each reasoning task was allowed to run for at most 5 minutes.

[2] `http://www.mindswap.org/2003/pellet/index.shtml`

Table A.1: Statistics of Test Ontologies

| $KB$ | incl. | eq. | disj. | func. | dom. | rng. | RBox | $C(a)$ | $R(a,b)$ |
|---|---|---|---|---|---|---|---|---|---|
| vicodi_0 | | | | | | | | 16942 | 36711 |
| vicodi_1 | | | | | | | | 33884 | 73422 |
| vicodi_2 | 193 | 0 | 0 | 0 | 10 | 10 | 10 | 50826 | 110133 |
| vicodi_3 | | | | | | | | 67768 | 146844 |
| vicodi_4 | | | | | | | | 84710 | 183555 |
| semintec_0 | | | | | | | | 17941 | 47248 |
| semintec_1 | | | | | | | | 35882 | 94496 |
| semintec_2 | 55 | 0 | 113 | 16 | 16 | 16 | 6 | 53823 | 141744 |
| semintec_3 | | | | | | | | 71764 | 188992 |
| semintec_4 | | | | | | | | 89705 | 236240 |
| lubm_1 | | | | | | | | 18128 | 49336 |
| lubm_2 | 36 | 6 | 0 | 0 | 25 | 18 | 9 | 40508 | 113463 |
| lubm_3 | | | | | | | | 58897 | 166682 |
| lubm_4 | | | | | | | | 83200 | 236514 |
| wine_0 | | | | | | | | 247 | 246 |
| wine_1 | | | | | | | | 741 | 738 |
| wine_2 | | | | | | | | 1235 | 1230 |
| wine_3 | | | | | | | | 1729 | 1722 |
| wine_4 | | | | | | | | 2223 | 2214 |
| wine_5 | 126 | 61 | 1 | 6 | 6 | 9 | 9 | 2717 | 2706 |
| wine_6 | | | | | | | | 5187 | 5166 |
| wine_7 | | | | | | | | 10127 | 10086 |
| wine_8 | | | | | | | | 20007 | 19926 |
| wine_9 | | | | | | | | 39767 | 39606 |
| wine_10 | | | | | | | | 79287 | 78966 |
| dolce | 203 | 27 | 42 | 2 | 253 | 253 | 522 | 0 | 0 |
| galen | 3237 | 699 | 0 | 133 | 0 | 0 | 287 | 0 | 0 |

## A.2 Test Ontologies

We based our tests on ontologies available in the Semantic Web community. To obtain sufficiently large test ontologies, we used ABox replication—duplication of ABox axioms with appropriate renaming of individuals. The information about the structure of ontologies we used is summarized in Table A.1. All test ontologies are available on the KAON2 Web site.[3]

An ontology about European history was manually created in the EU-funded VICODI project.[4] The TBox is relatively small and simple: it consists of role and concept inclusion axioms, and domain and range specifications; it does not contain disjunctions, existential quantification, or number restrictions. However, the ABox is relatively large, with many interconnected instances. With vicodi_0, we denote the ontology from the project, and

[3] `http://kaon2.semanticweb.org/download/test\_ontologies.zip`
[4] `http://www.vicodi.org/`

with vicodi_$n$ the one obtained by replicating $n$ times the ABox of vicodi_0.

An ontology about financial services was created in the SEMINTEC project[5] at the University of Poznan. Like VICODI, this ontology is relatively simple: it does not use existential quantifiers or disjunctions; it does, however, contain functionality assertions and disjointness constraints. With semintec_0, we denote the ontology from the project, and with semintec_$n$ the one obtained by replicating $n$ times the ABox of semintec_0.

Lehigh University Benchmark (LUBM)[6] was developed by [GPH04] as a benchmark for testing performance of ontology management and reasoning systems. The ontology describes organizational structure of universities, and is relatively simple: it does not use disjunctions or number restrictions, but it does use existential quantifiers, so it is in Horn-$\mathcal{ALCHI}$ fragment. Each lubm_$n$ is generated automatically by specifying the number $n$ of universities.

The Wine[7] ontology contains a classification of wines. It uses nominals, which our algorithms cannot handle. Therefore, we apply a sound but incomplete approximation: we replace each enumerated concept $\{i_1, \ldots, i_n\}$ with a new concept $O$, and add assertions $O(i_k)$. Thus obtained ontology is relatively complex: it contains functionality axioms, disjunctions, and existential quantifiers. With wine_0, we denote the original ontology, and with wine_$n$ the one obtained by replicating $2^n$ times the ABox of wine_0.

This approximation of nominals is incomplete for query answering: for completeness one should further add a clause $\neg O(x) \vee x \approx i_1 \vee \ldots \vee x \approx i_n$. Furthermore, Pellet fully supports nominals, so one may question whether the Wine ontology is suitable for our tests. Unfortunately, in our search for test data, we could easily find ontologies with a complex TBox but without an ABox, or ontologies with an ABox and only a simple TBox, or a TBox with nominals. The (approximated) Wine ontology was the best ontology we found that contained both a nontrivial TBox and an ABox. We also used this approximated ontology in tests with Pellet, in order to ensure that all systems are dealing with the same problem.

DOLCE[8] is a foundational ontology developed at the Laboratory for Applied Ontology of Italian National Research Council. It is very complex, and no reasoner currently available can handle it. Therefore, the ontology has been factored into several modules. We used the DOLCE OWL version 397, up to the Common module (this includes the DOLCE-Lite, ExtDnS, Modal and Common modules).

GALEN[9] is a medical terminology ontology developed in the GALEN project [RNG93]. It has a very large and complex TBox, and has traditionally been used as a benchmark for terminological reasoning.

[5] `http://www.cs.put.poznan.pl/alawrynowicz/semintec.htm`
[6] `http://swat.cse.lehigh.edu/projects/lubm/index.htm`
[7] `http://www.schemaweb.info/schema/SchemaDetails.aspx?id=62`
[8] `http://www.loa-cnr.it/DOLCE.html`
[9] We obtained GALEN through private communication with Ian Horrocks.

The results of the tests are shown in Figure A.1. Tests which ran either out of memory or out of time are denoted with a value of 10000.

## A.3 Querying Large ABoxes

**VICODI.** Because VICODI does not contain existential quantifiers or disjunctions, it can be converted into disjunctive datalog directly, without invoking the reduction algorithm. Hence, reasoning with VICODI requires only an efficient deductive database. From the ontology author we received the following two queries, used in the project: ? -

$$\begin{aligned} Q_{V_1}(x) &\equiv \mathit{Individual}(x) \\ Q_{V_2}(x,y,z) &\equiv \mathit{Military\text{-}Person}(x), \mathit{hasRole}(y,x), \mathit{related}(x,z) \end{aligned}$$

The results show that Pellet and RACER spend the bulk of their time in checking ABox consistency by computing a completion of the ABox. Because the ontology is simple, no branch splits are performed, so the process yields a single completion representing a model. Query answering is then very fast, as it amounts to model lookup.

According to the authors of Racer, the gap in performance between Pellet and Racer should be resolved in the next release of Racer.

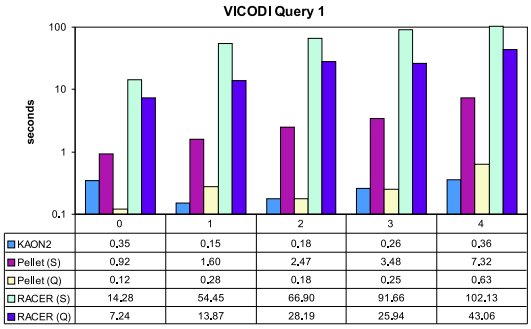
**SEMINTEC.** The SEMINTEC ontology is also very simple; however, it is interesting because it contains functional roles and therefore requires equality reasoning. From the ontology author, we obtained the following two queries, used in the project: ? -

$$\begin{aligned} Q_{S_1}(x) &\equiv \mathit{Person}(x) \\ Q_{S_2}(x,y,z) &\equiv \mathit{Man}(x), \mathit{isCreditCardOf}(y,x), \mathit{Gold}(y), \\ &\qquad \mathit{livesIn}(x,z), \mathit{Region}(z) \end{aligned}$$

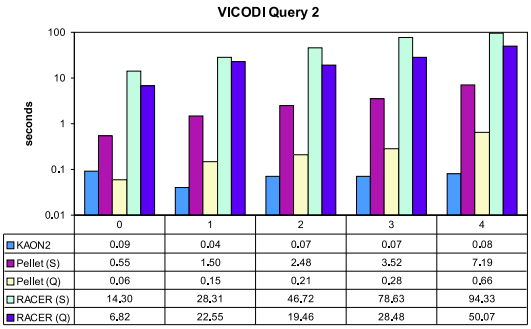
The SEMINTEC ontology is roughly of the same size as the VICODI ontology; however, the time that KAON2 takes to answer a query to SEMINTEC are one order of magnitude larger than for the VICODI ontology. This is mainly due to equality, which is difficult for deductive databases.

**LUBM.** LUBM is comparable in size to the VICODI and the SEMINTEC ontologies, but its TBox contains complex concepts. It uses existential quantifiers, so our reduction algorithm must be used to eliminate function symbols. Also, the ontology does not contain disjunctions and equality, so the translation produces an equality-free Horn program.

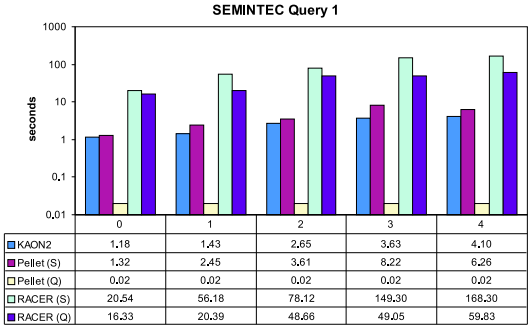
We wanted a mix of simple and complex queries, so we selected three queries from



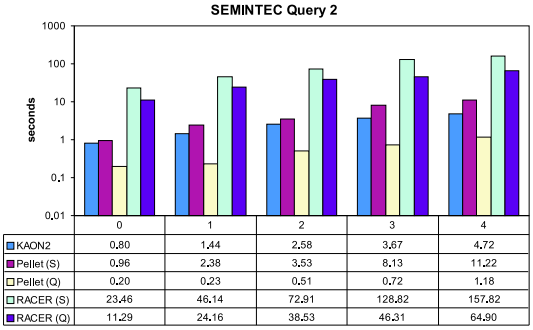

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| KAON2 | 0.35 | 0.15 | 0.18 | 0.26 | 0.36 |
| Pellet (S) | 0.92 | 1.60 | 2.47 | 3.48 | 7.32 |
| Pellet (Q) | 0.12 | 0.28 | 0.18 | 0.25 | 0.63 |
| RACER (S) | 14.28 | 54.45 | 66.90 | 91.66 | 102.13 |
| RACER (Q) | 7.24 | 13.87 | 28.19 | 25.94 | 43.06 |



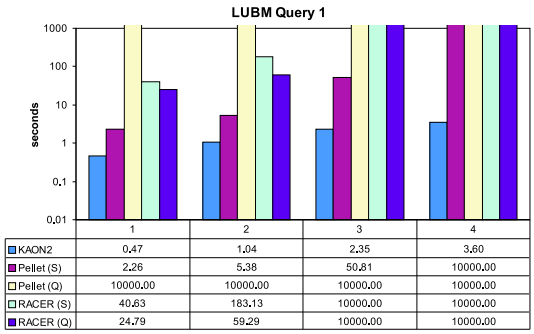

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| KAON2 | 0.09 | 0.04 | 0.07 | 0.07 | 0.08 |
| Pellet (S) | 0.55 | 1.50 | 2.48 | 3.52 | 7.19 |
| Pellet (Q) | 0.06 | 0.15 | 0.21 | 0.28 | 0.66 |
| RACER (S) | 14.30 | 28.31 | 46.72 | 78.63 | 94.33 |
| RACER (Q) | 6.82 | 22.55 | 19.46 | 28.48 | 50.07 |



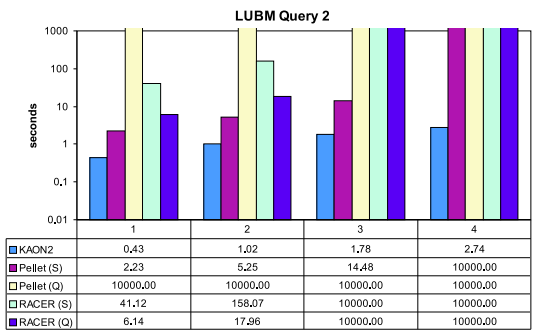

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| KAON2 | 1.18 | 1.43 | 2.65 | 3.63 | 4.10 |
| Pellet (S) | 1.32 | 2.45 | 3.61 | 8.22 | 6.26 |
| Pellet (Q) | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 |
| RACER (S) | 20.54 | 56.18 | 78.12 | 149.30 | 168.30 |
| RACER (Q) | 16.33 | 20.39 | 48.66 | 49.05 | 59.83 |



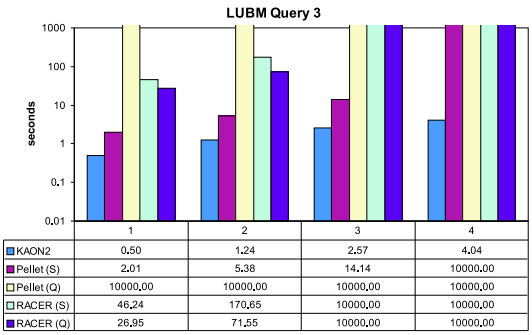

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| KAON2 | 0.80 | 1.44 | 2.58 | 3.67 | 4.72 |
| Pellet (S) | 0.96 | 2.38 | 3.53 | 8.13 | 11.22 |
| Pellet (Q) | 0.20 | 0.23 | 0.51 | 0.72 | 1.18 |
| RACER (S) | 23.46 | 46.14 | 72.91 | 128.82 | 157.82 |
| RACER (Q) | 11.29 | 24.16 | 38.53 | 46.31 | 64.90 |



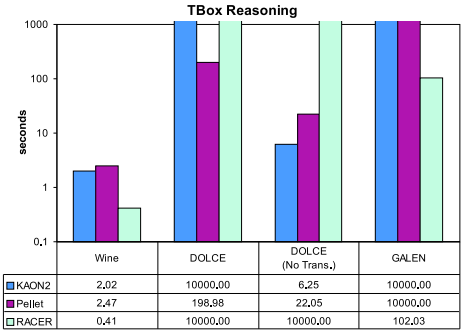

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| KAON2 | 0.47 | 1.04 | 2.35 | 3.60 |
| Pellet (S) | 2.26 | 5.38 | 50.81 | 10000.00 |
| Pellet (Q) | 10000.00 | 10000.00 | 10000.00 | 10000.00 |
| RACER (S) | 40.63 | 183.13 | 10000.00 | 10000.00 |
| RACER (Q) | 24.79 | 59.29 | 10000.00 | 10000.00 |




| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| KAON2 | 0.43 | 1.02 | 1.78 | 2.74 |
| Pellet (S) | 2.23 | 5.25 | 14.48 | 10000.00 |
| Pellet (Q) | 10000.00 | 10000.00 | 10000.00 | 10000.00 |
| RACER (S) | 41.12 | 158.07 | 10000.00 | 10000.00 |
| RACER (Q) | 6.14 | 17.96 | 10000.00 | 10000.00 |



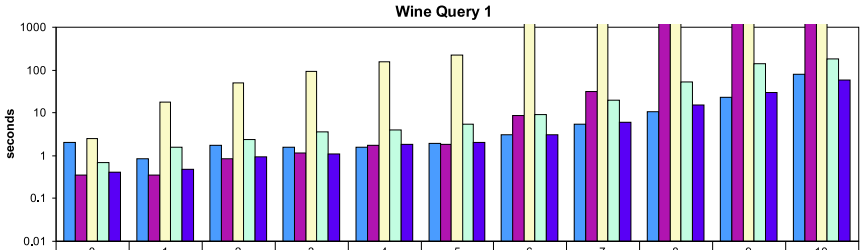

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| KAON2 | 0.50 | 1.24 | 2.57 | 4.04 |
| Pellet (S) | 2.01 | 5.38 | 14.14 | 10000.00 |
| Pellet (Q) | 10000.00 | 10000.00 | 10000.00 | 10000.00 |
| RACER (S) | 46.24 | 170.65 | 10000.00 | 10000.00 |
| RACER (Q) | 26.95 | 71.55 | 10000.00 | 10000.00 |




| | Wine | DOLCE | DOLCE (No Trans.) | GALEN |
|---|---|---|---|---|
| KAON2 | 2.02 | 10000.00 | 6.25 | 10000.00 |
| Pellet | 2.47 | 198.98 | 22.05 | 10000.00 |
| RACER | 0.41 | 10000.00 | 10000.00 | 102.03 |

**Note:** (S) — one-time setup time (including ABox consistency check)
(Q) — time required to process the query

Figure A.1: Test Results




| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| KAON2 | 2.02 | 0.85 | 1.72 | 1.60 | 1.61 | 1.95 | 3.16 | 5.34 | 10.78 | 23.16 | 80.44 |
| Pellet (S) | 0.36 | 0.36 | 0.86 | 1.15 | 1.76 | 1.88 | 8.61 | 31.79 | 10000.00 | 10000.00 | 10000.00 |
| Pellet (Q) | 2.47 | 17.50 | 50.44 | 92.48 | 154.82 | 224.77 | 10000.00 | 10000.00 | 10000.00 | 10000.00 | 10000.00 |
| RACER (S) | 0.68 | 1.58 | 2.33 | 3.53 | 3.93 | 5.31 | 8.91 | 19.89 | 51.53 | 138.30 | 182.77 |
| RACER (Q) | 0.41 | 0.47 | 0.94 | 1.12 | 1.82 | 2.07 | 3.10 | 6.02 | 15.35 | 29.35 | 59.63 |

Figure A.1: Test Results (continued)

the LUBM Web site: ? -

$$
\begin{aligned}
Q_{L_1}(x) &\equiv \mathit{Chair}(x) \\
Q_{L_2}(x,y) &\equiv \mathit{Chair}(x), \mathit{worksFor}(x,y), \mathit{Department}(y), \\
&\quad \mathit{subOrganizationOf}(y, \text{http://www.University0.edu}) \\
Q_{L_3}(x,y,z) &\equiv \mathit{Student}(x), \mathit{Faculty}(y), \mathit{Course}(z), \mathit{advisor}(x,y), \\
&\quad \mathit{takesCourse}(x,z), \mathit{teacherOf}(y,z)
\end{aligned}
$$

As our results show, LUBM does not pose significant problems for KAON2; namely, the translation produces an equality-free Horn program, which KAON2 evaluates in polynomial time. Although LUBM is roughly of the same size as VICODI, both Pellet and Racer performed better on the latter; namely, Pellet was not able to answer any of the LUBM queries within the given resource constraints, and Racer performed significantly better on VICODI than on LUBM. We were surprised by this result: the ontology is still Horn, so an ABox completion can be computed in advance and used as a cache for query answering. By analyzing a run of Pellet on lubm_1 in a debugger, we observed that the system performs disjunctive reasoning (i.e., it performs branch splits). Further investigation showed that this is due to *absorption* [Hor97]—a well-known optimization technique used by all tableau reasoners. Namely, an axiom of the form $C \sqsubseteq D$, where $C$ is a complex concept, increases the amount of don't-know nondeterminism in a tableau because it yields a disjunction $\neg C \sqcup D$ in the label of each node. If possible, such an axiom is transformed into an equivalent *definition* axiom $A \sqsubseteq C'$ (where $A$ is an atomic concept), which can be handled in a deterministic way. The LUBM ontology contains several axioms that are equivalent to $A \sqsubseteq B \sqcap \exists R.C$ and $B \sqcap \exists R.C \sqsubseteq A$. Now the latter axiom contains a complex concept on the left-hand side of $\sqsubseteq$, so it is absorbed into an equivalent axiom $B \sqsubseteq A \sqcup \forall R.\neg C$. Whereas this is a definition axiom, it contains a disjunction on the right-hand side, and thus causes branch splits.

**Wine.** The Wine ontology is a fairly complex ontology, using advanced DL constructors such as disjunctions and equality. The translation of nominals is incomplete, so we ran only the following query: ? -

$$Q_{W_1}(x) \equiv AmericanWine(x)$$

The results show that the ontology complexity affects the performance: wine_0 is significantly smaller than, say, lubm_1, but the time required to answer the query is roughly the same. The degradation of performance in KAON2 is mainly due to disjunctions. On the theoretical side, disjunctions increase the data complexity of our algorithm from $\mathrm{P}$ to $\mathrm{NP}$[HMS05]. On the practical side, the technique for answering queries in disjunctive programs used in KAON2 should be further optimized.

## A.4 TBox Reasoning

Although TBox reasoning was not in the focus of our work, to better understand the limitations of our algorithms, we also conducted several TBox reasoning tests. In particular, we measured the time required to compute the subsumption hierarchies of Wine, DOLCE, and GALEN ontologies. Furthermore, we observed that a considerable source of complexity for KAON2 on DOLCE are the transitivity axioms, so we also performed the tests for a version of DOLCE in which all transitivity axioms were removed.

Our results indicate that the performance of TBox reasoning in KAON2 lags behind the performance of the state-of-the-art tableau reasoners. This should not come as a surprise: in the past decade, many optimization techniques were developed that optimize TBox reasoning in tableau algorithms; these techniques are not directly applicable to the resolution setting. Still, KAON2 can classify DOLCE without transitivity axioms, which is known to be a fairly complex ontology. Hence, we believe that developing additional optimization techniques for resolution algorithms might yield some interesting and practically useful results.

By analyzing the ontologies for which KAON2 was unable to compute the subsumption hierarchy within given resource limits, we noticed that they all contain many $\mathcal{ALCHIQ}$-clauses of types 3 and 7 with the same role symbol, which generate many consequences. This explains why KAON2 is not able to classify the original DOLCE ontology, but why it works well if the transitivity axioms are removed: the transformation used to deal with transitivity introduces axioms which, when clausified, produce many clauses of types 3 and 7.

# Bibliography


[Bac90] F. Bacchus. *Representing and Reasoning with Probabilistic Knowledge*. MIT Press, 1990.

[BADW04] Christopher Brewster, Harith Alani, Srinandan Dasmahapatra, and Yorick Wilks. Data-driven ontology evaluation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, Lisbon, 2004. European Language Resources Association.

[BGH+05] Janez Brank, Marko Grobelnik, Peter Haase, Dunja Mladenić, Johanna Völker, and Denny Vrandečić. A framework for ontology evaluation. In *submitted*, 2005.

[BJSSA04] A. Burton-Jones, V.C. Storey, V. Sugumaran, and P. Ahluwalia. A semiotic metrics suite for assessing the quality of ontologies. In *Data and Knowledge Engineering*, 2004.

[CV05] P. Cimiano and J. Voelker. A framework for ontology learning and data-driven change discovery. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB'2005)*, 2005.

[Dey01] A. K. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5(1):4–7, 2001.

[DP04] Z. Ding and Y. Peng. A probabilistic extension to ontology language OWL. In *Proceedings of the 37th Hawaii International Conference on System Sciences*, 2004.

[EGH+04a] M. Ehrig, T. Gabel, P. Haase, Y. Sure, C. Tempich, and J. Voelker. Data manual - initial version. SEKT informal deliverable 7.1.1.b, Institute AIFB, University of Karlsruhe, 2004.

[EGH+04b] M. Ehrig, T. Gabel, P. Haase, Y. Sure, C. Tempich, and J. Voelker. Use cases - initial version. SEKT informal deliverable 7.1.1.a, Institute AIFB, University of Karlsruhe, 2004.

[EHSH05] Marc Ehrig, Peter Haase, Nenad Stojanovic, and Mark Hefke. Similarity for ontologies - a comprehensive framework. In *13th European Conference on Information Systems*, MAY 2005.

[esw05] The semantic web: Research and applications. In C. Bussler, J. Davies, D. Fensel, and R. Studer, editors, *Second European Semantic Web Conference, ESWC 2005*, LNCS, Heraklion, Crete, Greece, May 2005. Springer.

[FBGL98] M.S. Fox, M. Barbuceanu, M. Gruninger, and J. Lin. An organization ontology for enterprise modelling. *Simulating organizations: Computational models of institutions and groups*, 1998.

[FLGPSS] M. Fernández-López, A. Gómez-Pérez, J. P. Sierra, and A. P. Sierra. Building a chemical ontology using Methontology and the Ontology Design Environment. *IEEE Intelligent Systems*, 14(1). 1999.

[GMF04] Ramanathan V. Guha, Rob McCool, and Richard Fikes. Contexts for the semantic web. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 2004.

[GP04] A. Gómez-Pérez. Ontology evaluation. In Staab and Studer [SS04], chapter 13, pages 251–274.

[GPH04] Y. Guo, Z. Pan, and J. Heflin. An Evaluation of Knowledge Base Systems for Large OWL Datasets. In *Proc. ISWC 2004*, volume 3298 of *LNCS*, pages 274–288, Hiroshima, Japan, November 7–11 2004. Springer.

[Gua98] N. Guarino. Formal ontology and information systems. volume 46 of *Frontiers in Artificial Intelligence and Applications*, Trento, Italy, 1998. IOS-Press.

[GW04] N. Guarino and C. A. Welty. An overview of OntoClean. In Staab and Studer [SS04], chapter 8, pages 151–172.

[Hea92] M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, 1992.

[HHSTS05] P. Haase, A. Hotho, L. Schmidt-Thieme, and Y. Sure. Collaborative and usage-driven evolution of personal ontologies. In Bussler et al. [esw05], pages 486–499.

[HM01] V. Haarslev and R. Möller. RACER System Description. In *Proc. IJCAR 2001*, volume 2083 of *LNAI*, pages 701–706, Siena, Italy, June 18–23 2001. Springer.

[HMS05] U. Hustadt, B. Motik, and U. Sattler. Data Complexity of Reasoning in Very Expressive Description Logics. In *Proc. IJCAI 2005*, pages 466–471, Edinburgh, UK, July 30 – August 5 2005.

[Hor97] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, UK, 1997.

[HPSvH03] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From shiq and rdf to owl: the making of a web ontology language. *J. Web Sem.*, 1(1):7–26, 2003.

[HS05a] P. Haase and L. Stojanovic. Consistent evolution of OWL ontologies. In Bussler et al. [esw05], pages 182–197.

[HS05b] P. Haase and Y. Sure. Usage tracking for ontology evolution. SEKT deliverable 3.2.1, Institute AIFB, University of Karlsruhe, 2005.

[HST00] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.

[HV05] Peter Haase and Johanna Völker. Ontology learning and reasoning – dealing with uncertainty and inconsistency. In *Proceedings of the Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, Nov 2005.

[HvHH+05] Peter Haase, Frank van Harmelen, Zhisheng Huang, Heiner Stuckenschmidt, and York Sure. A framework for handling inconsistency in changing ontologies. In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *Proceedings of the Fourth International Semantic Web Conference (ISWC2005)*, volume 3729 of *LNCS*, pages 353–367. Springer, NOV 2005.

[KN03] M. Klein and N. Noy. A component-based framework for ontology evolution. In *Proc. of the WS on Ont. and Distr. Sys., IJCAI '03*, Acapulco, Mexico, August9, 2003.

[LTGP04] A. Lozano-Tello and A. Gomez-Perez. ONTOMETRIC: A Method to Choose the Appropriate Ontology. *Journal of Database Management*, 15(2), 2004.

[Mil95] G. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[MS02] Alexander Maedche and Steffen Staab. Measuring similarity between ontologies. In *Proc. Of the European Conference on Knowledge Acquisition and Management - EKAW-2002. Madrid, Spain, October 1-4, 2002*, volume 2473 of *LNCS/LNAI*. Springer, 2002.

[PS04] B. Parsia and E. Sirin. Pellet: An OWL-DL Reasoner, Poster at ISWC 2004, Hiroshima, Japan, 2004, 2004.

[PSK05] B. Parsia, E. Sirin, and A. Kalyanpur. Debuging OWL ontologies. In *Proc. of the 14th World Wide Web Conference (WWW2005)*, Chiba, Japan, May 2005.

[RNG93] A. L. Rector, W. A. Nowlan, and A. Glowinski. Goals for concept representation in the galen project. In *Proc. SCAMC '93*, pages 414–418, Washington DC, USA, November 1–3 1993.

[SAS03] Y. Sure, J. Angele, and S. Staab. OntoEdit: Multifaceted inferencing for ontology engineering. *Journal on Data Semantics*, LNCS(2800):128–152, 2003.

[SHG03] N. Stojanovic, J. Hartmann, and J. Gonzalez. Ontomanager - a system for usage-based ontology management. In *In Proc. of FGML Workshop. SIG of German Information Society (FGML - Fachgruppe Maschinelles Lernen GI e.V.)*, 2003.

[SMMS02] Ljiljana Stojanovic, Alexander Mädche, Boris Motik, and Nenad Stojanovic. User-driven ontology evolution management. In *European Conf. Knowledge Eng. and Management (EKAW 2002)*, pages 285–300. Springer-Verlag, 2002.

[SS02a] N. Stojanovic and L. Stojanovic. Usage-oriented evolution of ontology-based knowledge management systems. In *Int. Conf. on Ontologies, Databases and Applications of Semantics, (ODBASE 2002), Irvine, CA*, LNCS, pages 230–242, 2002.

[SS02b] Y. Sure and R. Studer. On-To-Knowledge methodology. In J. Davies, D. Fensel, and F. van Harmelen, editors, *On-To-Knowledge: Semantic Web enabled Knowledge Management*, chapter 3, pages 33–46. J. Wiley and Sons, 2002.

[SS04] S. Staab and R. Studer, editors. *Handbook on Ontologies in Information Systems*. International Handbooks on Information Systems. Springer, 2004.

[SSGS03] L. Stojanovic, N. Stojanovic, J. Gonzalez, and R. Studer. Ontomanager - a system for the usage-based ontology management. In *ODBASE 2003*, volume 2888, pages 858–875. Springer, Dec 2003.

[Sto04] L. Stojanovic. *Methods and Tools for Ontology Evolution*. PhD thesis, University of Karlsruhe, 2004.

[STV+05] Y. Sure, C. Tempich, D. Vrandečić, H.S. Pinto, E. Paslaru Bontas, and M. Hefke. Sekt methodology: Evaluation of guidelines. SEKT deliverable 7.1.2, Institute AIFB, University of Karlsruhe, DEC 2005.

[Sup05] K. Supekar. A peer-review approach for ontology evaluation. In *8th Int. Protégé Conference, Madrid, Spain*, July 2005.

[TD03] O. Tamine and R. Dillmann. Kavido: a web-based system for collaborative research and development processes. *Computers in Industry*, 52(1):29–45, 2003.

[TPSS05] C. Tempich, H. S. Pinto, Y. Sure, and S. Staab. An argumentation ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies (DILIGENT). In Bussler et al. [esw05], pages 241–256.

[VNCN05] P. Velardi, R. Navigli, A. Cuchiarelli, and F. Neri. Evaluation of ontolearn, a methodology for automatic population of domain ontologies. In P. Buitelaar, P. Cimiano, and B. Magnini, editors, *Ontology Learning from Text: Methods, Applications and Evaluation*. IOS Press, 2005. to appear.

[Vol04] R. Volz. *Web Ontology Reasoning With Logic Databases*. PhD thesis, Universität Fridericiana zu Karlsruhe (TH), Germany, 2004.

[VS05] J. Voelker and Y. Sure. Data-driven change discovery. evaluation. SEKT deliverable 3.3.2, Institute AIFB, University of Karlsruhe, 2005.

[VVS05] J. Völker, D. Vrandečić, and Y. Sure. Automatic evaluation of ontologies (AEON). In *Proc. of the 4th International Semantic Web Conference (ISWC'05)*, Nov 2005.