

1. On-To-Knowledge: Semantic Web Enabled Knowledge Management

York Sure¹, Hans Akkermans⁶, Jeen Broekstra², John Davies³, Ying Ding⁶, Alistair Duke³, Robert Engels⁴, Dieter Fensel⁶, Ian Horrocks⁹, Victor Iosif⁵, Arjohn Kampman², Atanas Kiryakov⁷, Michel Klein⁶, Thorsten Lau⁸, Damyan Ognyanov⁷, Ulrich Reimer⁸, Kiril Simov⁷, Rudi Studer¹, Jos van der Meer², and Frank van Harmelen⁶

¹ Institute AIFB, University of Karlsruhe

Postfach, 76128 Karlsruhe, Germany

Contact: sure@aifb.uni-karlsruhe.de

Project website: <http://www.ontoknowledge.org>

² Administrator, Amersfoort, The Netherlands

³ BT, Ipswich, UK

⁴ CognIT, Oslo, Norway

⁵ EnerSearch AB, Gothenburg, Sweden

⁶ Free University Amsterdam, Amsterdam, The Netherlands

⁷ OntoText, Sofia, Bulgaria

⁸ Swiss Life, Zurich, Switzerland

⁹ University of Manchester, Manchester, UK

Summary.

On-To-Knowledge builds an ontology-based tool environment to improve knowledge management, dealing with large numbers of heterogeneous, distributed, and semi-structured documents typically found in large company intranets and the World Wide Web. The project's target results are: (i) a toolset for semantic information processing and user access; (ii) OIL, an ontology-based inference layer for the World Wide Web; (iii) an associated methodology and validation by industrial case studies. This chapter offers an overview of the *On-To-Knowledge* approach to knowledge management.

1.1 Introduction

The World Wide Web (WWW) has drastically changed the availability of electronically available information. Currently, there are around one billion documents in the WWW, which are used by more than 300 million users internationally. And that number is growing fast. However, this success and exponential growth makes it increasingly difficult to find, to access, to present, and to maintain the information required by a wide variety of users. Document management systems now on the market have severe weaknesses:

- *Searching information:* Existing keyword-based searches can retrieve irrelevant information that includes certain terms in different meanings. They also miss information when different terms with the same meaning about the desired content are used.

- *Extracting information:* Currently, human browsing and reading is required to extract relevant information from information sources. This is because automatic agents do not possess the common sense knowledge required to extract such information from textual representations, and they fail to integrate information distributed over different sources.
- *Maintaining* weakly structured text sources is a difficult and time-consuming activity when such sources become large. Keeping such collections consistent, correct, and up-to-date requires mechanized representations of semantics that help to detect anomalies.
- *Automatic document generation* would enable adaptive websites that are dynamically reconfigured according to user profiles or other aspects of relevance. Generation of semi-structured information presentations from semi-structured data requires a machine-accessible representation of the semantics of these information sources.

However, the competitiveness of many companies depends heavily on how they exploit their corporate knowledge and memory. Most information in modern electronic media is mixed media and rather weakly structured. This is not only true of the Internet but also of large company intranets. Finding and maintaining information is a tricky problem in weakly structured representation media. Increasingly, companies have realized that their intranets are valuable repositories of corporate knowledge. But as volumes of information continue to increase rapidly, the task of turning them into useful knowledge has become a major problem. Tim Berners-Lee envisioned a *Semantic Web* (cf. [1.1] [1.2]) that provides automated information access based on machine-processable semantics of data and heuristics that use these meta data. The explicit representation of the semantics of data, accompanied with domain theories (i.e., ontologies), will enable a Web that provides a qualitatively new level of service. It will weave together an incredibly large network of human knowledge and will complement it with machine processability. Various automated services will help the user achieve goals by accessing and providing information in machine-understandable form. This process may ultimately create extremely knowledgeable systems with various specialized reasoning services systems that can support us in nearly all aspects of life and that will become as necessary to us as access to electric power.

Ontologies (cf. [1.3]) are key enabling technology for the Semantic Web. They need to interweave human understanding of symbols with their machine processability. This clearly warrants a closer look at the nature of ontologies and at the question of whether they can actually provide such a service, and if so, how. Ontologies were developed in artificial intelligence to facilitate knowledge sharing and re-use. Since the early nineties, ontologies have become a popular research topic. They have been studied by several artificial intelligence research communities, including knowledge engineering, natural-language processing and knowledge representation. More recently, the concept of ontology is also gaining tremendous ground in fields, such as intel-

ligent information integration, cooperative information systems, information retrieval, electronic commerce, and knowledge management. The reason ontologies are becoming so popular is largely due to what they promise: *a shared and common understanding of a domain that can be communicated between people and application systems.*

The EU IST-1999-10132 project *On-To-Knowledge*¹ develops methods and tools to employ the full power of the ontological approach for facilitating knowledge management. The *On-To-Knowledge* tools will help knowledge workers to access company-wide information repositories in an efficient, natural and intuitive way.

The contents of this chapter are structured as follows. Section 1.2 begins with a description of the tool environment we developed in On-To-Knowledge to facilitate Semantic Web technology for information access and knowledge management. Section 1.3 goes on to examine the Ontology Inference Layer (OIL), which defines a flexible framework for defining ontology languages in the Semantic Web. Working in cooperation with our American colleagues at DAML², we defined the DAML+OIL language that has now been submitted as a Web standard at W3C³, the standardization committee for the World Wide Web. Section 1.4 discusses the methodological framework for our tool environment and its application in several industrial case studies, including large organizational memories and knowledge management in virtual enterprises. Finally, Section 1.5 presents our conclusions and general outlook.

1.2 Tool Environment for Ontology-based Knowledge Management

A key outcome of the On-To-Knowledge project is the resulting software toolset. Several consortium partners are participating in the effort to realize in software the underpinning ideas and theoretical foundations of the project. A major objective of the project is to create intelligent software to support users in both accessing information and in the maintenance, conversion, and acquisition of information sources. The tools are integrated in a three-layered architecture (cf. Figure 1.1). The layers consists of (i) the user front end layer on top, (ii) a middleware layer in the middle and (iii) an extraction layer at the bottom. Each tool represents certain functionalities. The layering allows for a modular design of applications that bundle some or all of the functionalities provided. Most of the tools presented in the figure are described subsequently below. As a minimum requirement all tools support OIL core that has been

¹ <http://www.ontoknowledge.org>

² <http://www.daml.org>

³ <http://www.w3c.org>

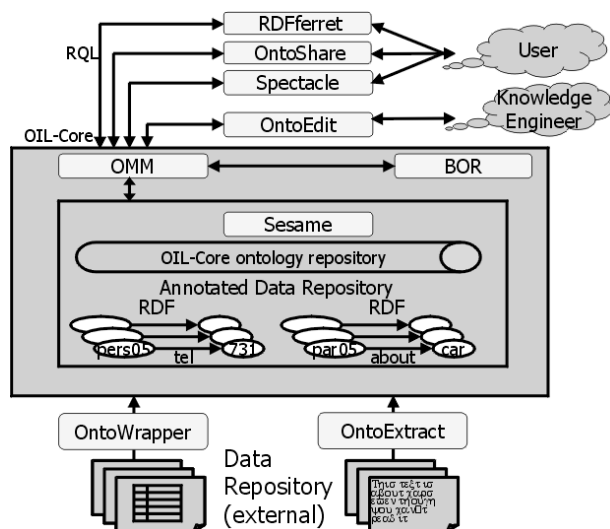


Fig. 1.1. The technical architecture of On-To-Knowledge

designed to be exactly the part of OIL that coincides with RDF(S)⁴ (cf. Section 1.3 for a detailed description).

1.2.1 *RDFferret*: Full Text Searching plus RDF Querying

RDFferret combines full text searching with RDF querying. This combined approach seems to be very promising due to the fact that RDF-annotated information resources are likely to be complemented by non-annotated information for a considerable period to come, and that any given RDF description of a set of resources will give one particular perspective on the information described. *RDFferret* can be used like a conventional Internet search engine by entering a set of search terms or a natural language query and produces a list of links to relevant Web pages in the usual way. However, *RDFferret*'s indexing and retrieval technique is also designed to use domain knowledge that is made available in the form of ontologies specified as RDF Schemas. RDF resources are Web pages or (parts thereof) and such pages or segments are effectively ontological instances. Correspondingly, resource types are ontological classes. The information items processed by *RDFferret* are RDF resources, which may be Web pages or parts thereof. During indexing *RDFferret* assigns content descriptors to RDF resources. Content descriptors of a resource are terms (words and phrases) that *RDFferret* obtains from a full text analysis of the resource content and from processing all literal values that are directly

⁴ cf. <http://www.w3.org/TR/REC-rdf-syntax/> and <http://www.w3.org/TR/rdf-schema/>

related by a property. They also retain structural information about the ontology. In *RDFferret* the user can select from a list of all the resource types stored in the index. When searching by selecting a resource type, *RDFferret* adjusts its result list to show only resources of the selected type. The user is also presented with a search and navigation area. The search area shows the attributes of the selected resource type. For each attribute the user can input a search criterion. *RDFferret* combines the search criteria entered and matches the resulting query against its ontology-based index. In addition, resource types (ontological classes) related by some property to the currently selected type are displayed as hyperlinks. Clicking on such a type then selects that type and in turn displays those types that are related to it. Thus the user can browse the ontology in a natural and intuitive way.

Figure 1.2 shows a typical initial query by a user. The user has entered a free text query for information about an employee called “George Miller”. The search engine has returned a ranked list of 73 documents mentioning the terms “George” and/or “Miller”. At the top of the screenshot can be seen a drop-down list containing the selection “any...”. When returning the 73 results documents, *RDFferret* has also compiled a list of the classes to which each document belongs. This class list is then made available to the user via the drop-down list referred to. The user can then refine the search results by selecting one of the classes from the list.

1.2.2 **OntoShare: Community Support**

OntoShare enables the storage of best practice information according to an ontology and the automatic dissemination of new best practice information to relevant co-workers. It also allows users to browse or search the ontology in order to find the most relevant information to the problem that they are dealing with at any given time. The ontology helps new users to navigate and acts as a store for key learning and best practices accumulated through experience. In addition, the ontology helps users to become familiar with new domains. It provides a sharable structure for the knowledge base, and a common language for communication between user groups. Each user can define his own relevant parts of the ontology (i.e. personal concepts) that are integrated into a single coherent ontology available to all users (cf. Figure 1.3).

1.2.3 **Spectacle: Information Presentation**

Spectacle is a content presentation platform featuring custom-made information presentations, aimed at supporting the information needs of its users. This means not only that the right information should be delivered to the user, but also that it needs to be presented (structured, formatted, rendered) in a manner appropriate for that specific user.

Spectacle is used to disclose both the content of databases, document repositories and other enterprise information sources, as well as the semantics

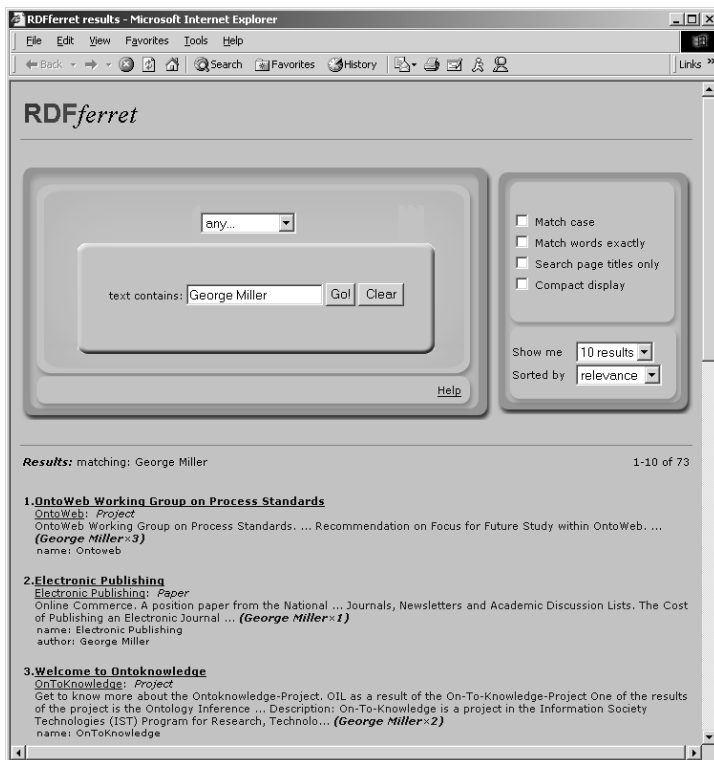


Fig. 1.2. The query interface of RDFferret

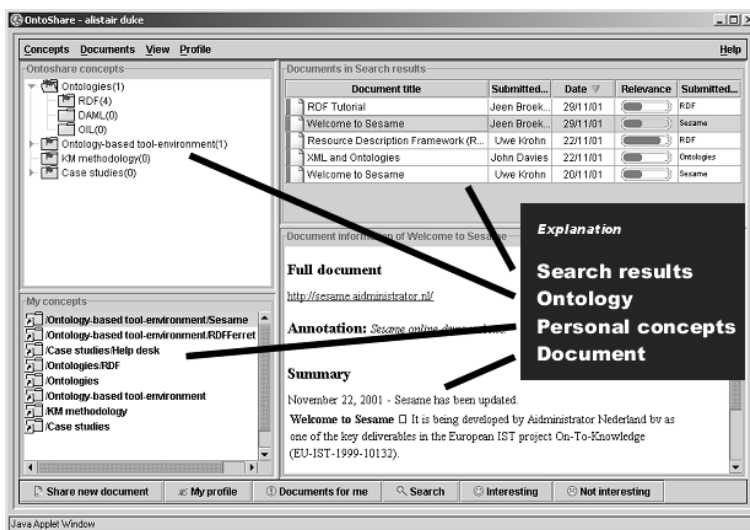


Fig. 1.3. Community of knowledge sharing

of that information from Semantic Web resources. The platform consists of the Spectacle server and programming libraries for generating both Web-based and graphical information presentations.

For the end user, Spectacle transforms the task of gathering information from a search task (formulating explicit queries) to a browsing task (using navigation heuristics) by presenting each user with the navigational means appropriate for his or her task. This results in more efficiency in retrieving the right information, both in terms of retrieval accuracy as well as time spent on the task.

Spectacle can present information in two different ways: (i) it can create *hypertext interfaces*, containing selected content, design and an appropriate navigation structure, based on the semantics of the information (cf. Figure 1.4), (ii) it can present the information by *graphical visualization* (cf. Figure 1.5).

A key benefit of the first approach is that it allows for an easy and flexible presentation of the same information in different ways, for each of the envisioned tasks or user groups. Furthermore, it has all the usual benefits of a generated Web site (like having a consistent design, being up-to-date) and it also takes advantage of the expressivity and flexibility provided by Semantic Web standards such as RDF, RDF Schema and DAML+OIL.

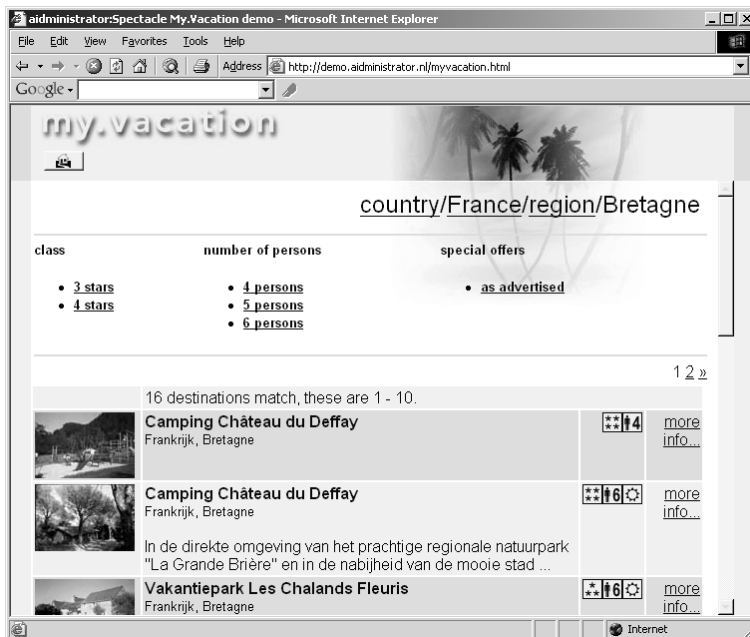


Fig. 1.4. Provision of navigational structures with Spectacle

A benefit of the second approach is that it can offer insights and kinds of information access that are not possible with conventional publishing methods such as Web sites. For example, overview and analysis of large sets of objects requires an effective and compact graphical presentation. Similarly, presentation of the relations between these objects is virtually impossible without the support of a graphical visualization.

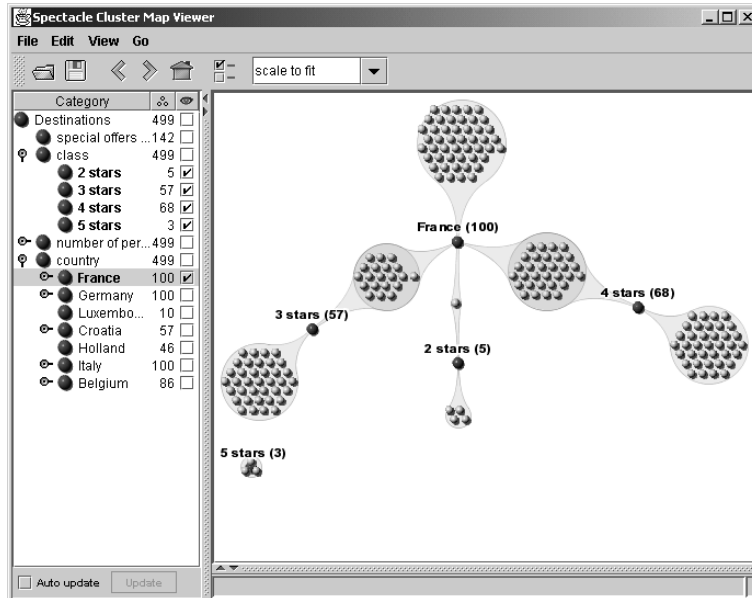


Fig. 1.5. Visualization of object sets with Spectacle

1.2.4 OntoEdit: Ontology Development

OntoEdit [1.4] is a collaborative ontology engineering environment that is easily expandable through a flexible plug-in framework. OntoEdit supports ontology engineers while inspecting, browsing, codifying and modifying ontologies in each step of the ontology development process (cf. Figure 1.11). Modeling ontologies using OntoEdit involves modelling at a conceptual level, viz. (i) as independently of a concrete representation language as possible, and (ii) using GUI's representing views on conceptual structures (concepts, concept hierarchy, relations, axioms) rather than codifying conceptual structures in ASCII. In addition, OntoEdit provides a simple instance editor to insert facts according to a modelled ontology. The conceptual model of an ontology is stored internally using a powerful ontology model, which can be mapped onto different, concrete representation languages (e.g. DAML+OIL

or RDF(S)). Ontologies can be directly imported from and exported to Sesame. Figure 1.6 shows an example from the Swiss Life case study described in Section 1.4.

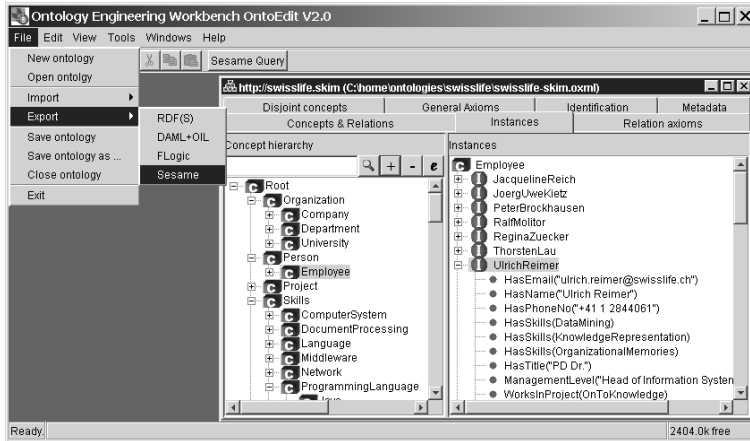


Fig. 1.6. Ontology development with OntoEdit

Collaboration is supported on different levels. Firstly, to facilitate the collaboration *between domain experts and ontology engineers* we developed several plug-ins (the plug-in framework is described in [1.5]) for OntoEdit that extend its functionalities, e.g. OntoKick and Mind2Onto. **OntoKick** targets at (i) creation of the requirement specification document for ontologies and (ii) extraction of relevant structures for the building of a semi-formal ontology description. **Mind2Onto** targets at the integration of brainstorming processes to build relevant structures of the semi-formal ontology description. Secondly, to support the *distributed development of ontologies* we implemented a transaction management based on a client/server architecture in which the OntoEdit clients connect to an ontology server and can change or extend the ontology. All clients are immediately informed of modifications the other ontologists do to the ontology. Engineers can store comments (e.g. explaining design decisions) in a documentation field for each concept and relation. By this way, one of the main features of ontologies, i.e. their consensual character, is supported. Collaborating ontologists must agree on the modelling decisions that are made. Therefore the possibility to monitor the development process of all collaborators is essential for reaching the goal of a shared ontology. We refer to [1.4] for a detailed description of both the plug-ins and the support for distributed ontology development.

1.2.5 Ontology Middleware Module: Integration Platform

The **Ontology Middleware Module** (OMM, cf. [1.6]) can be seen as “administrative” software infrastructure that makes the knowledge management tools easier for integration in real-world applications. The major features supported are:

- Change management for ontologies allows work with, revert to, extraction, and branching of different states and versions (cf. e.g. the next subsection on *OntoView*);
- Access control (security) system with support for role hierarchies including comprehensive and precise restrictions (down to object/record-level) that enable business-logic enforcement;
- Meta-information for ontologies, specific resources (classes, instances), and statements.

These three aspects are tightly integrated to provide the same level of the handling of knowledge in the process of its development and maintenance as source control systems (such as e.g. the Concurrent Versions System (CVS)⁵) provide for software. On the other hand, for end-user applications, OMM can be seen as equivalent to the database security, change tracking and auditing systems. Our OMM is carefully designed to support both use cases.

In a nutshell, OMM extends the storage and query facilities of Sesame with a **Knowledge Control System** (KCS, cf. Figure 1.7), additional support for multi-protocol access (e.g. HTTP, RMI, SOAP) and reasoning services.

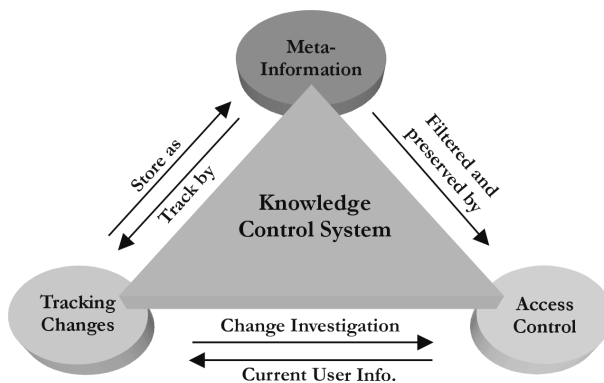


Fig. 1.7. Features of the Knowledge Control System

An example for a reasoning service is **BOR** – a reasoner that is currently being developed and complies with the DAML+OIL model-theoretic semantics. It is a modular component that can be plugged in to extend the query

⁵ <http://www.cvshome.org/>

facilities already provided e.g. by Sesame. It addresses most of the classic reasoning tasks for description logics, including realization and retrieval. Few innovative services, such as model checking and minimal ontology extraction, are also integral part of the system. The full set of functional interfaces will allow a high level of management and querying of DAML+OIL ontologies.

1.2.6 OntoView: Change Management for Ontologies

OntoView [1.7] is a change management tool for ontologies and is part of the Ontology Middleware Module. Change management is especially important when ontologies will be used in a decentralized and uncontrolled environment like the Web, where changes occur without co-ordination. The main function of OntoView is to provide a transparent interface to arbitrary versions of ontologies. To achieve this, it maintains an internal specification of the relation between the different variants of ontologies. This specification consists of three aspects: (i) the *meta-data* about changes (author, date, time etc), (ii) the *conceptual relations* between versions of definitions in the ontologies, and (iii) the *transformations* between them. This specification is partly derived from the versions of ontologies themselves, but also uses additional human input about the meta-data and the conceptual effects of changes.

To help the user to specify this information, OntoView provides the utility to compare versions of ontologies and highlight the differences. This helps in finding changes in ontologies, even if those have occurred in an uncontrolled way, i.e., possibly by different people in an unknown order. The comparison function is inspired by UNIX `diff`, but the implementation is quite different. Standard `diff` compares file version at line-level, highlighting the lines that textually differ in two versions. OntoView, in contrast, compares version of ontologies at a *structural* level, showing which definitions of ontological concepts or properties are changed.

There are different types of change. Each type is highlighted in a different color, and the actually changed lines are printed in boldface. An example of the visual representation of the result of a comparison is shown in Figure 1.8.

The comparison function distinguishes between the following types of change:

- Non-logical change, e.g. in a natural language description. This are changes in the label of an concept or property, or in comment inside definitions.
- Logical definition change. This is a change in the definition of a concept that affects its formal semantics. Examples of such changes are alterations of subclass statements, or changes in the domain or range of properties. Additions or deletions of local property restriction in a class are also logical changes. The second and third change in the Figure 1.8 (class “Male” and property “hasParent”) are examples of such changes.
- Identifier change. This is the case when a concept or property is given a new identifier, i.e. a renaming.

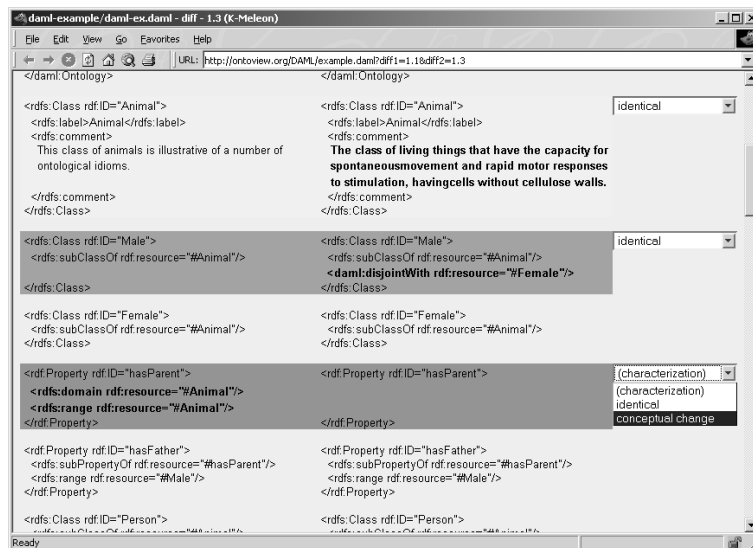


Fig. 1.8. The result of a comparison of two ontologies with OntoView

- Addition of definitions.
- Deletion of definitions.

The comparison function also allows the user to specify the conceptual implication of the changes. For the first three types of changes, the user is given the option to label them either as “identical” (i.e., although the specification is changes, it still refers to the same concept), or as “conceptual change”. In the latter case, the user can specify the conceptual relation between the two version of the concept. For example, by stating that the property “hasParent_{1.0}” is a sub-property of “hasParent_{2.0}”.

Another function is the possibility to analysis effects of changes. Changes in ontologies do not only affect the data and applications that use them, but they can also have unintended, unexpected and unforeseeable consequences in the ontology itself [1.8]. The system provides some basic support for the analysis of these effects. First, on request it can also highlight the places in the ontology where conceptually changed concepts or properties are used. For example, if a property “hasChild” is changed, it will highlight the definition of the class “Mother”, which uses the property “hasChild”. This function can also exploit the transitivity of properties to show the propagation of possible changes through the ontology. A foreseen second effect analysis feature is a connection to FaCT (cf. [1.9]), which allows to check the formal consistency of the suggested conceptual relations between different versions of definitions.

1.2.7 Sesame: Repository for Ontologies and Data

Sesame [1.10] is a system that allows persistent storage of RDF data and schema information and subsequent online querying of that information. Sesame has been designed with scalability, portability and extensibility in mind. Sesame itself has been implemented in Java, which makes it portable to almost any platform. It also abstracts from the actual repository used by means of a standardized API. This API makes Sesame portable to any repository (DBMS or otherwise) that is able to store RDF triples. Currently, only implementations based on DBMS's exist. At the same time, this API enables swift addition of new modules that operate on RDF and RDF Schema data. One of the most prominent modules of Sesame is its query engine. This query engine supports an OQL-style query language called RQL (cf. [1.11]). RQL supports querying of both RDF data (e.g. instances) and schema information (e.g. class hierarchies, domains and ranges of properties). RQL also supports path-expressions through RDF graphs, and can combine data and schema information in one query. The streaming approach used in Sesame (data is processed as soon as available) makes for a minimal memory footprint. This streaming approach also makes it possible for Sesame to scale to huge amounts of data. In short, Sesame can scale from devices as small as palm-top computers to powerful enterprise servers. A final feature of Sesame is its flexibility in communicating with other tools. Currently, Sesame itself only supports communication over HTTP, support for other protocols is added through the Ontology Middleware Module on top of it. Sesame has now been released as Open Source under the GNU Lesser General Public License (LGPL)⁶.

1.2.8 CORPORUM: Information Extraction

The **CORPORUM toolset** [1.12] consists of two parts, viz. **OntoExtract** and **OntoWrapper**, and has two related, though different, tasks: interpretation of natural language texts and extraction of specific information from free text. Whereas the former process can be performed autonomously by CORPORUM tools, the latter task requires a user who defines business rules for extracting information from tables, (phone) directories, home-pages, etc. Although this task is not without its challenges, most effort focuses on the former task, which involves natural language interpretation on a syntactic and lexical level, as well as interpretation of the results of that level (discourse analysis, co-reference and collocation analysis, etc.). The CORPORUM system outputs a variety of (symbolic) knowledge representations, including semantic (network) structures and visualizations thereof, light-weight ontologies, text summaries, automatically generated thesauri (related words/concepts), etc. Thus, extracted information is represented in

⁶ Sesame is available for download at the Sourceforge project website: <http://sourceforge.net/projects/sesame/>

RDF(S)/DAML+OIL, augmented with Dublin Core Meta Data wherever possible, and submitted to the Sesame data repository mentioned previously.

Typically, the CORPORUM system does not incorporate background knowledge itself, but relies on its extraction and analysis capabilities in combination with any knowledge available in the Sesame repository. The availability of knowledge, however, is not a prerequisite.

1.3 OIL: Inference Layer for the Semantic World Wide Web

In the introduction, ontologies were identified as a key technology for making progress in the tasks outlined there: searching, extracting and maintaining information in a weakly-structured environments such as company intranets, or even in an open environment like the WWW.

The tools discussed in Section 1.2 all exploited ontologies as their common operating ground: an ontology was created and refined manually (OntoEdit) or extracted semi-automatically (OntoExtract), raw information sources were structured on the basis of an ontology (OntoWrapper), this structured data was stored in an ontology-based repository (Sesame), and the data could be queried using the vocabulary from an ontology. Finally, information could be shared (OntoShare), searched (RDF*ferret*) or browsed (Spectacle) by users on the basis of such ontological vocabularies.

All of this of course requires the existence of a language to express such ontologies. Some basic requirements for such a language are:

- sufficient expressivity for the applications and tasks mentioned in the preceding sections
- sufficiently formalized to allow machine processing
- integrated with existing Web technologies and standards

Although much work has been done on ontology languages in the AI community (see e.g. [1.13] for a recent overview), it is particularly the 3rd requirement that motivated us to design a new language (baptized OIL) for our purposes. In this section, we will briefly describe the constructions in the OIL language, and then discuss its most important features and design decisions.

1.3.1 Combining Description Logics with Frame Languages

The OIL language (cf. [1.14], [1.15], and [1.16]) is designed to combine frame-like modelling primitives with the increased (in some respects) expressive power, formal rigor and automated reasoning services of an expressive description logic. OIL also comes “Web enabled” by having both XML and RDFS based serializations (as well as a formally specified “human readable”

form, which we will use here⁷). The frame structure of OIL is based on XOL [1.17], an XML serialization of the OKBC-lite knowledge model [1.18]. In these languages classes (concepts) are described by frames, which consist of a list of super-classes and a list of slot-filler pairs. A slot corresponds to a role in a DL, and a slot-filler pair corresponds to either a universal value restriction or an existential quantification. OIL extends this basic frame syntax so that it can capture the full power of an expressive description logic. These extensions include:

- Arbitrary boolean combinations of classes (called class expressions) can be formed, and used anywhere that a class name can be used. In particular, class expressions can be used as slot fillers, whereas in typical frame languages slot fillers are restricted to being class (or individual) names.
- A slot-filler pair (called a slot constraint) can itself be treated as a class: it can be used anywhere that a class name can be used, and can be combined with other classes in class expressions.
- Class definitions (frames) have an (optional) additional field that specifies whether the class definition is primitive (a subsumption axiom) or non-primitive (an equivalence axiom). If omitted, this defaults to primitive.
- Different types of slot constraint are provided, specifying universal value restrictions, existential quantification and various kinds of cardinality constraint.
- Global slot definitions are extended to allow the specification of superslots (subsuming slots) and of properties such as transitivity, and symmetry.
- Unlike many frame languages, there is no restriction on the ordering of class and slot definitions, so classes and slots can be used before they are defined.
- OIL also provides axioms for asserting disjointness, equivalence and coverings with respect to class expressions.

Many of these points are standard for a description logic, but are novel for a frame language. OIL is also more restrictive than typical frame languages in some respects. In particular, it does not support collection types other than sets (e.g., lists or bags), and it does not support the specification of default fillers. These restrictions are necessary in order to maintain the formal properties of the language (e.g., monotonicity) and the correspondence with description logics.

1.3.2 Web Interface

As part of the Semantic Web activity of the W3C, a very simple Web-based ontology language had already been defined, namely RDF Schema. This language only provides facilities to define class- and property-names, inclusion axioms for both classes and properties (subclasses and subproperties), and to

⁷ <http://www.ontoknowledge.org/oil/syntax/>

define domain and range constraints on properties. Instances of such classes and properties are defined in RDF.

OIL has been designed to be a superset of the constructions in RDF Schema: all valid RDF Schema expressions are also valid OIL expressions. Furthermore, the syntax of OIL has been designed such that any valid OIL document is also a valid RDF(S) document when all the elements from the OIL-namespace are ignored. The RDF Schema interpretation of the resulting subdocument is guaranteed to be sound (but of course incomplete) with respect to the interpretation of the full OIL document.

This guarantees that any RDF Schema agent can correctly process arbitrary OIL documents, and still correctly capture some of the intended meaning. The full details of how this has been achieved, and the trade-offs involved in this can be found in [1.19].

1.3.3 Layering

For many of the applications from section 1, it is unlikely that a single language will be ideally suited for all uses and all users. In order to allow users to choose the expressive power appropriate to their application, and to allow for future extensions, a layered family of OIL languages has been described. The sublanguage OIL Core has been defined to be exactly the part of OIL that coincides with RDF(S). This amounts to full RDF(S), without some of RDF's more dubious constructions: containers and reification.

The standard language, is called "Standard OIL", and when extended with the ability to assert that individuals and tuples are, respectively, instances of classes and slots), is called "Instance OIL". Finally, "Heavy OIL" is the name given to a further layer that will include as yet unspecified language extensions. This layering is depicted in Figure 1.9.

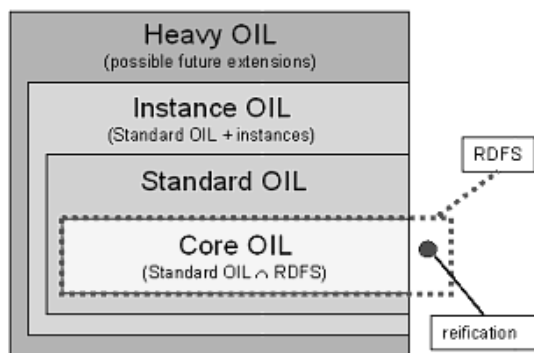


Fig. 1.9. The layered language model of OIL

Figure 1.10 illustrates an OIL ontology (using the human readable serialization), developed in a skills management case study by Swiss Life.

```

class-def Department
instance-of ITDept Department
class-def Skills
  slot-constraint SkillsLevel cardinality 1
slot-def HasSkills
  domain Employee
  range Skills
slot-def WorksInProject
  domain Employee
  range Project
  inverse ProjectMembers
class-def defined ITProject
  subclass-of Project
  slot-constraint ResponsibleDept has-value ITDept
slot-def ManagementLevel
  domain Employee
  range one-of "member" "head-of-group"
              "head-of-dept" "CEO"

class-def Publishing
  subclass-of Skills
class-def DocumentProcessing
  subclass-of Skills
class-def DesktopPublishing
  subclass-of Publishing and DocumentProcessing
instance-of GeorgeMiller Employee
related HasSkills GeorgeMiller DesktopPublishing3
instance-of DesktopPublishing3 DesktopPublishing
related SkillsLevel DesktopPublishing3 3

```

Fig. 1.10. OIL illustration

The following points are noteworthy:

- **Skills** are restricted to being of a single level through a cardinality constraint
- **WorksInProject** and **ProjectMembers** are defined to be each others inverse
- **ITProjects** are defined to be exactly those projects whose **ResponsibleDept** is the **ITDept**
- **DeskTopPublishing** is defined to be in the intersection of **Publishing** and **DocumentProcessing**

1.3.4 Current Status

Meanwhile, OIL has been adopted by a joined EU/US initiative that developed a language called DAML+OIL⁸, which has now been submitted to the Ontology Working Group of the W3C⁹, the standardization committee of the

⁸ For information about DAML+OIL cf. <http://www.daml.org/2001/03/daml+oil-index> and <http://www.w3.org/TR/daml+oil-reference>.

⁹ cf. <http://www.w3.org/2001/sw/WebOnt/> for information about the Ontology Working Group, cf. <http://www.w3.org/TR/webont-req/> for the latest W3C working draft on requirements for a Ontology Web Language

WWW. We can soon expect a recommendation for a Web ontology language; hopefully, it will feature many of the elements and aspects on which OIL is based.

1.3.5 Future Developments

In November 2001, the W3C started a Working Group (WG) for defining a Web Ontology language. This WG is chartered to take DAML+OIL as its starting point, now continuing on the evolving standard OWL (Ontology Web Language). Over 40 of the W3C members from academia and industry are currently participating in this effort. It is most likely that such a Web Ontology language will range in power somewhere between the rather simple RDF Schema and the rather rich Standard OIL language.

Other efforts are underway to define extensions for the ontology language, such as an ontology-query language, or an extension with rules (which would allow for example role chaining, as done in Horn logic).

1.4 Business Applications in Semantic Information Access

Tool development in On-To-Knowledge was strongly driven by several case studies. Some of them started in the earliest stages of the project so that the project was geared towards practical demands from the very beginning. The case studies also serve as a means to evaluate the project results. In addition, a methodology for employing ontology-based tools for knowledge management was developed and tested together with the case studies.

1.4.1 On-To-Knowledge Methodology

The On-To-Knowledge case studies explore a broad bandwidth of knowledge management challenges. Each of them has its own characteristics and usually evaluates due to his nature only a subset of the generic methodology. The path of an On-To-Knowledge application driven ontology development process is sketched in Figure 1.11. This methodology for employing ontology-based tools for knowledge management applications was developed by applying an initial baseline methodology (cf. [1.20]) based on CommonKADS (cf. [1.21]) in the case studies and continuously improving it with the insight gained from experience. First lessons learned from applying the methodology in the skills management case study are presented in [1.22].

The main stream indicates activities that finally lead to a refined, evaluated and applied ontology that has to be maintained, i.e. *feasibility study*, *ontology kickoff*, *refinement*, *evaluation* and *maintenance and evolution*. Each flag indicates major outcomes of the related activity, i.e. the CommonKADS

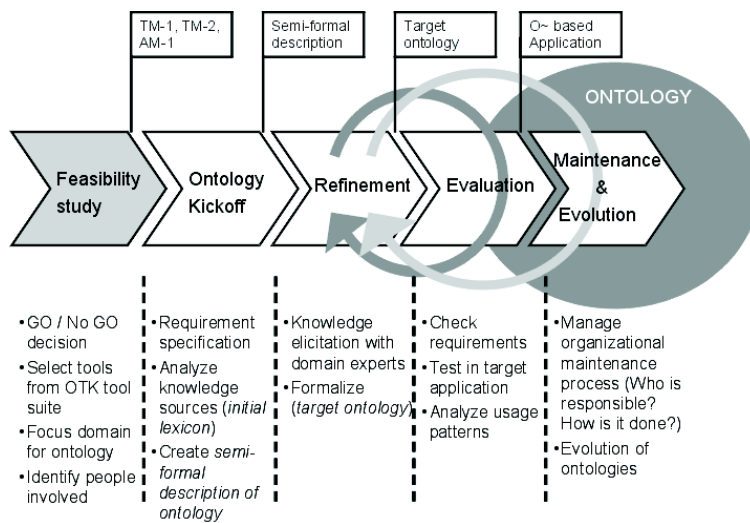


Fig. 1.11. Methodology for On-To-Knowledge

worksheets *TM-1*, *TM-2*, *AM-1*, *semi-formal description*, *target ontology* and *ontology-based application*. Below every activity the most important steps of the activity are sketched. Refinement, evaluation and maintenance may need to be performed in iterative cycles.

The approach consists of five major phases. It starts with a feasibility study (based on CommonKADS) to identify the parties involved, to focus the domain for the ontology-based application and to support go/no-go decisions for projects. During the kick-off phase, the requirement specification for the ontology-based application is acquired. It contains, among other things, valuable knowledge sources (e.g. domain experts identified during the feasibility study) and documents (e.g. index lists useful in building the ontology). Through analysis of the knowledge sources, a baseline ontology is developed that usually contains the most relevant concepts and relations of the focused domain and is modelled on a conceptual level. During the next phase, the refinement, knowledge is elicited with domain experts, which serves to enlarge the ontology with more fine-grained concepts and relations of the domain. The approach ends with a formalization phase, where the refined ontology is transferred into a formal representation language, such as OIL. This target ontology serves as a base for developing a prototype application to evaluate the target ontology in the next phase, the evaluation. The prototype helps to check whether the initial requirements from the kick-off phase are fulfilled. It may be necessary to perform several refinement-evaluation cycles before all requirements are met, which leads to the deployment of the ontology within the target application.

As the real world continues to change, so do the specifications for ontologies. To reflect these changes, ontologies must be maintained frequently, as are other software components. One point that we should stress here is that the maintenance of ontologies is primarily an organizational process. Ontologies require strict rules for their update-delete-insert processes. We recommend that ontology engineer group changes to the ontology and initiate the switch-over to a new version of the ontology after thoroughly testing possible effects to the application, viz. performing additional cyclic refinement and evaluation phases.

1.4.2 Information Search

Two of the case studies were carried out by Swiss Life. One of these approached the problem of finding relevant passages in a very large document about the International Accounting Standard (IAS) on the extranet (over 1000 pages). This document is used by accountants who need to know certain aspects of the IAS accounting rules. As the IAS standard uses very strict terminology, it is only possible to find relevant text passages when the correct terms are used in the query. Very often, this leads to poor search results. With the help of the ontology extraction tool OntoExtract, an ontology was automatically learned from the document. The ontology consists of 1,500 concepts linked by 47,000 weighted semantic associations. It supports users in reformulating their initial queries when the results fall short of expectations. This is done by offering terms from the ontology that are strongly associated with (one of) the query terms used in the initial query. An evaluation of user behavior showed that 70% of the queries involved a reformulation step. On average, 1.5 refinements were made. Thus, although the ontology is structurally quite simple, it greatly improves search results. Another advantage to using a simple ontology is that it requires no manual effort to build.

1.4.3 Skills Management

Swiss Life's second case study is a skills management application (cf. [1.22]) that uses manually constructed ontologies about skills, job functions, and education. These consist of 800 concepts with several attributes, arranged into a hierarchy of specializations. There are also semantic associations between these concepts. Additionally the skills management system integrates numerous information sources like an address-database to support employees while creating their own personal homepage on the company's intranet. The homepage includes information about personal skills, job functions, and education. The ontology allows for a comparison of skills descriptions among employees, and ensures the use of uniform terminology in skills descriptions and queries for employees with certain skills. Moreover, the ontology can automatically

extend queries with more general, more specialized, or semantically associated concepts. This enables controlled extension of search results, where necessary.

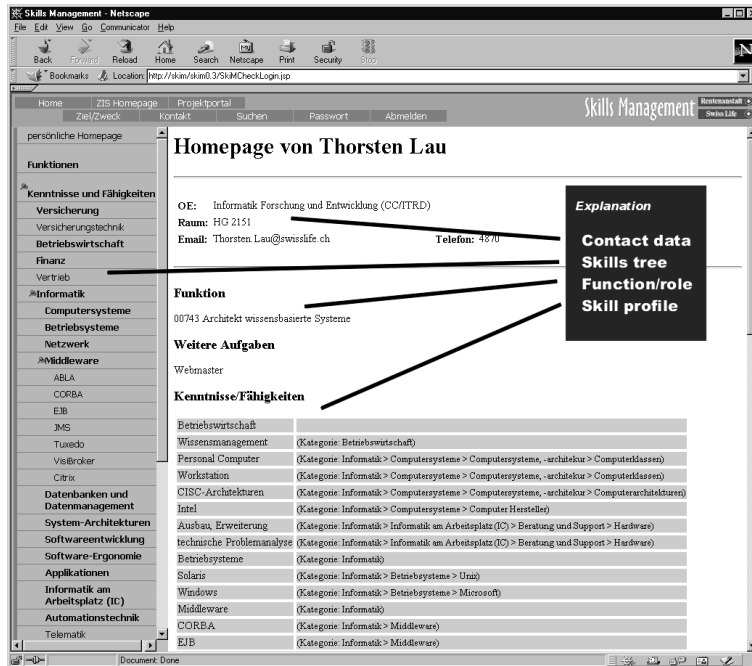


Fig. 1.12. Skills management case study at Swiss Life

1.4.4 Exchanging Knowledge in a Virtual Organization

The case study done by EnerSearch AB focuses on validating the industrial value of the project's results with respect to the needs of a virtual organization. The goal of the case study is to improve knowledge transfer between EnerSearch's in-house researchers and outside specialists via the existing website. The study also aims to help the partners from shareholding companies to obtain up-to-date information about research and development results.

The main problem with the current website is that its search engine supports free text searches rather than content-based information retrieval, which makes it fairly difficult to find information on certain topics. To remedy this, the entire Web site was annotated by concepts from an ontology developed using semi-automatic extraction from documents on the EnerSearch's current website. The *RDFferret* search engine is used to extend free text searches to searches of annotations. Alternatively, the Spectacle tool enables

users to obtain search results arranged into topic hierarchies, which can then be browsed. This offers users a more explorative route to finding the information the need.

The case study evaluation will start with pre-trial interviews of the users, followed by a test of the complete system. The third and final step will then be the post trial interviews to evaluate the improvements made. Three groups with different interests and needs will be involved in the evaluation: (i) researchers from different fields, (ii) specialists from the shareholders organization and (iii) outsiders from different fields.

1.5 Conclusions

The Web and company intranets have boosted the potential for electronic knowledge acquisition and sharing. Given the sheer size of these information resources, there is a strategic need to move up in the data - information - knowledge chain. On-To-Knowledge takes a necessary step in this process by providing innovative tools for semantic information processing and thus for much more selective, faster, and meaningful user access. This environment deals with three aspects:

- Acquiring ontologies and linking them to large amounts of data. For reasons of scalability, this process must be automated based on information extraction and natural language processing technology. To ensure quality, the process also requires human input in terms of building and manipulating ontologies based on ontology editors.
- Storing and maintaining ontologies and their instances. We developed an RDF Schema repository that provides database technology and simple forms of reasoning over Web information sources.
- Querying and browsing semantically enriched information sources. We developed semantically enriched search engines, browsing and knowledge sharing support that makes use of machine processable semantics of data.

The technology developed has been proven to be useful in a number of case studies. We can improve information access in the large intranets of sizeable organizations. The technology has been used to facilitate electronic knowledge sharing and re-use for customer relationship management and knowledge management in virtual organizations.

We also encountered a number of shortcomings in our current approach. Ontologies help to establish consensual terminologies that make sense to both sites. Computers are able to process information based on their machine-processable semantics. Humans are able to make sense of that information based on their knowledge of real-world semantics. Building ontologies that are a pre-requisite for – and result of – the common understanding of large user groups is no trivial task. A model or “protocol” for driving the network that

maintains the process of evolving ontologies is the real challenge for making the semantic Web a reality. Most work on ontologies views them in terms of an isolated theory containing a potentially large number of concepts, relationships, and constraints that further detach formal semantics from them. To tap into the full potential advantages they offer the Semantic Web, ontologies must be structured as interwoven networks that make it possible to deal with heterogeneous needs in the communication processes that they are supposed to mediate. Moreover, these ontologies need to shift over time because the processes they mediate are based on consensual representation of meaning. It is the network of ontologies and their dynamic nature that make future research necessary. Actual challenges in the current work on ontologies are what glue ontology networks together in space and time. Instead of a central, top-down process, we require a distributed process of emerging and aligned ontologies. Most existing technology focuses on building ontologies as graphs based on concepts and relationships. Our current understanding is far below par when it comes to proper methodological and tool support for building up networks, where the nodes represent small and specialized ontologies. This is especially true of the noisy and dynamically changing environment that the Web is and will continue to be.

References

- 1.1 T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web, Scientific American, May 2001.
- 1.2 D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster (eds.). Spinning the Semantic Web, MIT Press, Boston, to appear 2002.
- 1.3 D. Fensel. Ontologies: Silver bullet for knowledge management and electronic commerce. Springer-Verlag, Berlin, 2001.
- 1.4 Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer and D. Wenke. OntoEdit: Collaborative ontology engineering for the Semantic Web. In: Proceedings of the International Semantic Web Conference (ISWC) 2002, June 9-12 2002, Sardinia, Italia, 2002, LLNCS, Springer.
- 1.5 S. Handschuh. Ontoplugins - a flexible component framework. Technical report, University of Karlsruhe, May 2001.
- 1.6 A. Kiryakov, K. Iv. Simov and D. Ognyanov. Ontology Middleware: Analysis and design, Deliverable 38, On-To-Knowledge project, March 2002.
- 1.7 M. Klein and D. Fensel. OntoView – Web-based ontology versioning. Submitted, draft at <http://www.cs.vu.nl/~mcaklein/papers/ontoview.pdf>, 2002.
- 1.8 D. L. McGuinness, R. Fikes, J. Rice and Steve Wilder. An environment for merging and testing large ontologies. In Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000), pages 483–493, San Francisco, USA. Morgan Kaufmann, 2000.
- 1.9 I. Horrocks. Using an expressive description logic: FaCT or fiction? In: Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy, June 2-5, 1998, pages 636649. Morgan Kaufmann, 1998.
- 1.10 J. Broekstra, A. Kampman, F. van Harmelen. Sesame: An architecture for storing and querying RDF data and schema information. In [1.2], to appear.

- 1.11 G. Karvounarakis, V. Christophides, D. Plexousakis and S. Alexaki. Querying RDF Descriptions for Community Web Portals. In: Proceedings of The French National Conference on Databases 2001 (BDA'01), pp. 133-144, Agadir, Maroc, 29 October - 2 November, 2001.
- 1.12 R. Engels and B.A. Bremdal. CORPORUM: A workbench for the Semantic Web. Semantic Web Mining workshop. PKDD/ECML - 01. Freiburg, Germany, 2001.
- 1.13 O. Corcho and A. Gomez Perez. A roadmap to ontology specification languages. In R. Dieng and O. Corby (eds.), Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'00), volume 1937 of LNAI, 80–96. Springer Verlag, 2000.
- 1.14 D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In Knowledge Acquisition, Modeling, and Management, Proceedings of the European Knowledge Acquisition Conference (EKAW-2000), R. Dieng et al. (eds.), Lecture Notes in Artificial Intelligence, LNAI 1937, Springer-Verlag, October 2000.
- 1.15 D. Fensel, I. Horrocks, F. van Harmelen, D. McGuinness, and P. F. Patel-Schneider. OIL: Ontology infrastructure to enable the Semantic Web, IEEE Intelligent System, 16(2), 2001.
- 1.16 F. van Harmelen and I. Horrocks. Questions and answers about OIL. IEEE Intelligent Systems, 15(6):69–72, 2000.
- 1.17 P. D. Karp, V. K. Chaudhri, and J. Thomere. XOL: An XML-based ontology exchange language. Version 0.3, July 1999.
- 1.18 V. K. Chaudhri, A. Farquhar, R. Fikes, P. D. Karp, and J. Rice. OKBC: A programmatic foundation for knowledge base interoperability. In Proceedings of the 15th Nat. Conference on Artificial Intelligence (AAAI'98), 1998.
- 1.19 J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelen, and I. Horrocks. Enabling knowledge representation on the Web by extending RDF schema. In Proceedings of the Tenth International World Wide Web Conference (WWW10), Hong Kong, May 2001.
- 1.20 S. Staab, H.-P. Schnurr, R. Studer and Y. Sure. Knowledge processes and ontologies. IEEE Intelligent Systems, 16(1), January/February 2001.
- 1.21 G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde and B. Wielinga. Knowledge Engineering and Management – The CommonKADS Methodology. The MIT Press, Cambridge, Massachusetts; London, England, 1999.
- 1.22 Th. Lau and Y. Sure. Introducing ontology-based skills management at a large insurance company. In: Proceedings of the Modellierung 2002 “Modellierung in der Praxis - Modellierung fr die Praxis”, Tutzing, Germany, 25.-27. March, 2002.