# On-To-Knowledge: Semantic Web Enabled Knowledge Management

**Dieter Fensel, Frank van Harmelen, Ying Ding, Michel Klein, Hans Akkermans**
*Free University Amsterdam VUA, Division of Mathematics and Informatics*
*De Boelelaan 1081a, NL-1081 HV Amsterdam, The Netherlands*
http://www.ontoknowledge.org ; contact: dieter@cs.vu.nl
**Jeen Broekstra, Arjohn Kampman, Jos van der Meer,** *AIdministrator, The Netherlands*
**York Sure, Rudi Studer,** *AIFB, University of Karlsruhe, Germany*
**Uwe Krohn, John Davies,** *BT, Ipswich, UK*
**Robert Engels,** *CognIT, Oslo, Norway*
**Victor Iosif,** *Enersearch AB, Gothenburg, Sweden*
**Atanas Kiryakov**, *OntoText, Sofia, Bulgaria*
**Thorsten Lau, Ulrich Reimer,** *Swiss Life, Zurich, Switzerland*
**Ian Horrocks**, *University of Manchester, UK*

**Abstract.** *On-To-Knowledge* builds an ontology-based tool environment to speed up 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: (1) a toolset for semantic information processing and user access; (2) OIL, an ontology-based inference layer on top of the World Wide Web; (3) an associated methodology and validation by industrial case studies. This paper offers an overview of the *On-To-Knowledge* approach to knowledge management.

## 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. [Berners-Lee et al., 2001] [Fensel et al., to appear]) 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. [Fensel, 2001]) 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 intelligent 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 contents of this paper are structured as follows. Section 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 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 DAML,[1] we defined the DAML+OIL language that has now been submitted as a web standard at W3C[2], the standardization committee for the World Wide Web. Section 4 discusses the methodological framework for our tool environment and its application in several industrial case studies, including large organizational memories, customer relationship management, and knowledge management in virtual enterprises. Finally, section 5 presents our conclusions and general outlook.

## 2. Tool Environment for Ontology-based Knowledge Management

A key deliverable 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. These tools are based on a three-layered architecture (see Figure 1). Most of the tools presented here in Figure 1 are described below.

[1] www.daml.org
[2] www.w3c.org

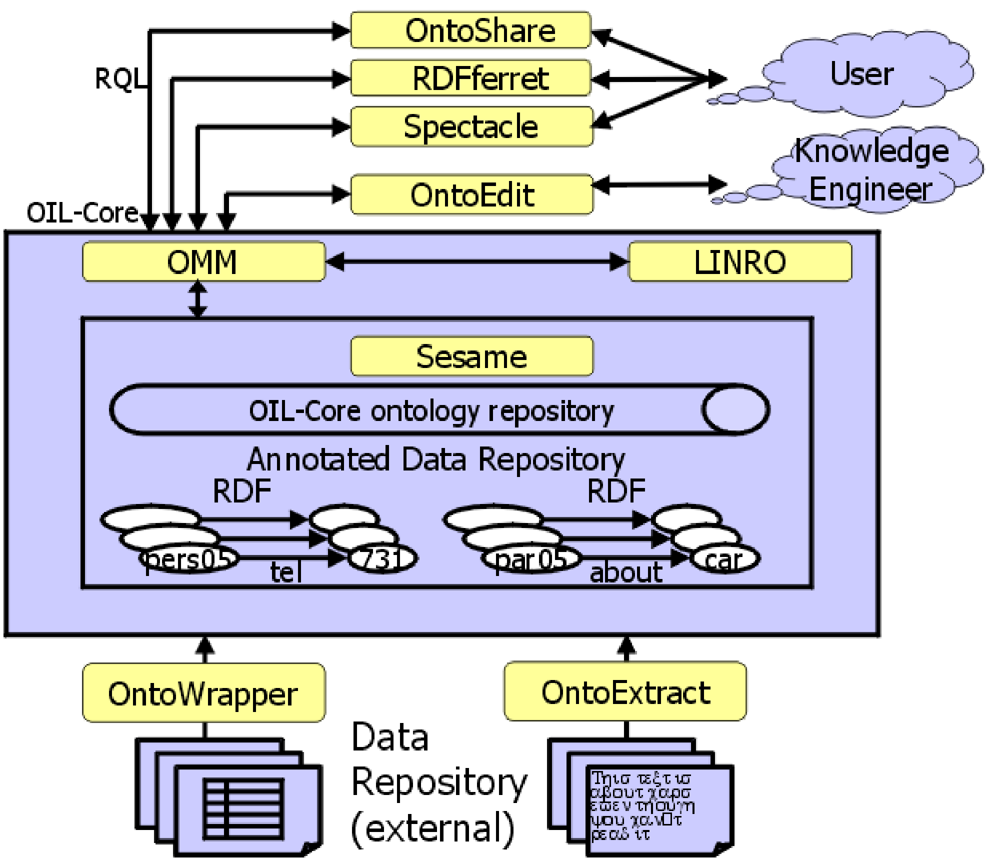
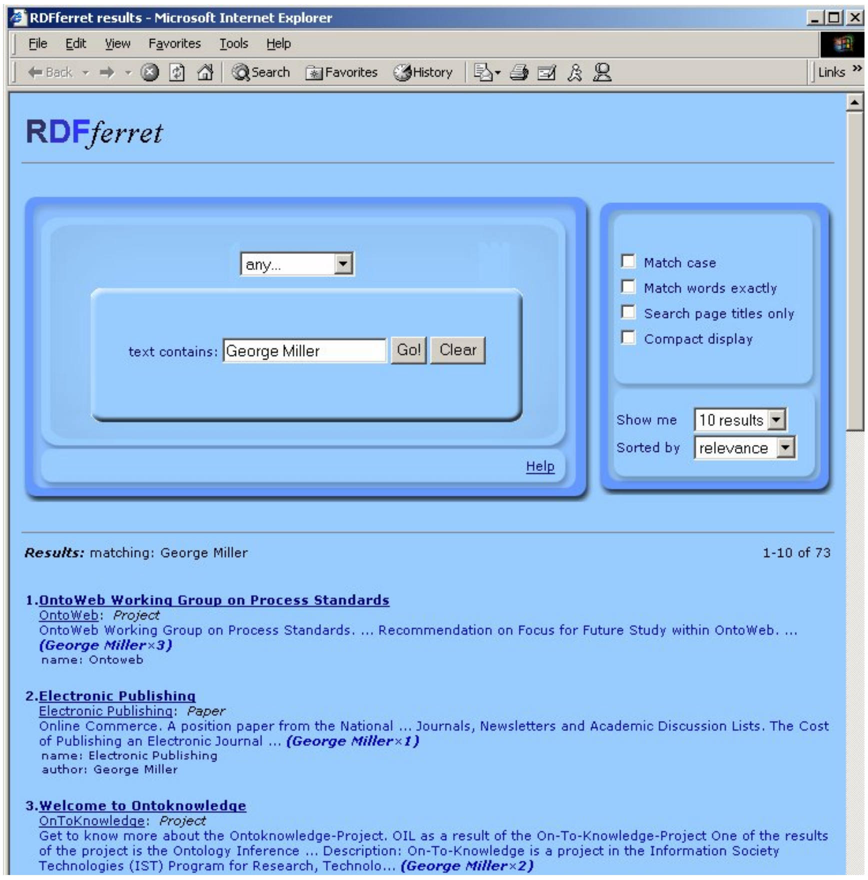


*Figure 1. The technical architecture of On-To-Knowledge.*

**RDF*ferret*** combines full text searching with RDF querying. Observing 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, RDF*ferret* combines full text searching with RDF querying. RDF*ferret* 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, RDF*ferret*'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 RDF*ferret* are RDF resources, which may be Web pages or parts thereof. During indexing RDF*ferret* assigns content descriptors to RDF resources. Content descriptors of a resource are terms (words and phrases) that RDF*ferret* obtains from a full text analysis of the resource content and from processing all literal values that are directly related by a property. They also retain structural information about the ontology. In RDF*ferret* the user can select from a list of all the resource types stored in the index. When searching by selecting a resource type, RDF*ferret* 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. RDF*ferret* 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 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, RDF*ferret* 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.


RDFferret results - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History Links
RDFferret
Help

any...

text contains: George Miller Go! Clear

☐ Match case
☐ Match words exactly
☐ Search page titles only
☐ Compact display

Show me 10 results
Sorted by relevance

**Results:** matching: George Miller 1-10 of 73

1.OntoWeb Working Group on Process Standards
OntoWeb: *Project*
OntoWeb Working Group on Process Standards. ... Recommendation on Focus for Future Study within OntoWeb. ...
***(George Miller×3)***
name: Ontoweb

2.Electronic Publishing
Electronic Publishing: *Paper*
Online Commerce. A position paper from the National ... Journals, Newsletters and Academic Discussion Lists. The Cost of Publishing an Electronic Journal ... ***(George Miller×1)***
name: Electronic Publishing
author: George Miller

3.Welcome to Ontoknowledge
OnToKnowledge: *Project*
Get to know more about the Ontoknowledge-Project. OIL as a result of the On-To-Knowledge-Project One of the results of the project is the Ontology Inference ... Description: On-To-Knowledge is a project in the Information Society Technologies (IST) Program for Research, Technolo... ***(George Miller×2)***

*Figure 2. The query interface* RDF*ferret.*

**OntoShare** enables the storage of best practice information in 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 to orientate new users 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.

**Spectacle** organizes the presentation of information. This presentation is ontology driven. Ontological information, such as classes or specific attributes of information, is used to generate exploration contexts for users. An exploration context makes it easier for users to explore a domain. The context is related to certain tasks, such as finding information or buying products. The context consists of three modules:

- Content: specific content needed to perform a task
- Navigation: suitable navigation disclosing the information
- Design: applicable design displaying the selected content

The modules are independent. For instance, the same content can be explored with different navigation modules. In order to address different target audiences, different Exploration Contexts are generated. In this process, it is important that the modules remain autonomous entities that can be re-used. The content selected comes from sources distributed in - and outside - an organisation. Navigation is based on business and organisational decisions. It provides a route through the content presented. Navigation will differ per user and user group. The design controls the look and feel of the exploration context, including the user interaction. Spectacle consists of the following parts:

- Spectacle server, which handles all interaction between users and exploration contexts;
- Libraries for creating large scale exploration contexts in a spectacle server;
- Graphical user interface for building small-scale exploration contexts.

**OntoEdit** [Staab et al., 2000] makes it possible to inspect, browse, codify and modify ontologies, and thus serves to support the ontology development and maintenance task. Modelling 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 (see Figure 3). The conceptual model of an ontology is stored internally using a powerful ontology model, which can be mapped onto different, concrete representation languages. The functionalities of OntoEdit are easily expandable through a flexible plug-in framework.

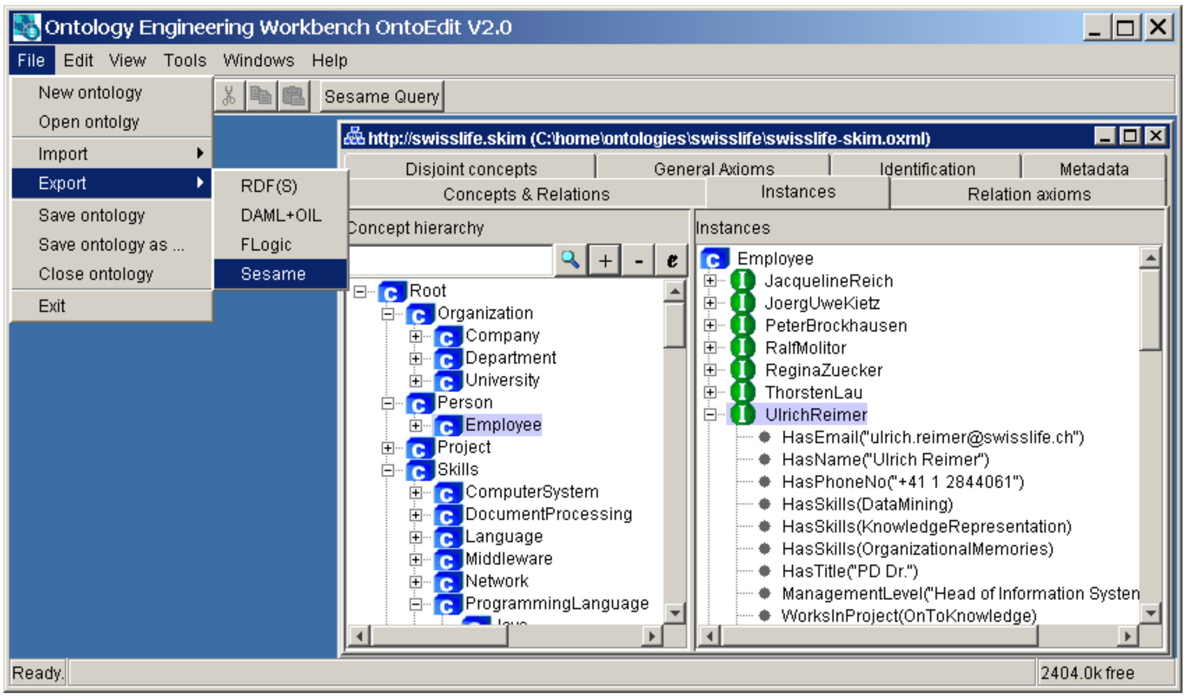


*Figure 3. Screenshot of OntoEdit.*

**Ontology Middleware Module** (OMM) can be seen as the key integration component in the OTK technical solution architecture. It supports well-defined application programming interfaces (OMAPI) used for access to knowledge and deals with such matters as:

- Ontology versioning, including branching.
- Security – user profiles and groups are used to control the rights for access, modifications, and publishing;
- Meta-information and ontology lookup – support for meta-properties (such as Status, Last-Updated-By, Responsible, Comments, etc.) for whole ontologies, as well as for separate concepts and properties. Ontology and concept lookup according to meta-information is possible.
- Access via a number of protocols: HTTP, RMI, EJB, CORBA, and SOAP.

Reasoning services are also implemented so as to extend the functionality provided by SESAME. The goal was to enable an even wider set of applications, such as information extraction and automatic ontology integration. A strategy called pre-reasoning was used to implement workarounds for a number of logical problems proven to be computationally intractable for languages as expressive as OIL. Tasks, such as taxonomy construction and instance classification (realization), are handled off-line.

**LINRO** provides additional reasoning services so as to extend the functionality provided by SESAME. Most of the classical reasoning tasks for description logics (DL) are available, including realization and retrieval. The goal was to enable even wider set of applications, such as information extraction and automatic ontology integration. A strategy called pre-reasoning was used to implement workarounds for a number of logical problems proven to be computationally intractable for languages as expressive as OIL. Tasks such as taxonomy construction and instance classification (realization) are being handled off-line.

**Sesame** [Broekstra et al, to appear] 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. 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 only supports communication over HTTP. However, support for other protocols can be added transparently, simplifying the integration of Sesame into a wide range of environments.

**CORPORUM toolset (OntoExtract and OntoWrapper):** Information extraction and ontology generation is performed by means of the CORPORUM toolset [Engels et al., 2000] and is situated in the *extraction layer*. CORPORUM 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 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.

## 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 2 all exploited ontologies as their common operating ground: raw information source 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 the 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. [Corcho & Gomez Perez, 2000] for a recent overview), it is particularly the 3rd requirement that motivated us to design a new language (baptised 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.

### *Combining Description Logics with Frame Languages*

The OIL language (cf. [Fensel et al., 2000], [Fensel et al., 2001], and [Harmelen & Horrocks, 2000]) is to designed to combine frame-like modeling 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 here[3]). The frame structure of OIL is based on XOL [Karp et al., 1999], an XML serialization of the OKBC-lite knowledge model [Chaudhri et al., 1998]. 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.

[3] http://www.ontoknowledge.org/oil/syntax/

- 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.

### *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 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 [Broekstra *et al.*, 2000].

### *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 4.

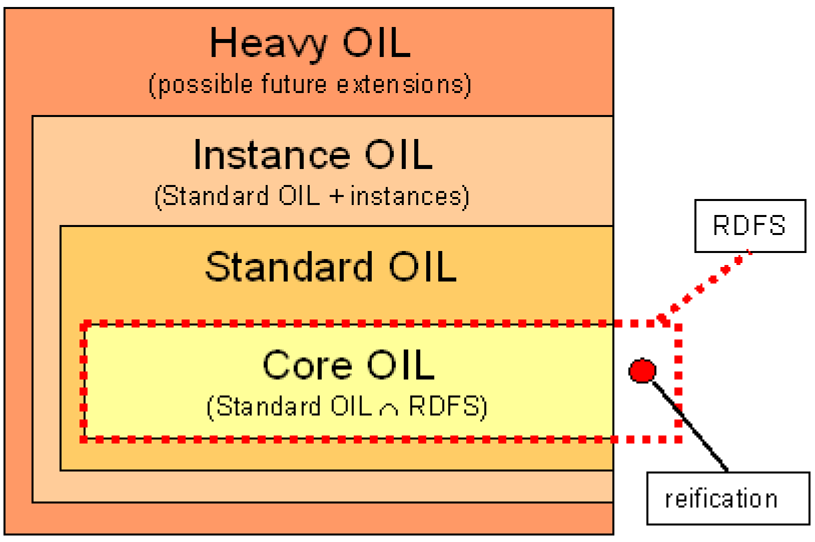


*Figure 4. The layered language model of OIL.*

Figure 5 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
```

Figure 5. OIL illustration.

The following points are noteworthy:

- Skills are restricted to being of  a single level trough 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

### *Current status*

Meanwhile, OIL has been adopted by a joined EU/US initiative that developed a language called DAML+OIL[4], which has now been submitted to the Web Ontology Group of the W3C[5], the standardization committee of the 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.

### *Future developments*

In November 2001, the W3C started a Working Group for defining a Web Ontology language. This WG is chartered to take DAML+OIL as its starting point. 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).

## 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.

### *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 behaviour showed that 70% of the queries involved a reformulation step. On average, 1.5

[4] http://www.daml.org.
[5] http://www.w3c.org.

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.

## *Skills Management*

Swiss Life's second case study is a skills management application 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. The skills management system makes it easy for employees to create in a personal home page on the company's intranet that includes information about personal skills, job functions, and education. The ontology allows 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.

## *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.

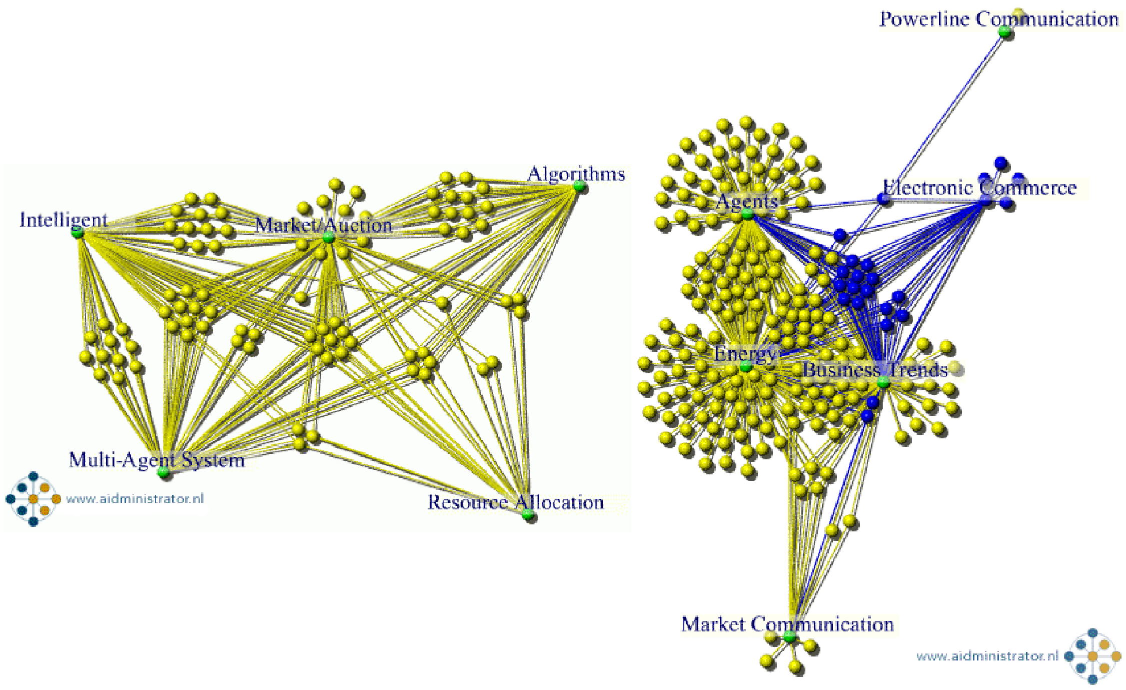


*Figure 6. Automatically generated semantic structure maps of the EnerSearch website.*

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 RDF*ferret* 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 they need (see Figure 6).

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: (1) researchers from different fields, (2) specialists from the shareholders organization and (3) outsiders from different fields.

### *Methodology*

A methodology for employing ontology-based tools for knowledge management applications was developed by applying an initial baseline methodology (cf. [Staab et al., 2001]) in the case studies and continuously improving it with the insight gained from experience. This methodology provides guidelines for introducing knowledge management concepts and tools into enterprises, helping knowledge providers and seekers to present knowledge efficiently and effectively. In On-To-Knowledge, ontologies are the core element. Thus, the methodology has a strong focus on ontology development.

The approach consists of five major phases. It starts with a *feasibility study* 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 would 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.

## 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

[Chaudhri et al., 1998] 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.

[Corcho & Gomez Perez, 2000] 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 *LNA*I, 80–96. Springer Verlag, 2000

[Broekstra *et al.*, 2000] 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.

[Broekstra et al., to appear] J. Broekstra, A. Kampman, F. van Harmelen. *Sesame: An Architecture for Storing and Querying RDF Data and Schema Information*. In [Fensel et al., to appear].

[Berners-Lee et al., 2001] T. Berners-Lee, J. Hendler, and O. Lassila: The Semantic Web, *Scientific American*, May 2001.

[Engels et al., 2000] R. Engels and B.A. Bremdal. *CORPORUM: A Workbench for the Semantic Web*. Semantic Web Mining workshop. PKDD/ECML – 01. Freiburg, Germany, 2001.

[Fensel, 2001] D. Fensel: *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag, Berlin, 2001.

[Fensel et al., 2000] 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.

[Fensel et al., to 2001] 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.

[Fensel et al., to appear] D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster (eds.): *Semantic Web Technology*, MIT Press, Boston, to appear 2002.

[Harmelen & Horrocks, 2000] F. van Harmelen and I. Horrocks: Questions and answers about OIL. *IEEE Intelligent Systems,* 15(6):69–72, 2000.

[Karp et al., 1999] P. D. Karp, V. K. Chaudhri, and J. Thomere. XOL: An XML-based ontology exchange language. Version 0.3, July 1999.

[Staab et al., 2000] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H.-P. Schnurr, R. Studer, and Y. Sure. Semantic community web portals. In WWW9—Proceedings of the 9th International World Wide Web Conference, Amsterdam, The Netherlands, May, 15-19, 2000. Elsevier, 2000.

[Staab et al., 2001] S. Staab, H.-P. Schnurr, R. Studer, and Y. Sure: Knowledge Processes and Ontologies, *IEEE Intelligent Systems*, 16(1), January/February 2001.