OIL: An Ontology Infrastructure for the Semantic Web

Dieter Fensel and Frank van Harmelen, *Vrije Universiteit, Amsterdam*
Ian Horrocks, *University of Manchester, UK*
Deborah L. McGuinness, *Stanford University*
Peter F. Patel-Schneider, *Bell Laboratories*

Ontologies play a major role in supporting information exchange across various networks. A prerequisite for such a role is the development of a joint standard for specifying and exchanging ontologies. The authors present OIL, a proposal for such a standard.

Researchers in artificial intelligence first developed *ontologies* to facilitate knowledge sharing and reuse. Since the beginning of the 1990s, ontologies have become a popular research topic, and several AI research communities—including knowledge engineering, natural language processing, and knowledge representation—have investigated them. More recently, the notion of an ontology is becoming widespread in fields such as intelligent information integration, cooperative information systems, information retrieval, electronic commerce, and knowledge management. Ontologies are becoming popular largely because of what they promise: a shared and common understanding that reaches across people and application systems.

Currently, ontologies applied to the World Wide Web are creating the *Semantic Web*.[1] Originally, the Web grew mainly around HTML, which provides a standard for structuring documents that browsers can translate in a canonical way to render those documents. On the one hand, HTML's simplicity helped spur the Web's fast growth; on the other, its simplicity seriously hampered more advanced Web applications in many domains and for many tasks. This led to XML (see Figure 1), which lets developers define arbitrary domain- and task-specific extensions (even HTML appears as an XML application—XHTML).

XML is basically a defined way to provide a serialized syntax for tree structures—it is an important first step toward building a Semantic Web, where application programs have direct access to data semantics. The *resource description framework*[2] has taken an important additional step by defining a syntactical convention and a simple data model for representing machine-processable data semantics. RDF is a standard for the Web metadata the World Wide Web Consortium (www.w3c.org/rdf) develops, and it defines a data model based on triples: object, property, and value. The *RDF Schema*[3] takes a step further into a richer representation formalism and introduces basic ontological modeling primitives into the Web. With RDFS, we can talk about classes, subclasses, subproperties, domain and range restrictions of properties, and so forth in a Web-based context. We took RDFS as a starting point and enriched it into a full-fledged Web-based ontology language called OIL.[4] We included these aspects:

- A more intuitive choice of some of the modeling primitives and richer ways to define concepts and attributes.
- The definition of a formal semantics for OIL.
- The development of customized editors and inference engines to work with OIL.

Ontologies: A revolution for information access and integration

Many definitions of ontologies have surfaced in the last decade, but the one that in our opinion best characterizes an ontology's essence is this: "An ontology is a formal, explicit specification of a shared conceptualization."[5] In this context, *conceptualization* refers to an abstract model of some phenomenon in the world that identifies that phenomenon's relevant concepts. *Explicit* means that the type of concepts used and the constraints on their use are explicitly defined, and *formal* means that the ontology should be machine understandable. Different degrees of formality are possible. Large ontologies such as WordNet (www.cogsci.princeton.edu/~wn) provide a thesaurus for over 100,000 terms explained in natural language. On the other end of the spectrum is CYC (www.cyc.com), which provides formal axiomating theories for many aspects of commonsense knowledge. *Shared* reflects the notion that an ontology captures consensual knowledge—that is, it is not restricted to some individual but is accepted by a group.

The three main application areas of ontology technology are knowledge management, Web commerce, and electronic business.

Knowledge management

KM is concerned with acquiring, maintaining, and accessing an organization's knowledge. Its purpose is to exploit an organization's intellectual assets for greater productivity, new value, and increased competitiveness. Owing to globalization and the Internet's impact, many organizations are increasingly geographically dispersed and organized around virtual teams. With the large number of online documents, several document management systems have entered the market. However, these systems have weaknesses:

- *Searching information*: Existing keyword-based searches retrieve irrelevant information that uses a certain word in a different context; they might miss information when different words about the desired content are used.
- *Extracting information*: Current human browsing and reading requires extracting relevant information from information sources. Automatic agents lack the commonsense knowledge required to extract such information from textual representations, and they fail to integrate information spread over different sources.
- *Maintaining*: Sustaining weakly structured text sources is difficult and time-consuming when such sources become large. Keeping such collections consistent, correct, and up to date requires a mechanized representation of semantics and constraints that can help detect anomalies.
- *Automatic document generation*: Adaptive Web sites that enable dynamic reconfiguration according to user profiles or other relevant aspects could prove very useful. The generation of semistructured information presentations from semistructured data requires a machine-accessible representation of the semantics of these information sources.

Using ontologies, semantic annotations will allow structural and semantic definitions of documents. These annotations could provide completely new possibilities: intelligent search instead of keyword matching, query answering instead of information retrieval, document exchange between departments through ontology mappings, and definitions of views on documents.

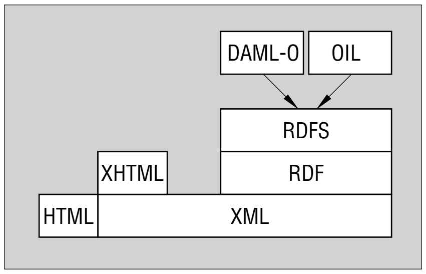


Figure 1. The layer language model for the Web.

Web commerce

E-commerce is an important and growing business area for two reasons. First, e-commerce extends existing business models—it reduces costs, extends existing distribution channels, and might even introduce new distribution possibilities. Second, it enables completely new business models and gives them a much greater importance than they had before. What has up to now been a peripheral aspect of a business field can suddenly receive its own important revenue flow.

Examples of business field extensions are online stores; examples of new business fields are shopping agents and online marketplaces and auction houses that turn comparison shopping into a business with its own significant revenue flow. The advantages of online stores and their success stories have led to a large number of shopping pages. The new task for customers is to find a shop that sells the product they're seeking, getting it in the desired quality, quantity, and time, and paying as little as possible for it. Achieving these goals through browsing requires significant time and only covers a small share of the actual offers. Shopbots visit several stores, extract product information, and present it to the customer as an instant market overview. Their functionality is provided through *wrappers*, which use keyword search to find product information together with assumptions on regularities in presentation format and text extraction heuristics. This technology has two severe limitations:

- *Effort*: Writing a wrapper for each online store is time-consuming, and changes in store presentation or organization increase maintenance.
- *Quality*: The extracted product information is limited (it contains mostly price information), error-prone, and incomplete. For example, a wrapper might extract the product price, but it usually misses indirect costs such as shipping.

Most product information is provided in natural language; automatic text recognition is still a research area with significant unsolved problems. However, the situation will drastically change in the near future when standard representation formalisms for data structure and semantics are available. Software agents will then *understand* product information. Meta online stores will grow with little effort, which will enable complete market transparency in the various dimensions of the diverse product properties. Ontology mappings, which translate different product descriptions, will replace the low-level programming of wrappers, which is based on text extraction and format heuristics. An ontology will describe the various products and help navigate and search automatically for the required information.

Electronic business

E-commerce in the business-to-business field (B2B) is not new—initiatives to support it in business processes between different companies existed in the 1960s. To exchange business transactions electronically, sender and receiver must agree on a

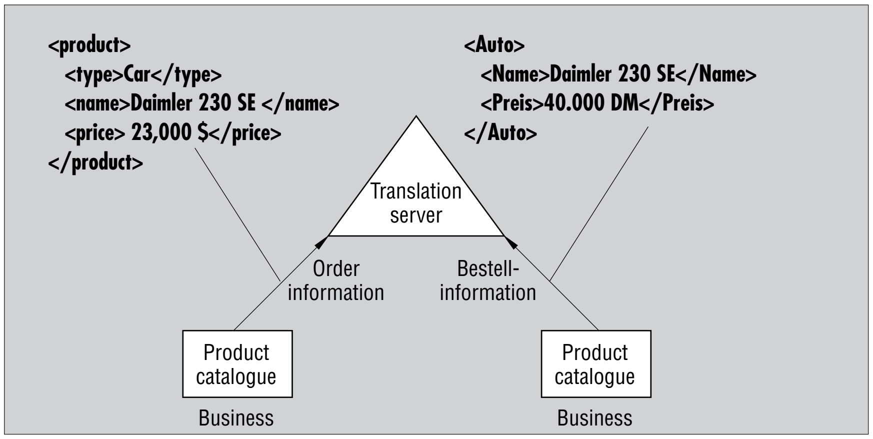


Figure 2. The translation of structure, semantics, and language.

standard (a protocol for transmitting content and a language for describing content). A number of standards arose for this purpose—one of them is the UN initiative, Electronic Data Interchange for Administration, Commerce, and Transport (Edifact). In general, the automation of business transactions has not lived up to the propagandists' expectations, partly because of the serious shortcomings of approaches such as Edifact: It is a procedural and cumbersome standard, making the programming of business transactions expensive and error-prone, and it results in large maintenance efforts. Moreover, the exchange of business data over extranets is not integrated with other document exchange processes—Edifact is an isolated standard.

Using the Internet's infrastructure for business exchange will significantly improve this situation. Standard browsers can render business transactions and transparently integrate them into other document exchange processes in intranet and Internet environments. However, the fact that HTML does not provide a means for presenting rich syntax and data semantics hampers this. XML, which is designed to close this gap in current Internet technology, drastically changes the situation. We can model B2B communication and data exchange with the same means available for other data exchange processes, we can render transaction specifications on standard browsers, and maintenance is cheap. However, although XML provides a standard serialized syntax for defining data structure and semantics, it does not provide standard data structures and terminologies to describe business processes and exchanged products. Therefore, XML-based e-commerce will need ontologies in two important ways:

- *Standard ontologies* must cover the various business areas. In addition to official standards, vertical marketplaces (Internet portals) could generate de facto standards—if they can attract significant shares of a business field's online transactions. Examples include Dublin Core, Common Business Library (CBL), Commerce XML (cXML), ecl@ss, Open Applications Group Integration Specification (OAGIS), Open Catalog Format (OCF), Open Financial Exchange (OFX), Real Estate Transaction Markup Language (RETML), RosettaNet, UN/SPSC (see www.diffuse.org), and UCEC.
- *Ontology-based translation services* must link different data structures in areas where standard ontologies do not exist or where a particular client needs a translation from his or her terminology into the standard. This translation service must cover structural and semantic as well as language differences (see Figure 2).

Ontology-based trading will significantly extend the degree to which data exchange is automated and will create completely new business models in participating market segments.

Why OIL?

Effective, efficient work with ontologies requires support from advanced tools. We need an advanced ontology language to express and represent ontologies. This language must meet three requirements:

- It must be highly intuitive to the human user. Given the success of the frame-based and object-oriented modeling paradigm, an ontology should have a frame-like look and feel.
- It must have a well-defined formal semantics with established reasoning properties to ensure completeness, correctness, and efficiency.
- It must have a proper link with existing Web languages such as XML and RDF to ensure interoperability.

Many of the existing languages such as CycL,[6] KIF,[7] and Ontolingua[8] fail. However, OIL[9] matches these criteria and unifies the three important aspects that different communities provide: epistemologically rich modeling primitives as provided by the frame community, formal semantics and efficient reasoning support as provided by description logics, and a standard proposal for syntactical exchange notations as provided by the Web community.

Frame-based systems

The central modeling primitives of predicate logic are predicates. Frame-based and object-oriented approaches take a different viewpoint. Their central modeling primitives are classes (or frames) with certain properties called *attributes*. These attributes do not have a global scope but apply only to the classes for which they are defined—we can associate the "same" attribute (the same attribute name) with different range and value restrictions when defined for different classes. A frame provides a context for modeling one aspect of a domain. Researchers have developed many other additional refinements of these modeling constructs, which have led to this modeling paradigm's incredible success.

Many frame-based systems and languages have emerged, and, renamed as object orientation, they have conquered the software engineering community. OIL incorporates the *essential modeling primitives* of frame-based systems—it is based on the notion of a *concept* and the definition of its superclasses and attributes. Relations can also be defined not as an attribute of a class but as an independent entity having a certain domain and range. Like classes, relations can fall into a hierarchy.

Description logics

DL describes knowledge in terms of concepts and role restrictions that can automatically derive classification taxonomies. Knowledge representation research's main thrust is to provide theories and systems for expressing structured knowledge and for accessing and reasoning with it in a principled way. In spite of the discouraging theoretical complexity of the results, there are now efficient implementations for DL languages, which we explain later. OIL inherits from DL its formal semantics and the efficient reasoning support.

Web standards: XML and RDF

Modeling primitives and their semantics are one aspect of an ontology language, but we still have to decide about its syntax. Given the Web's current dominance and importance, we must formulate a syntax of an ontology exchange language with existing Web standards for information representation. First, OIL has a well-defined syntax in XML based on a document type definition and an XML Schema definition. Second, OIL is an extension of RDF and RDFS. With regard to ontologies, RDFS provides two important contributions: a standardized syntax for writing ontologies and a standard set of modeling primitives such as **instance-of** and **subclass-of** relationships.

OIL's layered architecture

A single ontology language is unlikely to fulfill all the needs of the Semantic Web's large range of users and applications. We therefore organized OIL as a series of ever-increasing layers of sublanguages. Each additional layer adds functionality and complexity to the previous one. Agents (humans or machines) that can only process a lower layer can still partially understand ontologies expressed in any of the higher layers. A first and very important application of this principle is the relation between OIL and RDFS. As Figure 3 shows, core OIL coincides largely with RDFS (with the exception of RDFS's reification features). This means that even simple RDFS agents can process OIL ontologies and pick up as much of their meaning as possible with their limited capabilities.

Standard OIL aims to capture the necessary mainstream modeling primitives that provide adequate expressive power and are well understood, thus precisely specifying the semantics and making complete inference viable.

Instance OIL includes a thorough individual integration. Although the previous layer—Standard OIL—includes modeling constructs that specify individual fillers in term definitions, Instance OIL includes a full-fledged database capability.

Heavy OIL will include additional representational (and reasoning) capabilities. A more expressive rule language and metaclass facilities seem highly desirable. We will define these extensions of OIL in cooperation with the DAML (DARPA Agent Markup Language; www.daml.org) initiative for a rule language for the Web.

OIL's layered architecture has three advantages:

- An application is not forced to work with a language that offers significantly more expressiveness and complexity than is needed.
- Applications that can only process a lower level of complexity can still catch some of an ontology's aspects.
- An application that is aware of a higher level of complexity can still understand ontologies expressed in a simpler ontology language.

Defining an ontology language as an extension of RDFS means that every RDFS ontology is a valid ontology in the new language (an OIL processor will also understand RDFS). However, the other direction is also possible: Defining an OIL extension as closely as possible to RDFS allows maximal reuse of existing RDFS-based applications and tools. However, because the ontology language usually contains new aspects (and therefore a new vocabulary, which an RDFS processor does not know), 100 percent compatibility is impossible. Let's look at an example. The following OIL expression defines **herbivore** as a class, which is a subclass of **animal** and disjunct to all **carnivores**:

```
<rdfs:Class rdf:ID="herbivore">
  <rdf:type
      rdf:resource="http://www.
         ontoknowledge.org/oil/RDFS-
           schema/#DefinedClass"/>
  <rdfs:subClassOf rdf:resource="#animal"/>
  <rdfs:subClassOf>
      <oil:NOT>
          <oil:hasOperand rdf:resource="
            #carnivore"/>
      </oil:NOT>
  </rdfs:subClassOf>
</rdfs:Class>
```

An application limited to pure RDFS can still capture some aspects of this definition:

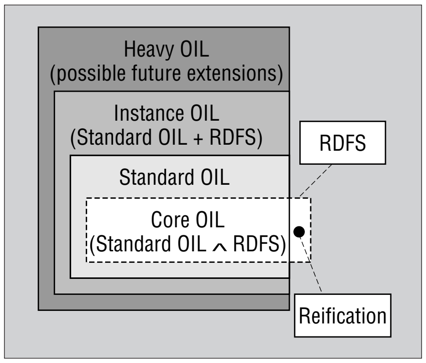


Figure 3. OIL's layered language model.

```
<rdfs:Class rdf:ID="herbivore">
  <rdfs:subClassOf rdf:resource="#animal"/>
  <rdfs:subClassOf>
                 ...
  </rdfs:subClassOf>
</rdfs:Class>
```

It encounters that **herbivore** is a subclass of **animal** and a subclass of a second class, which it cannot understand properly. This seems to preserve complicated semantics for simpler applications.

An illustration of the OIL modeling primitive

An OIL ontology is itself annotated with metadata, starting with such things as title, creator, creation date, and so on. OIL follows the W3C Dublin Core Standard on bibliographical meta date for this purpose.

Any ontology language's core is its hierarchy of class declarations, stating, for example, that DeskJet printers are a subclass of printers. We can declare classes as *defined*, which indicates that the stated properties are not only necessary but also sufficient conditions for class membership. Instead of using single types in expressions, we can combine classes in logical expressions indicating intersection, union, and complement of classes.

We can declare *slots* (relations between classes) together with logical axioms, stating whether they are functional (having at most one value), transitive, or symmetric, and stating which (if any) slots are inverse. We can state range restrictions as part of a slot declaration as well as the number of distinct values that a slot may have. We can further restrict slots by *value-type* or *has-value* restrictions. A value-type restriction demands that every value of the property must be of the stated

```
class-def Product
slot-def Price
  domain Product
slot-def ManufacturedBy
  domain Product
class-def PrintingAndDigitalImagingProduct
  subclass-of Product
class-def HPProduct
  subclass-of Product
  slot-constraint ManufacturedBy
    has-value "Hewlett Packard"
class-def Printer
  subclass-of PrintingAndDigitalImagingProduct
slot-def PrinterTechnology
  domain Printer
slot-def Printing Speed
  domain Printer
slot-def PrintingResolution
  domain Printer
class-def PrinterForPersonalUse
  subclass-of Printer
class-def HPPrinter
  subclass-of HPProduct and Printer
class-def LaserJetPrinter
  subclass-of Printer
  slot-constraint PrintingTechnology
    has-value "Laser Jet"
class-def HPLaserJetPrinter
  subclass-of LaserJetPrinter and HPProduct
class-def HPLaserJet1100Series
  subclass-of HPLaserJetPrinter and PrinterFor
    PersonalUse
  slot-constraint PrintingSpeed
    has-value "8 ppm"
  slot-constraint PrintingResolution
    has-value "600 dpi"
class-def HPLaserJet1100se
  subclass-of HPLaserJet1100Series
  slot-constraint Price
    has-value "$479"
class-def HPLaserJet1100xi
  subclass-of HPLaserJet1100Series
  slot-constraint Price
    has-value "$399"
```

Figure 4. A small printer ontology in OIL.

type; has-value restrictions require the slot to have at least values from the stated type.

A crucial aspect of OIL is its formal semantics.[10] An OIL ontology is given a formal semantics by mapping each class into a set of objects and each slot into a set of pairs of objects. This mapping must obey the constraints specified by the definitions of the classes and slots. We omit the details of this formal semantics, but it must exist and be consulted whenever necessary to resolve disputes about the meaning of language constructions. It is an ultimate reference point for OIL applications.

Figure 4 shows a very simple example of an OIL ontology provided by SemanticEdge (www.interprice.com). It illustrates OIL's most basic constructs.

This defines a number of classes and organizes them in a class hierarchy (for example, **HPProduct** is a subclass of **Product**). Various properties (or slots) are defined, together with the classes to which they apply (such as a **Price** is a property of any **Product**, but a **PrintingResolution** can only be stated for a **Printer**, an indirect subclass of **Product**). For certain classes, these properties have restricted values (for example, the price of any HPLaserJet1100se is restricted to $479). In OIL, we can also combine classes by using logical expressions—for example, an **HPPrinter** is both an **HPProduct** and a **Printer** (and consequently inherits the properties from both classes).

OIL tools

OIL has strong tool support in three areas:

- *ontology editors*, to build new ontologies;
- *ontology-based annotation tools*, to link unstructured and semistructured information sources with ontologies; and
- *reasoning with ontologies*, which enables advanced query-answering services, supports ontology creation, and helps map between different ontologies.

Ontology editors

Ontology editors help human knowledge engineers build ontologies—they support the definition of concept hierarchies, the definition attributes for concepts, and the definition of axioms and constraints. They must provide graphical interfaces and conform to existing standards in Web-based software development. They enable the inspecting, browsing, codifying, and modifying of ontologies, and they support ontology development and maintenance tasks. Currently, two editors for OIL are available, and a third is under development:

- *OntoEdit* (see Figure 5) is an ontology-engineering environment developed at the Knowledge Management Group of the University of Karlsruhe, Institute AIFB (http://ontoserver.aifb.uni-karlsruhe.de/ontoedit). Currently, OntoEdit supports Frame-Logic, OIL, RDFS, and XML. It is commercialized from Ontoprise (www.ontoprise.de).
- *OILed* is a freely available and customized editor for OIL implemented by the University of Manchester and sponsored by the Vrije Universiteit, Amsterdam, and SemanticEdge (see http://img.cs.man.ac.uk/oil). OILed aims to provide a simple freeware editor that demonstrates—and stimulates interest in—OIL. OILed is not intended to be a full ontology development environment—it will not actively support the development of large-scale ontologies, the migration and integration of ontologies, versioning, argumentation, and many other activities that are involved in ontology construction. Rather, it is a NotePad for ontology editors that offers just enough functionality to let users build ontologies and demonstrate how to check them for consistency.
- *Protégé*[11] lets domain experts build knowledge-based systems by creating and modifying reusable ontologies and problem-solving methods (see www.smi.stanford.edu/projects/protege). Protégé generates domain-specific knowledge acquisition tools and applications from ontologies. More than 30 countries have used it. It is an ontology editor that can define classes and class hierarchy, slots and slot-value restrictions, and relationships between classes and properties of these relationships. The instances tab is a knowledge acquisition tool that can acquire instances of the classes defined in the ontology. Protégé, built at Stanford University, currently supports RDF—work on extending it to OIL is starting.

Ontology-based annotation tools

Ontologies can describe large instance populations. In OIL's case, two tools currently aid such a process. First, we can derive an XML DTD and an XML Schema definition from an ontology in OIL. Second, we can derive an RDF and RDFS definition for instances from OIL. Both provide means to express large volumes of semistructured information as instance information in OIL. More details appear elsewhere.[4,12,13]

Reasoning with ontologies: Instance and schema inferences

Inference engines for ontologies can reason about an ontology's instances and schema definition. For example, they can automatically derive the right position of a new concept in a given concept hierarchy. Such reasoners help build ontologies and use them for advanced information access and navigation. OIL uses the FaCT (Fast Classification of Terminologies, www.cs.man.ac.uk/~horrocks/FaCT) system to provide reasoning support for ontology design, integration, and verification. FaCT is a DL classifier that can provide consistency checking in modal and other similar logics. FaCT's most interesting features are its expressive logic, its optimized tableaux implementation (which has now become the standard for DL systems), and its Corba-based client–server architecture. FaCT's optimizations specifically aim to improve the system's performance when classifying realistic ontologies. This results in performance improvements of several orders of magnitude compared with older DL systems. This performance improvement is often so great that it is impossible to measure precisely because nonoptimized systems are virtually nonterminating with ontologies that FaCT can easily deal with.[14] For example, for a large medical terminology ontology developed in the GALEN project,[15] FaCT can check the consistency of all 2,740 classes and determine the complete class hierarchy in approximately 60 seconds of CPU (450-MHz Pentium III) time. FaCT can be accessed through a Corba interface.

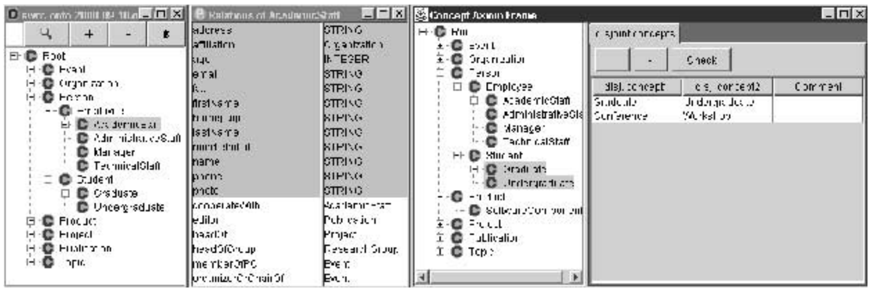


Figure 5. A screen shot of OntoEdit.

Applications of OIL

Earlier, we sketched three application areas for ontologies: knowledge management, Web commerce, and e-business. Not surprisingly, we find applications of OIL in all three areas. On-To-Knowledge (www.ontoknowledge.org)[16] extends OIL to a full-fledged environment for knowledge management in large intranets and Web sites. Unstructured and semistructured data is automatically annotated, and agent-based user interface techniques and visualization tools help users navigate and query the information space. Here, On-To-Knowledge continues a line of research that began with SHOE[17] and Ontobroker:[18] using ontologies to model and annotate the semantics of information resources in a machine-processable manner. On-To-Knowledge is carrying out three industrial case studies to evaluate the tool environment for ontology-based knowledge management.

Swiss Life: Organizational memory

Swiss Life[19] (www.swisslife.ch) implements an intranet-based front end to an organizational memory with OIL. The starting point is the existing intranet information system, called ZIS, which has considerable drawbacks. Its great flexibility allows for its evolution with actual needs, but this also makes finding certain information difficult. Search engines help only marginally. Clearly, formalized knowledge is connected with weakly structured background knowledge here—experience shows that this is extremely bothersome and error-prone to maintain. The only way out is to apply content-based information access so that we no longer have a mere collection of Web pages but a full-fledged information system that we can rightly call an organizational memory.

British Telecom: Call centers

Call centers are an increasingly important mechanism for customer contact in many industries. What will be required in the future is a new philosophy in customer interaction design. Every transaction should emphasize the uniqueness of both the customer and the customer service person—this requires effective knowledge management (see www.bt.com/innovations), including knowledge about the customer and about the customer service person, so that customers are directed to the correct person in a meaningful and timely way. Some of BT's call centers are targeted to identify opportunities for effective knowledge management. More specifically, call center agents tend to use a variety of electronic sources for information when interacting with customers, including their own specialized systems, customer databases, the organization's intranet, and, perhaps most important, case bases of best practices. OIL provides an intuitive front-end tool to these heterogeneous information sources to ensure smooth transfer to others.

EnerSearch: Virtual enterprise

EnerSearch is a virtual organization researching new IT-based business strategies and customer services in deregulated energy markets (www.enersearch.se).[20] EnerSearch is a knowledge creation company—knowledge that must transfer to its shareholders and other interested parties. Its Web site is one of the mechanisms for this, but finding information on certain topics is difficult—the current search engine supports free-text search rather than content-based search. So, EnerSearch applies the OIL toolkit to enhance knowledge transfer to researchers in the virtual organization in different disciplines and countries and specialists from shareholding companies interested in getting up-to-date R&D information.

OIL has several advantages: it is properly grounded in Web languages such as XML Schemas and RDFS, and it offers different levels of complexity. Its inner layers enable efficient reasoning support based on FaCT, and it has a well-defined formal semantics that is a baseline requirement for the Semantic Web's languages. Regarding its modeling primitives, OIL is not just another new language but reflects certain consensus in areas such as DL and frame-based systems. We could only achieve this by including a large group of scientists in OIL's development. OIL is also a significant source of inspiration for the ontology language

DAML+OIL (www.cs.man.ac.uk/~horrocks/DAML-OIL), developed through the DAML initiative. The next step is to start on a W3C working group on the Semantic Web, taking DAML+OIL as a starting point.

Defining a proper language is an important step to expanding the Semantic Web. Developing new tools, architectures, and applications is the real challenge that will follow.

Acknowledgments

We thank Hans Akkermans, Sean Bechhofer, Jeen Broekstra, Stefan Decker, Ying Ding, Michael Erdmann, Carole Goble, Michel Klein, Alexander Mädche, Enrico Motta, Borys Omelayenko, Steffen Staab, Guus Schreiber, Lynn Stein, Heiner Stuckenschmidt, and Rudi Studer, all of whom were involved in OIL's development.

References

1. T. Berners-Lee, *Weaving the Web*, Orion Business Books, London, 1999.
2. O. Lassila and R. Swick, *Resource Description Framework (RDF) Model and Syntax Specification*, *W3C Recommendation*, World Wide Web Consortium, Boston, 1999, www.w3.org/TR/REC-rdf-syntax (current 6 Dec. 2000).
3. D. Brickley and R.V. Guha, *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation*, World Wide Web Consortium, Boston, 2000, www.w3.org/TR/rdf-schema (current 6 Dec. 2000).
4. J. Broekstra et al., "Enabling Knowledge Representation on the Web by Extending RDF Schema," *Proc. 10th Int'l World Wide Web Conf.*, Hong Kong, 2001.
5. T.R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, 1993, pp. 199–220.
6. D.B. Lenat and R.V. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*, Addison-Wesley, Reading, Mass., 1990.
7. M.R. Genesereth, "Knowledge Interchange Format," *Proc. Second Int'l Conf. Principles of Knowledge Representation and Reasoning* (KR 91), J. Allenet et al., eds., Morgan Kaufmann, San Francisco, 1991, pp. 238–249; http://logic.stanford.edu/kif/kif.html (current 9 Mar. 2001).
8. A. Farquhar, R. Fikes, and J. Rice, "The Ontolingua Server: A Tool for Collaborative Ontology Construction," *Int'l. J. Human–Computer Studies*, vol. 46, 1997, pp. 707–728.

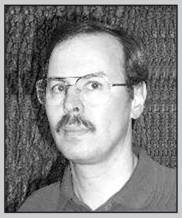
The Authors

Dieter Fensel's biography appears in the Guest Editors' Introduction on page 25.

Ian Horrocks is a lecturer in computer science with the Information Management Group at the University of Manchester, UK. His research interests include knowledge representation, automated reasoning, optimizing reasoning systems, and ontological engineering, with particular emphasis on the application of these techniques to the World Wide Web. He received a PhD in computer science from the University of Manchester. He is a member of the OIL language steering committee and the Joint EU/US Committee on Agent Markup Languages, and is coeditor of the DAML+OIL language specification. Contact him at the Dept. of Computer Science, Univ. of Manchester, Oxford Rd., Manchester, M13 9PL, UK; horrocks@cs.man.ac; www.cs.man.ac.uk/~horrocks.

Frank van Harmelen is a senior lecturer in the AI Department at Vrije Universiteit in Amsterdam. His research interests include specification languages for knowledge-based systems, using languages for validation and verification of KBS, developing gradual notions of correctness for KBS, and verifying weakly structured data. He received a PhD in artificial intelligence from the University of Edinburgh. Contact him at Dept. of AI, Faculty of Sciences, Vrije Universiteit Amsterdam de Boelelaan 1081a, 1081HV Amsterdam, Netherlands; frank.van.harmelen@cs.vu.nl; www.cs.vu.nl/~frankh.

Deborah L. McGuinness is the associate director and senior research scientist for the Knowledge Systems Laboratory at Stanford University. Her main research areas include ontologies, description logics, reasoning systems, environments for building and maintaining information, knowledge-enhanced search, configuration, and intelligent commerce applications. She also runs a consulting business dealing with ontologies and artificial intelligence for business applications and serves on several technology advisory boards and academic advisory boards. She received a PhD from Rutgers University in reasoning systems. Contact her at the Knowledge Systems Laboratory, Stanford Univ., Stanford, CA 94305; dlm@ksl.stanford.edu; www.ksl.stanford.edu/people/dm.



Peter F. Patel-Schneider is a member of the technical staff at Bell Labs Research. His research interests center on the properties and use of description logics. He is also interested in rule-based systems, including standard systems derived from OPS as well as newer formalisms such as R++. He received his PhD from the University of Toronto. Contact him at Bell Labs Research, 600 Mountain Ave., Murray Hill, NJ 07974; pfps@research.bell-labs.com; www.bell-labs.com/user/pfps.

9. D. Fensel et al., "OIL in a Nutshell," *Proc. European Knowledge Acquisition Conference* (EKAW 2000), R. Dieng et al., eds., *Lecture Notes in Artificial Intelligence*, no. 1937, Springer-Verlag, Berlin, 2000, pp. 1–16.

10. I. Horrocks et al., *The Ontology Inference Layer OIL*, tech. report, Vrije Universiteit Amsterdam; www.ontoknowledge.org/oil/TR/oil.long.html (current 9 Mar. 2001).

11. W.E. Grosso et al., "Knowledge Modeling at the Millennium (The Design and Evolution of Protege-2000)," *Proc. 12th Workshop Knowledge Acquisition, Modeling, and Management*, 1999.

12. M. Klein et al., "The Relation between Ontologies and Schema-Languages: Translating OIL Specifications to XML Schema," *Proc. Workshop on Applications of Ontologies and Problem-Solving Methods, 14th European Conf. on Artificial Intelligence*, Berlin, 2000.

13. M. Erdmann and R. Studer, "How to Structure and Access XML Documents with Ontologies," *Data and Knowledge Eng.*, vol. 36, no. 3, 2001.

14. I. Horrocks and P.F. Patel-Schneider, "Optimizing Description Logic Subsumption," *J. Logic and Computation*, vol. 9, no. 3, June 1999, pp. 267–293.

15. A.L. Rector, W.A. Nowlan, and A. Glowinski, "Goals for Concept Representation in the GALEN Project," *Proc. 17th Ann. Symp. Computer Applications in Medical Care* (SCAMC 93), McGraw-Hill, New York, 1993, pp. 414–418.

16. D. Fensel et al., "On-To-Knowledge: Ontology-based Tools for Knowledge Management," *Proc. eBusiness and eWork 2000 Conf.* (EMMSEC 2000), 2000.

17. S. Luke, L. Spector, and D. Rager, "Ontology-Based Knowledge Discovery on the World Wide Web," *Proc. 13th Nat'l Conf. Artificial Intelligence* (AAAI 96), American Association for Artificial Intelligence, Menlo Park, Calif., 1996.

18. D. Fensel et al., "Lessons Learned from Applying AI to the Web," *J. Cooperative Information Systems*, vol. 9, no. 4, Dec. 2000, pp. 361–382.

19. U. Reimer et al., eds., *Proc. Second Int'l Conf. Practical Aspects of Knowledge Management* (PAKM 98), 1998.

20. F. Ygge and J.M. Akkermans, "Decentralized Markets versus Central Control: A Comparative Study," *J. Artificial Intelligence Research*, vol. 11, July–Dec. 1999, pp. 301–333.