Semantic Web Methods for Knowledge Management

Zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaften
(Dr. rer. pol.)

an der Fakultät für
Wirtschaftswissenschaften
der Universität Fridericiana zu Karlsruhe (TH)
genehmigte

DISSERTATION

von
Diplom-Informatiker Stefan Decker

Tag der mündlichen Prüfung: 22. Februar 2002
Referent: Prof. Dr. Rudi Studer
1. Korreferent: Prof. Dr. Peter Knauth
2. Korreferent: Prof. Dr. Gio Wiederhold

2002 Karlsruhe

What information consumes is rather obvious:
It consumes the attention of its recipients.
Hence a wealth of information creates a
poverty of attention, and a need to
allocate that attention efficiently
among the overabundance
of information sour-
ces that might
consume
it.

Nobel Laureate Economist
Herbert A. Simon.

Acknowledgement

The main results of this thesis were obtained during my time at the Institute AIFB at the University of Karlsruhe, Germany. I would like to express my gratitude to my advisor, Prof. Dr. Rudi Studer, for his support, patience, and encouragement throughout my graduate studies. It is not often that one finds an advisor and colleague who always finds the time for listening to the little problems and roadblocks that unavoidably crop up in the course of performing research.
I am also grateful to Prof. Dr. Peter Knauth and Prof. Gio Wiederhold who were willing to serve on my dissertation committee on a very short notice, and to Prof. Dr. Wolfried Stucky and Prof. Dr. Andreas Geyer-Schulz, who served on the examination committee.
My thanks also go to all the colleagues at the Institute AIFB - in particular to Dr. Michael Erdmann - together we worked on the nitty-gritty details of the Ontobroker architecture, and Michael helped me to shape my ideas about the future of the Web. The vision, energy and dedication of Dr. Dieter Fensel has been invaluable for this thesis and the Ontobroker project - without him that project had not happened. Michael Schenk, Ulf Steinberg, and Thorsten Wolf, who wrote their diploma theses under my supervision, contributed heavily to Ontobroker. Dr. Wolfgang Weitz was always there for a discussion when I got stuck. I'm grateful to Dr. Jürgen Angele for the collaboration during my time in Karlsruhe.
I'm especially indebted to Prof. Dr. Gio Wiederhold, who gave me the opportunity to work at Stanford University and taught me a lot about research and life in general.
I thank Jade Reidy and Michael Sintek for proofreading drafts of this thesis - any remaining errors are of course mine.
My parents, Maria and Bernhard, receive my deepest gratitude and love for their dedication and the many years of support during my undergraduate studies that provided the foundation for this work.
Last, but not least, I would like to thank my wife Birgit and my children Jana and Robin for their understanding and love during the past years. Their support and encouragement was in the end what made this dissertation possible.

Table of Contents

Chapter 1 Introduction

1.1 Motivation

Two different developments provide the foundation for this thesis. The first is the growing importance of *knowledge* for our economies, which has fueled interest in the practical *management of knowledge*. The second is the formation of the World Wide Web, which provides a publicly accessible, shared global information space - a phenomenon which has never before existed in the history of mankind. Both topics are briefly introduced here.

1.1.1 Knowledge Management

Classical economics focused on production factors: labor, capital, and land. More recently, another factor turned out to be critical for our economies. The World Bank has recognized the importance of knowledge for development and describes its significance as follows:

> *"Knowledge is critical for development, because everything we do depends on knowledge. Simply to live, we must transform the resources we have into the things we need, and that takes knowledge. And if we want to live better tomorrow than today, if we want to raise our living standards as a household or as a country and improve our health, better educate our children, and preserve our common environment we must do more than simply transform more resources, for resources are scarce. We must use those resources in ways that generate ever higher returns for our efforts and investments. That, too, takes knowledge, and in ever greater proportion to our resources. For countries in the vanguard of the world economy, the balance between knowledge and resources has shifted so far toward the former that knowledge has become perhaps the most important factor determining the standard of living more than land, than tools, than labor. Today's most technologically advanced economies are truly knowledge-based. And as they generate new wealth from their innovations, they are creating millions of knowledge-related jobs in an array of disciplines that have emerged overnight: knowledge engineers, knowledge managers, knowledge coordinators."*[(1)]

Given the importance of knowledge as an economic factor, the practical management of knowledge becomes relevant: how to acquire, share, distribute and access knowledge within an organization and within business processes are questions that need to be answered. IT technologies, among other fields such as human resource management and corporate cultures, provide significant answers to these questions in the form of *Organizational Memory Information Systems (OMIS)*. This thesis focuses on the following problems:

- *Enterprise Modeling*: OMIS are not built within a vacuum - they exist in lively, evolving organizations and they need to support knowledge management tasks within business processes. Support is required for building, introducing and assessing OMIS, which helps to oversee the enormous amount of detail relevant to the building stage.

1. (World Bank, World Development Report: Knowledge for Development, Oxford University Press, 1998/99, Chapter 1, Page 1, see also: http://www.worldbank.org/wdr/wdr98/index.htm).

- *Maintaining an OMIS.* Centralized approaches require a central authority to maintain an OMIS, which hampers scalability since it constitutes a maintenance bottleneck. Distributed approaches to maintenance resolve the bottleneck.
- *Text documents vs. formalized knowledge.* Most knowledge within an organization is represented as text documents. Information retrieval techniques may be used to find and search documents, but they lack the capacity to answer queries which require the content of several documents. To answer such queries, a formalization of the knowledge is necessary. In conventional approaches machine-readable formal knowledge is separated from human readable text documents. As a consequence, maintenance of formal knowledge is expensive and does not scale well. Techniques that assist the integration of formal knowledge and text documents help to resolve this bottleneck.

1.1.2 The Web

Developments in recent years have brought about a revolution in how information is received and deployed. The World Wide Web has changed the way people spend their work and leisure time. To cite Tim Berners-Lee, the inventor of the World Wide Web:

> *"The dream behind the Web is of a common information space in which we communicate by sharing information. Its universality is essential: the fact that a hypertext link can point to anything, be it personal, local or global, be it draft or highly polished."*[(2)]

The dream has largely become a reality. With the new infrastructure it is possible to provide information to and from almost every entity in human society. An enormous pool of information and knowledge has become available, aiming at human consumption. The variety and mass of available information has its drawbacks: without additional infrastructures - search engines, web directories, and Web portals - it would be almost impossible to find useful information on the Web.

Even with the help of search engines, finding information on the Web requires human activities and attention, as search engines are not able to understand the semantics of the posed query and only deliver human readable documents.

Framing precise queries and getting precise answers back from the Web is not possible, even when search engines are deployed. Some of the techniques indicated above help.

1.2 Goals

The central hypothesis of this work is that the Web can be turned into a knowledge base, based on the current Web infrastructure. This technology is capable of Knowledge Management within organizations.

To prove the central hypothesis, the thesis needs to establish a number of subgoals:

1. *Knowledge Management* needs to be *supported* with an appropriate modeling and assessment approach, which provides a schema for modeling a particular knowledge management strategy in a specific situation.

2. See http://www.w3.org/People/Berners-Lee/ShortHistory.html, retrieved 12/20/2001.

2. A foundation is required for turning the Web into a knowledge base. The thesis shows that *Ontologies* provide this foundation. An appropriate representation technology is required, which effectively represents and reasons with ontologies in a Web setting.
3. *Formal knowledge* needs to be represented, integrated and extracted from Web documents. To avoid information being represented twice in Web pages - one time as human-readable text, the other as machine-readable metadata - it is beneficial to integrate the two sets of information.
4. *A query mechanism* is needed, which helps to query the formal knowledge extracted from Web pages, using the vocabulary defined in the ontologies.
5. A *case study* is required to demonstrate the feasibility of the approach for Knowledge Management
6. *Beyond Knowledge Management*. For validation purposes, the first steps beyond Knowledge Management (which, in the final analysis, is still aimed at providing information for human consumption) are investigated by examining what additional infrastructure is required to make the Web understandable by machines.

1.3 Approaches

A new modeling approach will be developed to support Knowledge Management. This is effected by combining different, orthogonal aspects of Knowledge Management. These aspects are: the *human factor*, *organization*, and *technology*, combining them with a process-oriented view of Knowledge Management and a view based on management science. The result is the *Knowledge Management Cube*.

The core of the thesis is the development of a Knowledge Management system called Ontobroker. Ontobroker provides a set of languages, tools and techniques, which create a pathway to the subgoals listed in the last section.

- The usage of logic programming techniques for the *representation of and reasoning with Ontologies* is investigated. A method will be developed that shows how more expressive, declarative languages can be compiled into normal logic programs.
- An annotation language for HTML and XML documents is developed to annotate these documents ‘redundancy-free’ with ontology-based metadata. The metadata is exploitable by the inference engine. ‘Redundancy-free’ is used here to denote that the actual text is used in the annotation - if the document is changed then the annotation is also changed.
- Novel query interfaces will be developed, which support the formulation of queries using a large ontology. The query interfaces support browsing and selection of terms using a new visualization method for ontologies.
- A case study will be presented - the Knowledge Annotation Initiative of the Knowledge Acquisition community. Ontobroker is applied in a large scale Knowledge Management case study, and assessed using the new Knowledge Management modeling approach. Within this case study an ontology was developed and deployed for the annotation of Web documents within the Knowledge Acquisition community.

- Beyond Knowledge Management, this thesis investigates which technologies are required for building the *Semantic Web* - a Web not primarily aimed at human consumption, but rather at intelligent agents. An *information foodchain* is defined for the Semantic Web by generalizing the Ontobroker approach and investigating how to build ontology representation languages on top of semi-structured data.

1.4 Outline

This thesis has three parts: the first part (chapters 2-5) introduces the different dimensions of knowledge management and develops a unified model for characterizing knowledge management strategies and tools. The second part (chapters 6-11) develops Ontobroker, the ontologies and metadata based knowledge management strategy, by exploiting ontology based metadata annotations of documents. The third part (chapters 12-16) generalizes part II by discussing more advanced Semantic Web topics such as an information foodchain for Semantic Web applications, the Resource Description Framework (RDF), and how to build Knowledge Representation Languages on top of RDF. The thesis ends with a summary and conclusions and by posing future research questions. In the following, the content of the chapters is described in more detail:

- Chapter 2 begins by clarifying the foundations of this thesis. A brief summary is given of different definitions of Knowledge Management in general and the notion of knowledge. Since information technology support for Knowledge Management is the focus of this thesis, the thesis looks at knowledge from a Computer Science perspective. Different knowledge representation approaches, from a static perspective (ontologies) and a dynamic perspective (Problem Solving Methods), are briefly discussed and contrasted to the concept of knowledge in CSCW, where knowledge is usually associated with documents.
- Chapter 3 reconsiders the different dimensions of Knowledge Management. Dimensions discussed are on the one hand the *human factor*, the *organizational*, and the *technological* aspects, and on the other hand a process oriented view of knowledge management with aspects such as "*knowledge acquisition*", "*knowledge assessment" and "knowledge dissemination*". These dimensions are amalgamated by deriving a unified model for the assessment of knowledge management strategies - the Knowledge Management Cube (KMC). The KMC is later used in this thesis to assess Ontobroker as a Knowledge Management approach.
- Chapter 4 discusses the technological aspects of Knowledge Management by defining the notion of Organizational Memory Information Systems (OMIS) and providing several classifications of OMIS (e.g., a process-oriented vs. product-oriented viewpoint). The notion of an OMIS is contrasted with other well-known notions of information systems, like Knowledge Based Systems.
- Chapter 5 develops a modeling schema for building OMIS: the chapter shows that task-oriented modeling is useful for connecting enterprise models on the process level with models representing concrete tasks suitable for information systems. Concrete notions based on UML for the enterprise models are provided. The modeling approach is compared with other approaches such as CommonKADS and ARIS. The chapter argues that PSMs, which were introduced in AI to capture and structure dynamic knowledge, are suitable for representing and structuring business processes. The chapter also presents an example application of the modeling schema by modeling the work processes of industry designers. Based on the created UML models, Knowledge Management strategies for the industrial design process are identified.

The goal of part II of this thesis is to show that the Web can be turned into an Organizational Memory Information System that can be queried like a knowledge base. The suggested approach called Ontobroker is based on ontology-based metadata. This part provides the means to create and deploy such metadata, which is a first step in transforming the Web into a distributed Knowledge Based System. Necessary components are identified and described in detail: an ontology, an inference engine able to reason with the ontology, an annotation language for HTML and XML documents, tool support for the annotation process, and a query interface for querying the Knowledge Base.

- Chapter 6 demonstrates that conventional search engines (e.g., GOOGLE) don't provide the exact query capabilities necessary to answer directed queries on the Web. The chapter then introduces the proposed alternative: the Ontobroker approach and architecture. The rationales for the architecture design decisions are provided, along with a clarification of how all the components work together. The next chapters provide detailed explanations of the components of the Ontobroker system.
- Chapter 7 presents the design and implementation of SiLRI, the representation language used to represent ontologies and the inference engine of Ontobroker. The representation language is a variant of F-Logic, enhanced with the capability to provide syntactical expressive rule bodies based on full first-order logic. The inference engine of SiLRI is based on Horn logic with a closed world negation at its core. The SiLRI representation language is compiled into Horn logic by negation. The SiLRI language can be regarded as a domain specific representation language. This chapter provides a method for constructing domain specific representation languages and compiling them back into available inference engines. As an example of building a new domain specific language, the chapter illustrates the combination of Chronolog, a language for linear temporal logic and F-Logic, which results in the language C-F-logic, which is capable of expressing the dynamic behavior of objects over time. In the forthcoming chapters F-logic is used to represent the ontologies and the instance data within Ontobroker. Furthermore, the chapter provides a new translation procedure realized using a variant of the Lloyd-Topor transformation.
- Chapter 8 handles the problem of document annotation. The chapter first classifies the Ontobroker approach to metadata. HTML-A is introduced - the annotation language used within Ontobroker for the ontology-based annotation of HTML documents. The chapter then presents OntoPad, a metadata-enhanced HTML-editor that supports the creation of ontology-based annotations of documents. HTML-A is then generalized to XAL; the XML Annotation Language (XAL). The additions are: instead of just annotating the anchor tag, XAL may be used to annotate arbitrary tags; multiple ontologies may be used for the annotation of the same document. XAL enables the reference and linking to arbitrary information in the document; regular expressions may be used to filter information inside annotations; XAL also supports the annotation of regular structures like lists or tables with minimal effort.
- Chapter 9 presents the consumer aspects of Ontobroker: the Information Crawler and the Query Interfaces. The Information Crawler extracts the metadata from HTML pages and translates the data into a format that is understandable by the inference engine. The query interfaces allow querying of the knowledge base and support the interactive construction of queries, guided by the ontologies.
- Chapter 10 presents a case study of Ontobroker. Ontobroker is characterized as a Knowledge Management System, using the Knowledge Management Cube developed in Part I of this thesis. A case study of the Knowledge Annotation Initiative of the Knowledge Acquisition

Community ($(KA)^2$) is presented, describing how the ontology was developed. The annotation process is detailed and results of $(KA)^2$ from a Knowledge Management point of view are also presented.

- Chapter 11 features related work that provides complete frameworks comparable to Ontobroker. Ontobroker is compared in detail to SHOE and WebKB. A focus of the comparison is the reasoning capabilities of the different systems.

The third part of this thesis generalizes the work presented in the second part and introduces the notion of the *Semantic Web*.[3] The Semantic Web is aimed at machine processable information and services (in contrast to HTML), with agents performing tasks for their users. This part provides the foundations for creating ontologies on the Web, based on semi-structured data and knowledge representation principles.

- Chapter 12 suggests an information foodchain for agents on the Web that presents necessary steps and technologies to reach an automated Web and shows how the technologies used in the foodchain are interdependent. Distinctions are drawn between metadata and ontology creation technologies, middleware (which articulates different ontologies and provides inference services), and end-user applications, such as agents or a community portal.
- Chapter 13 provides requirements for Knowledge Representation on the Web. An introduction to RDF and RDF Schema is provided. Then, RDF is compared to the Object Exchange Model (OEM), a semi-structured data format defined for data integration and frame-based knowledge representation systems.
- Chapter 14 compares RDF and XML as a foundation for Knowledge Representation on the Web. In this chapter it is shown why XML is not suitable as a foundation for the Semantic Web and how Knowledge Representation languages are constructed on top of RDF.
- Chapter 15 presents two ontology languages built on top of RDF: OIL and DAML+OIL. Both languages are based on Description Logics. A critical assessment of both languages is provided.
- Chapter 16 provides a summary and conclusions, as well as providing starting points for future research work.

Figure 1.1 provides a high-level overview of the different topics of this thesis.

3. cf. http://www.SemanticWeb.org

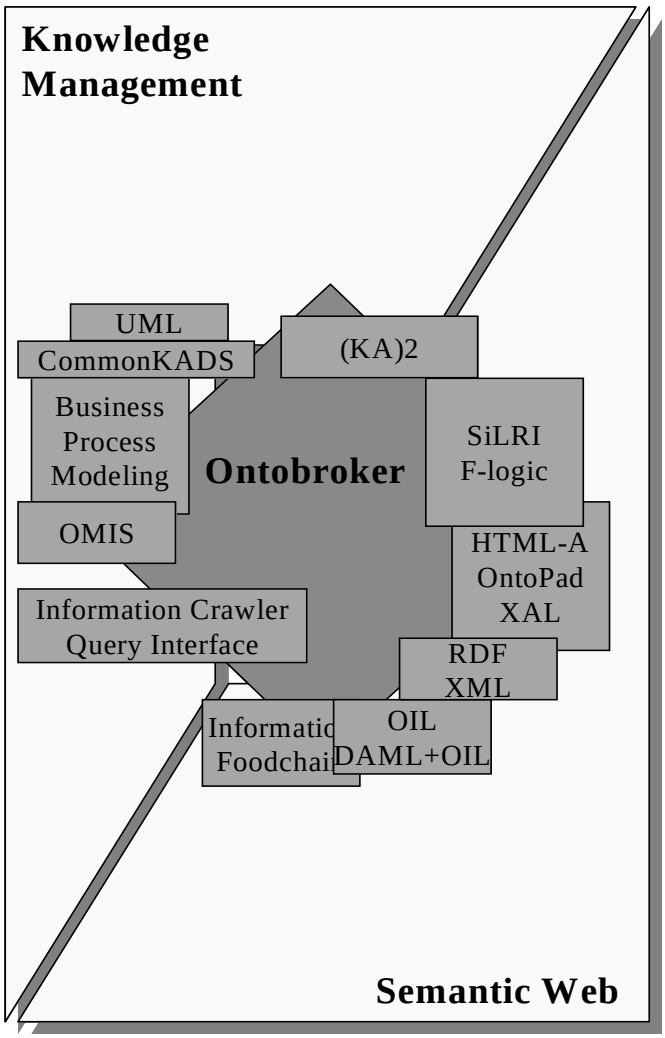


Fig. 1.1. High Level Overview of the Topics in this Thesis

PART I KNOWLEDGE MANAGEMENT AND ORGANIZATIONAL MEMORIES

"*Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world.*"
Albert Einstein

Chapter 2 Preliminaries: What is Knowledge and Knowledge Management?

This chapter gives a brief summary of different definitions of Knowledge Management and the notion of knowledge. Since information technology support for Knowledge Management is the focus of this thesis, the chapter discusses knowledge from a Computer Science (especially Artificial Intelligence and Computer Supported Cooperative Work) perspective.

2.1 What is Knowledge Management?

Interest in Knowledge Management in the business world cannot be pinpointed to a single reason, but is being promoted by the merging of a number of trends. Under the label of *lean management*, the last wave of enterprise reengineering efforts has removed middle management from many organizational structures. But for decades, just this level of employees were important stakeholders of know-how and experience about projects, customers, and products. They performed the important tasks of knowledge gathering, filtering, interpretation, and routing as a preprocessing step to preparing the decisions of upper management. Knowledge Management tries to compensate for (1) the loss of stable procedural knowledge by explication and formalization through using e.g., organization handbooks or workflow management systems, (2) the loss of project-, customer-, or project-related experiences and know-how by establishing best-practice or lessons-learned databases [van Heijst et al., 1998], and (3) the loss of the middle management information analysis and routing services through new IT solutions, e.g., intranets, data mining, or data warehouses [O'Leary, 1998c].

Together with the *globalization of businesses*, an enormous market pressure enforces ever-shorter product life cycles. On the other hand, modern information technology allows worldwide geographically dispersed development teams, virtual enterprises [Ribière & Matta, 1998] and close cooperation with suppliers, customer companies, and outsourced service providers. All these factors require complex communication and coordination flows, complex both in technical and in conceptual terms. The role of IT is to support the information and document distribution, to enable worldwide communication and synchronization. CSCW and groupware tools are often used in scenarios where jointly distributed collaboration is required. Furthermore, new customer-oriented management and quality principles like *Total Quality Management* (TQM) promote new styles of communication and decision-making in company departments. Following the *concurrent engineering* paradigm, expertise, information, and feedback from customers, as well as many previously separated and timely sequential development steps, are thrown together at very early stages in order to have a comprehensive and holistic view of requirements and design. This requires complex communication and collaboration by many people with different educational backgrounds, personal goals and perspectives. It also requires a combination of documents, information flows, and decision processes originating from previously separated groups of people. Here, IT can support cooperative work and store decision processes to enable subsequent decision tracing under altered conditions. It can also support the preparation of well-structured documentation required for quality certification.

All these business phenomena produce Knowledge Management-related activities comprising:

1. better exploitation of already available but insufficiently used documents,
2. formalization of business rules in workflows,
3. better usage of human skills and knowledge through CSCW techniques or enterprise yellow pages and competency databases, and
4. explication of experiences and know-how in best-practice databases, and much more.

Most of these activities can also be supported by information technology, and are in fact already partly supported by conventional systems [Davenport et al., 1996], [Bullinger et al., 1997]. However, the specific introduction of the term *knowledge* creates a different viewpoint: it is no longer sufficient to deliver huge amounts of information to users, instead it is important to support them in doing their *knowledge work*.

Having described the *Why* of Knowledge Management, some definitions from the literature about the '*What*' are given in the following: Knowledge Management (KM) is a holistic approach, which can be analyzed from different viewpoints. For this reason, it is difficult to give an exact definition. [Nonaka & Takeuchi, 1995] stress in their definition the importance of the distinction between tacit and explicit knowledge: "*Knowledge Management is the tacit and explicit knowledge framework for a dynamic human process of justifying personal belief toward the truth.*" Explicit knowledge is knowledge that is already extracted and consumable in books or other media. Tacit knowledge is not present in explicit form, and can often not even be articulated by a person who possesses the knowledge. Thomas Davenport [Davenport & Prusak, 1998] stresses the importance of the process and supply chain: "*Knowledge Management is a formal, structured initiative to improve the creation, distribution, or use of knowledge in an organization. It is a formal process of turning corporate knowledge into corporate value.*" This corporate value is also emphasized by Karl Erik Sveiby in [Sveiby & Lloyd, 1990]: "*Knowledge Management is the art of making money out of immaterial assets.*"

What remains to be clarified is how the *Knowledge* in Knowledge Management can be captured and processed. The following sections examine the technology areas which help to realize a Knowledge Management strategy, and also investigate what kind of knowledge is managed. Since information technology support for Knowledge Management is the focus of this thesis, knowledge is examined from a Computer Science perspective.

2.2 Knowledge in Computer Science

Figure 2.1, derived from [Aamodt & Nygård, 1995], serves as a basis for understanding the use of Knowledge in Computer Science. There is no fundamental difference in the representation of data, information and knowledge: everything is based on symbols. Only when examined together with the dimensions of Semiotics (syntax, semantics and pragmatics) is it possible to achieve a visible differentiation. For instance, the number '100' is abstract data, and has a special syntax (e.g., numbers are only allowed to consist of digits, a point and a sign). Data has no meaning, unless accompanied by added semantics such as 'km/h', which makes it a speed number. However, this is just information, and no human can act solely on this information. Only the addition of pragmatics leads to knowledge that is usable for decisions and actions (e.g., to reduce speed if '100km/h' represents a speed limit). Different subfields in computer science developed different interpretations of knowledge. The following sections discuss relevant subfields of computer science for Knowledge Management.

2.2.1 Knowledge in Artificial Intelligence

The area in Computer Science that is most influenced by the concept of knowledge is Artificial Intelligence (AI).

In AI concepts such as Knowledge Based Systems (KBS), Knowledge Level, Knowledge Modeling, and Knowledge Representation were invented and discussed (for an overview cf. [Studer et al., 1998], [Studer et al., 2000]). In the early 1980s the development of a KBS was seen as a *process* of transferring human knowledge to an implemented knowledge base. This transfer was based on the assumption that the knowledge, which is required by the KBS, already exists and only has to be collected and implemented [Musen, 1993]. It is interesting to note that research in AI indeed used the early definitions of knowledge in philosophy: it followed the ideas of Plato that knowledge is something inherently true. Eventually, this knowledge was implemented in some types of production rules, which were executed by an associated rule interpreter [Stefik, 1995]. A careful analysis of the various rule knowledge bases has shown, however, that the rather simple representation formalism of production rules did not support an adequate representation of different types of knowledge [Clancey, 1983]. Such a mixture of knowledge types, together with the lack of adequate justifications of the different rules, makes the maintenance of such knowledge bases very difficult and time consuming. Therefore, this transfer approach was only feasible for the

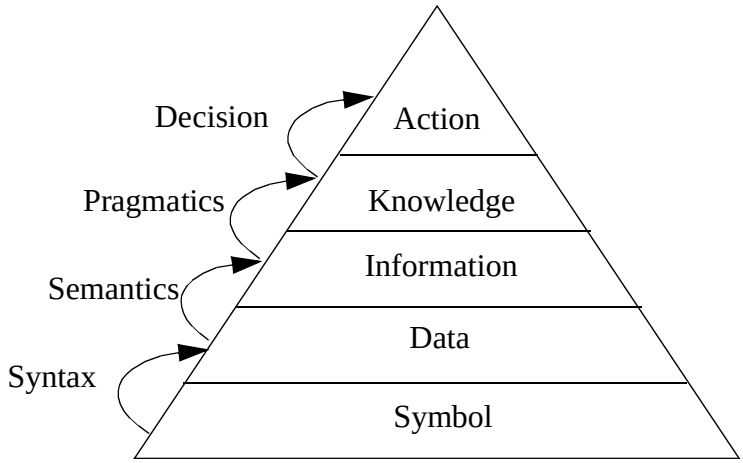


Fig. 2.1. Knowledge Pyramid based on [Aamodt & Nygård, 1995]

development of small prototypical systems as it failed to produce large, reliable and maintainable knowledge bases. It was recognized that the assumption of the transfer approach (that knowledge acquisition is the collection of already existing knowledge elements) was not correct due to the important role of tacit knowledge for an expert's problem-solving capabilities.

These deficiencies resulted in a paradigm shift from the transfer approach to the *modeling approach*. This paradigm shift was also inspired by Newell's notion of the *Knowledge Level* [Newell, 1982]. This knowledge level proposes the modeling of knowledge independent from its implementation and structuring knowledge models with respect to different knowledge types.

In the modeling framework, constructing a KBS means building a computer model with the aim of realizing problem-solving capabilities comparable to a domain expert. Since an expert is not necessarily aware of some knowledge that is part of his or her skills, this knowledge is not directly accessible, but has to be constructed and structured during the knowledge acquisition phase. This knowledge acquisition process is therefore seen as a model construction process.

Some observations can be made about this modeling view of the building process of a KBS.

- Like every model, such a model is only an *approximation* of reality.
- The modeling process is a *cyclic* process. New observations may lead to a refinement, modification, or completion of the already constructed model. On the other hand, the model may guide further acquisition of knowledge.
- The modeling process is dependent on the subjective interpretation of the knowledge engineer. Therefore this process is *faulty* and an evaluation of the model with respect to reality is indispensable for the creation of an adequate model.

[Clancey, 1983] provided several examples where knowledge engineers encoded implicit control knowledge by ordering production rules and premises of these rules such that the generic inference engine exhibited the dynamic behavior they aimed at. Thus, a large part of the modeling activities for building KBS is spent on making explicit the control knowledge that is contained in an expertise. This is the rationale that underlies Problem-Solving Methods (PSMs) [Benjamins & Fensel, 1998a], [Fensel, 2000]. PSMs refine the generic inference engines mentioned above to allow for a more direct control of the reasoning process. Since this control knowledge is specified independently from the application domain, reuse of this strategical knowledge is enabled for different domains and applications. In addition, PSMs abstract from a specific representation formalism in contrast to the general inference engines that rely on a specific representation of the knowledge. Finally, PSMs decompose the reasoning task of a knowledge-based system in a number of subtasks and inference actions that are connected by knowledge roles. Several problem solving method libraries are now available (see e.g., [Breuker & Van de Velde, 1994], [Benjamins, 1995], [Motta, 1999]) and a number of problem-solving method specification languages have been proposed, ranging from informal notations (e.g., CML [Schreiber et al., 1994]) to formal modeling languages (e.g., KARL [Fensel et al., 1998c], see [Fensel, 1995] for a survey). PSMs are introduced in more detail in Section 2.2.3.

Besides knowledge modeling also *knowledge representation* is also an important field of research in computer science and AI. Knowledge representation may be divided into two subfields: symbolic and subsymbolic knowledge representation, as demonstrated in connectionistic approaches. Examples for symbolic knowledge representation formalism are frames [Minsky, 1975] and semantic networks. Frames are related to object-oriented formalisms but usually do not have a formal foundation. Other Knowledge representation formalisms are based on logic. Frames and

semantic networks led to formalized counterparts, e.g., in the Frame-Logic approach [Kifer et al., 1995] and in Description or Terminological Logics (cf. [Nebel, 1996]), which grew out of the non-logic-based formalisms (e.g., from KL-ONE, cf. [Brachman, 1979], [Brachman & Schmolze, 1985]). Looking at the historical roots, McCarthy first proposed the use of logic in 1959 [McCarthy, 1959] as a basis for knowledge representation languages. In [McCarthy, 1989], John McCarthy wrote:

> *"Expressing information in declarative sentences is far more modular than expressing it in segments of computer programs or in tables. Sentences can be true in a much wider context than specific programs can be used. The supplier of a fact does not have to understand much about how the receiver functions or how or whether the receiver will use it. The same fact can be used for many purposes, because the logical consequences of collections of facts can be available."*

Classical logic of predicate calculus served as the main technical tool for the representation of knowledge in AI. It has a well-defined semantics and a well understood and powerful inference mechanism, and it proved to be sufficiently expressive for the representation of mathematical knowledge. It was soon realized, however, that for the representation of common sense knowledge, first order predicate calculus is inadequate for several reasons: although automated theorem proving has made remarkable progress in recent years, it is still difficult to reason with huge amounts of axioms and facts. Furthermore, for representing common sense knowledge, first order predicate logic turned out to be inadequate: common sense is non-monotonic, which means one has to withdraw deductions that were performed earlier. This leads to non-monotonic reasoning approaches, e.g., in logic programming (see [Baral & Gelfond, 1994], [Brewka & Dix, 1999] for an overview) as knowledge representation formalism.

2.2.2 Ontologies

The term 'Ontology' (Greek. on = *being*, logos = *to reason*) was originally coined in philosophy. The Encyclopedia Britannica remarks about 'Ontology':

> *"The theory or study of being as such; i.e., of the basic characteristics of all reality. Though the term was first coined in the 17th century, ontology is synonymous with metaphysics or "first philosophy" as defined by Aristotle in the 4th century BC. Because metaphysics came to include other studies (e.g., philosophical cosmology and psychology), ontology has become the preferred term for the study of being. It was brought into prominence in the 18th century by Christian Wolff, a German rationalist,..."*[(1)]

The use of the term Ontology in computer science has a more practical meaning than its use in philosophy. The study of metaphysics is not in the foreground in computer science, but what properties a machine must have to enable it to process data is being questioned within a certain domain of discourse. Here ontology is used as the term for a certain artifact.

Tom Gruber's widely cited answer to the question: *"What is an Ontology?"* is:

> *"An ontology is a specification of a conceptualization."* [(2)]

In [Gruber, 1993] this statement is elaborated on:

---

1. see Encyclopedia Britannica Online. <http://search.eb.com/bol/topic?eu=58583&sctn=1>, 2001]
2. see http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

> *"A body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them* [Genesereth and Nilsson, 1987]. *A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly."*

Specification means in this context an explicit representation by some syntactic means. [Gruber, 1993] is suggesting the Knowledge Interchange Format (KIF) [KIF, 1998] as a representation language for ontologies. KIF is a predicate logic-based language, and while predicate logic-based representation formats for ontologies are widely used in the AI community, the definition of an ontology as a specification of a conceptualization is more far reaching, and does not necessarily involve a predicate logic based representation language. However, most approaches to ontology modeling agree on the following primitives for representation purposes:

- The distinction between *classes* and *instances*, where classes are interpreted as a set of instances. Classes may be partially ordered using a binary relationship *is_a*, which can be interpreted as a subset relationship between two classes. The fact that an object is an element of a certain class is usually denoted with a binary relationship *instanceOf* or *hasType*.
- A set of properties (also called attributes or slots). Slots are binary relationships defined by classes, which usually have a certain *domain* and a *range*. Slots might be used to check if a certain set of instances with slots is valid with respect to a certain ontology.

[Guarino, 1997] distinguishes between several types of ontologies:

- *Top-level ontologies* aim at describing very general concepts like space, time, matter, object, event, action, etc., which are supposed to be application independent: it seems therefore reasonable to have unified top-level ontologies for large communities of users and applications. Figure 2.2 shows three top-level ontologies, restricted to the `is_a` hierarchy: the first three levels of the IEEE Standard Upper Ontology (SUO)[1] Version 1.16, the ABC Top Level Ontology [Lagoze and Hunter, 2001], and a top level ontology suggested by John Sowa [Sowa, 2000]. The IEEE Standard Upper Ontology effort is aiming at an upper ontology that will enable computers to utilize it for applications such as data interoperability, information search and retrieval, automated inference, and natural language processing.
  The ABC Top Level Ontology model has been developed within the Harmony International Digital Library project to provide a common conceptual model to facilitate interoperability between meta-data ontologies from different domains.
  John Sowas's Top Level Ontology is derived from "*ongoing research in philosophy, linguistics, and artificial intelligence.*" These three top-level ontologies illustrate the problem in that they aim to be useful for a large number of possible applications, since they are supposed to provide the foundations for more specific tasks. Although all ontologies are aiming at addressing similar problems, they are quite different. A top-level ontology mirrors the way the world is experienced. Given the enormous diversity in the world, it seems highly unlikely that the same top-level ontology can be used by a large variety of users. The view of the world, and thus the construction of top-level ontologies, is much too dependent on such matters as personal

1. http://suo.ieee.org/

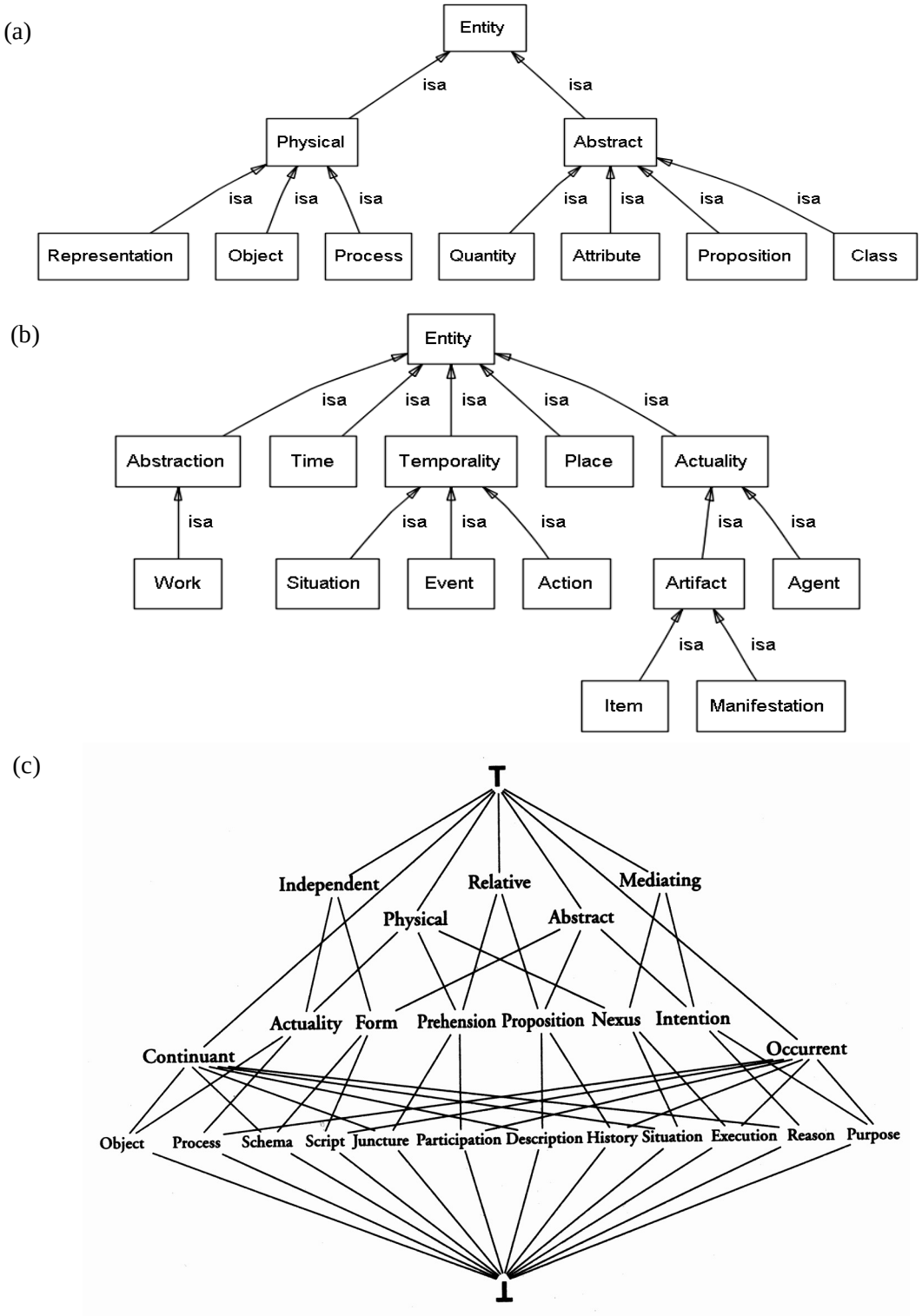


Fig. 2.2. Different Top-Level Ontologies: (a) shows the first three levels of the IEEE Standard Upper Ontology[1] v. 1.16, (b) shows the ABC Top Level Ontology [Lagoze and Hunter, 2001], (c) shows a top-level ontology by John Sowa [Sowa, 2000].

1. http://suo.ieee.org

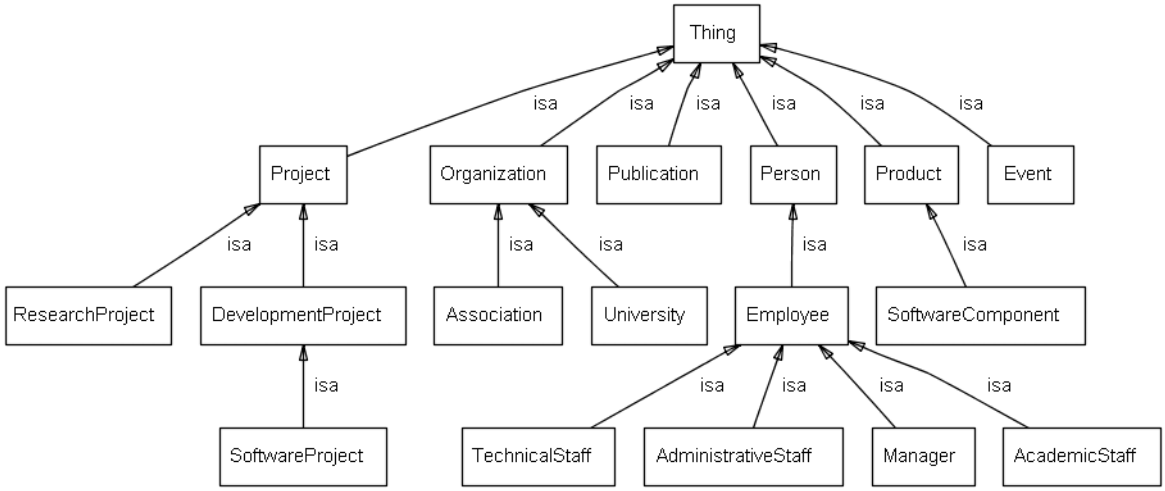


Fig. 2.3. Part of the (KA)2 Domain Ontology describing a scientific community

experience, education, culture, etc. Even such efforts as the IEEE Standard Upper Ontology, which don't have much cultural diversity (e.g., most participants listed[1] are from the US), show effects of diverging world views.

- *Domain ontologies* and *task ontologies* describe, respectively, the vocabulary related to a generic domain (such as medicine or automobiles) or a generic task or activity (such as diagnosing or selling), by specializing the terms introduced in the top-level ontology. Examples of ontologies for medicine include the Unified Medical Language System (UMLS)[2] [Tuttle et al., 1991], a rich single corpus of biomedical names used by health care providers in the United States, or the GeneOntology[3] [Gene Ontology Consortium, 2001], an ontology aimed at the dynamic, controlled vocabulary that can be applied to eukaryote[4] even as the knowledge about genes and protein roles in cells is accumulating and changing. Another example of a domain ontology is depicted in Figure 2.3: the $(KA)^2$ ontology, describing a scientific community, was used in the Ontobroker project (see Chapter 6 for more details about Ontobroker). A task ontology for parametric design is presented in [Fensel et al., 1997].
- *Application ontologies* describe concepts dependent both on a particular domain and task, which are often specializations of both the related ontologies. These concepts often correspond to roles played by domain entities while performing a certain activity, like replaceable unit or spare components.

[van Heijst et al., 1997] introduces *representation ontologies*. These describe a classification of primitives used by a knowledge representation language. An example of a representation ontology is the *Frame ontology* [Gruber, 1993], defining the frame-oriented primitives of Ontolingua. A more recent example of a representation ontology includes the RDF Schema specification of DAML+OIL[5] (see Chapter 15 for more details).

1. see http://suo.ieee.org/subscribers.html for a list of participants (retrieved 20 December 2001)
2. http://www.nlm.nih.gov/research/umls/ (retrieved 10 December 2001)
3. http://www.geneontology.org/ (retrieved 10 December 2001)
4. For the biological layman: Organisms with nucleated cells (retrieved 10 December 2001)
5. http://www.daml.org/2001/03/daml+oil.daml (retrieved 10 December 2001)

Note that the distinction between domain ontologies and other ontology types is sometimes useful, but by no means a strict classification: by adjusting the domain every ontology can be classified as a domain ontology.

The following list contains examples of academic explorations into the use of ontologies (derived from [Erdmann, 2001]):

- Natural Language Processing and Machine Translation (e.g., Wordnet [Miller, 1990], [Dahlgren, 1995]).
- Knowledge Engineering (esp. Problem Solving Methods [Fensel, 2000]).
- Knowledge Management (e.g., [Abecker et al., 1997], [Staab & Schnurr, 2000], [Sure et al., 2000]).
- Engineering and Construction (e.g., [Borst, Akkermans 97]).
- Information Retrieval and Information Articulation (e.g. [Kashyap, 1999], [Mena et al., 1998], [Mitra et al., 2000], [Mitra et al., 2001]).
- Web Catalogs (e.g., Yahoo [Labrou & Finin, 1999]).
- Ontology and meta-data based Search Engines (e.g. Ontobroker [Decker et al., 1999], SHOE [Heflin, 2001], OntoSeek [Guarino et al., 1999]).
- Digital Libraries (e.g., [Amann & Fundulaki, 1999])
- Enhanced User Interfaces (e.g., [Kessler, 1995]).
- Software Agents (e.g., [Gluschko et al., 1999], [Smith & Poulter, 1999]).
- Business Process Modeling: (e.g., [Decker et al., 1997], [TOVE, 1995], [Uschold et al., 1998].

Interestingly, there have been many industrial efforts to define vocabularies for different domains, without necessarily calling them ontologies but still fulfilling Gruber's general definition. These efforts include:

- the *Papinet effort,*[1] which develops, maintains and supports the implementation of global electronic business transaction standards for parties engaged in the buying, selling, and distribution of forest and paper products.
- the *HR-XML Consortium,*[2] an independent, non-profit organization dedicated to the development and promotion of standardized XML vocabularies for human resources (HR).
- the *Business Process Management Initiative,*[3] which aims to standardize management of business processes that span multiple applications, corporate departments, and business partners over the internet.
- the *ebXML Effort,*[4] which aims to provide an open XML-based infrastructure enabling the global use of electronic business information in an interoperable, secure and consistent manner by all parties.

1. see http://www.papinet.org/
2. see http://www.hr-xml.org/
3. see http://www.bpmi.org/
4. see http://www.ebxml.org/

These efforts use grammars that define document structures for the definition of their conceptual model. Chapter 14 takes a closer look at XML and other representation mechanisms for defining ontologies and argues that grammars are not an appropriate representation formalism.

In order to create an effective ontology of a particular domain, the ontology construction should be supported by (1) the use of a methodology that guides the ontology development process and (2) tools to inspect, browse, codify, modify and download the ontology. Examples of such methodologies include OntoKick [Sure, 2001a], METHONTOLOGY [Fernandez & Gomez-Perez, 1999a], Uschold's and Gruninger's methodology [Uschold & Gruninger, 1996] and that of Gruninger and Fox [Gruninger & Fox, 1995]. These methodologies have in common that they begin with an identification of the purpose of the ontology.

However, methodologies are more effective if they are supported by tools. Currently, two different types of tools for supporting the construction of ontologies can be distinguished.

- Single-User Environments, where a single user defines an ontology, and
- Multi-User Environments, where support is provided for the collaborative development of ontologies.

Single User Environments
In the following, different examples are provided for general types of ontology editors.

Class Browser in development environments for object-oriented languages, such as SmallTalk 80 and C++, can be regarded as basic tools for ontology editing. In these environments, developers can edit class hierarchies graphically. Such editors, however, are designed to be used by programmers and are not primarily intended for non-programmers.

In addition to these language-oriented tools, there are ontology editors based on knowledge-engineering and knowledge-acquisition principles. The Ontolingua server [Farquhar et al., 1997] is a set of tools that supports ontology development on the World Wide Web. These tools allow developers to browse, edit and publish ontologies using Web technologies (e.g., CGI scripts and Java).

*Rational Ros*e is a commercial visual-modeling environment for software engineering that allows developers to model conceptual structures and to generate source code (stubs) for different programming languages from them. Rose provides four major views: the use-case, logical, component, and deployment views. The logical view is equivalent to an ontology editor that supports the drawing of class diagrams.

Protégé (cf. [Eriksson et al., 1999]) is an integrated environment for ontology editing and metadata tool generation. Thus, an important difference between the above-listed ontology editors and the Protégé approach is that an essential requirement for Protégé is to generate knowledge-acquisition or (metadata) tools instead of just supporting ontology editing.

Protégé includes a set of tools that support the development of knowledge-based systems. Therefore, Protégé consists of two different parts: one part supporting the development of problem solving methods (see Section 2.2.3) and another part concentrating on the generation of domain-specific ontology-acquisition tools. Other examples of ontology editors include OntoEdit [Sure & Studer, 2001] and OILEd[1].

1. see http://img.cs.man.ac.uk/oil/

Multi User Environments
If there is no accepted authority that can provide a standard ontology, joint consensual creation of ontologies is needed.

Tadzebao and *WebOnto* (cf. [Domingue 1998]) aim at the distributed development of ontologies. WebOnto supports collaborative browsing, creation, and editing of ontologies by providing a direct manipulation interface that displays ontological expressions. WebOnto is easy to use, yet has facilities for scaling up to large ontologies. Finally, WebOnto complements the ontology discussion tool Tadzebao. Tadzebao supports asynchronous and synchronous discussions on ontologies. Asynchronous discussions are necessary because an ontology design team or community may be spread over large time zones. Synchronous discussions are useful a) when a dialogue is composed of many small utterances, b) when there is time pressure, or c) to give users a sense of belonging to a team or community.

Tadzebao guides knowledge engineers around ongoing dialogues on ontologies. Within Tadzebao, dialogues are centered around a notepad, which contains a series of pages. Each notepad page can contain a mixture of text, GIF images, hand drawn sketches and ontology components.

Other collaborative ontology development tools include e.g., the *Ontosaurus* browser [Swartout et al., 1996], which is a Web-based ontology editor similar to Ontolingua. However, Ontosaurus uses an HTML-interface and does not support other methods of communication as Tadzebao does.

2.2.3 Representing Procedural Knowledge

Problem-solving methods (PSMs) [Eriksson et al., 1995], as already mentioned, are used in most of the current knowledge engineering frameworks to capture and represent procedural knowledge. The technologies developed for PSMs are useful in a variety of application areas, where procedural knowledge has to be captured, described and processed. In the following, the PSM model as developed in [Angele et al., 1996] is described and its differences to the model described in [Fensel, 2000] are pointed out.

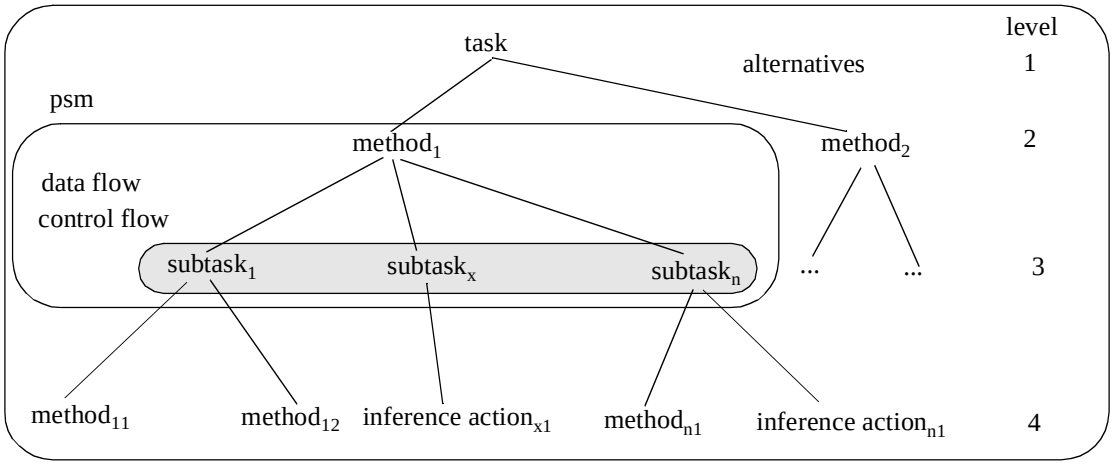


Fig. 2.4. Embedding the notion of PSM into the Method-to-task Paradigm

Given a specific task, a PSM is the specification of the functionality (or high-level description) of the problem solving behavior of the system to be built. It is a description of how the functionality can be achieved. A PSM decomposes a task into new subtasks. According to the divide-and-conquer principle this is done repeatedly down to a level of subtasks for which the solution is obvious and can be written down in a straightforward manner as an elementary inference action. In addition to

the decomposition, PSM has to specify the control flow of the subtasks (i.e. the ordering in which the subtasks can be accomplished), and the data flow (i.e. which data is used in which step). There are several possibilities to represent control flow and data flow.

While the odd levels in the hierarchy of Figure 2.4 are responsible for defining the goals, the even levels offer solutions for how to reach the goals. Solving particular tasks usually also involves domain knowledge that a particular PSM has to import.

Fensel's approach to specification of PSMs (in [Fensel, 2000]) relates tasks, problem-solving methods, and the domain model by means of an *adapter.*

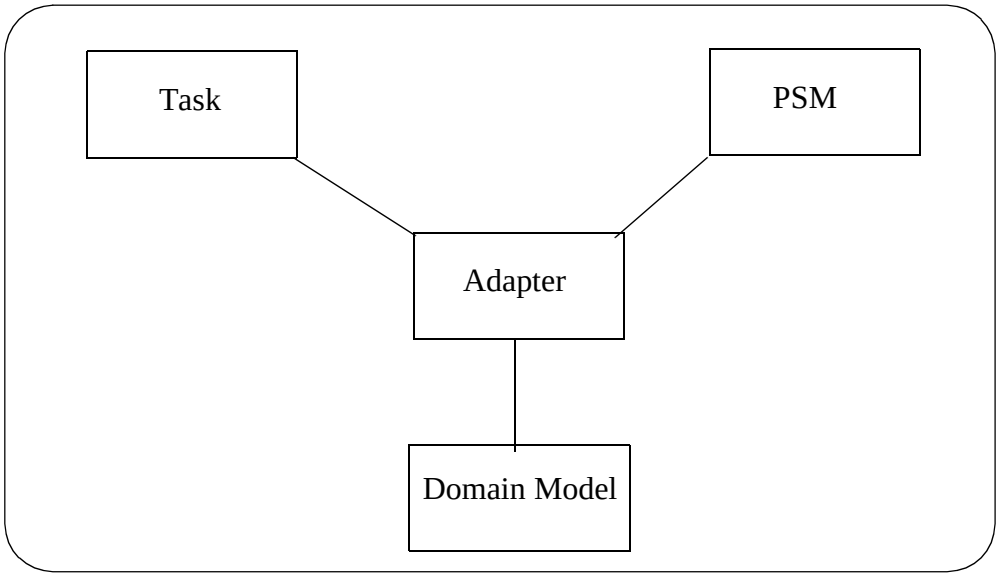


Fig. 2.5. Fensel's Four Component Architecture for Knowledge-based Systems.

The adapter is necessary to adjust the three other (reusable) parts to each other and to the specific application problem. This new element is used to introduce assumptions and to map the different terminologies.

2.2.4 Knowledge and Computer Supported Cooperative Work

According to [Nichols and Twidale, 1999] *Computer Supported Cooperative Work* (CSCW) is a research area that examines issues relating to the design of computer systems to support people working together. CSCW acknowledges that people work together as a way of managing complex tasks, where not all tasks can be automated. [Nichols and Twidale, 1999] argue that the broad definition is in part a reaction to what has been seen as a set of implicit design assumptions in many computer applications - that they are intended to support users to do their work on their own. System designers usually minimize the effects of a shared activity and try to create the illusion of the (presumed ideal) case of exclusive access to resources.

Thus, it is sensible to design systems that allow users to collaborate more effectively. Such systems can also open up opportunities for collaboration that have previously been impossible, overly complex or too expensive; for example, working not merely with colleagues in the same office but via video and audio links with colleagues in a different building or on a different continent. CSCW has a strong interdisciplinary tradition, drawing on researchers from computer science, sociology, management, psychology and communication (for more details about the context of CSCW cf.

[Bannon & Hughes, 1993]). Several possibilities for collaboration exist: people can not only collaborate in the same place (co-located) or in different places (remote) but also collaborate at the same time (synchronous) or separated in time (asynchronous). Figure 2.6, taken from [Nichols and Twidale, 1999], illustrates the different possibilities.

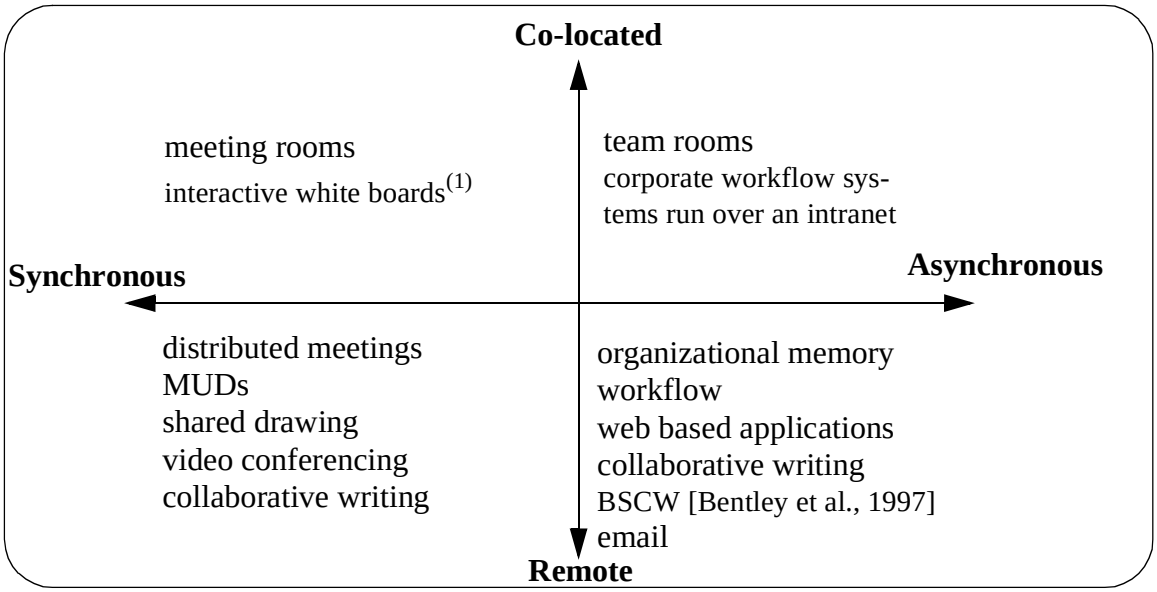


Fig. 2.6. The CSCW Spatial and Temporal Quadrants (taken from [Nichols and Twidale, 1999])

The type of knowledge managed by CSCW systems is usually informal and document centered. Users in typical CSCW scenarios create and edit documents together. There is a noticeable difference between the definitions of ‘knowledge’ in AI and CSCW: AI is typically dealing with formalized knowledge, whereas in a CSCW context, since formalization is very difficult and costly, the time necessary to formalize knowledge contained for example in documents is usually not spent. In order to enable the usability of AI techniques in CSCW and Knowledge Management systems this difference will need to be overcome. One possible way is to develop cost-effective techniques that help to formalize knowledge gradually - especially knowledge contained in documents.

Chapter 3 Dimensions of Knowledge Management[1]

This chapter takes a deeper look at Knowledge Management. The definitions of Knowledge Management mentioned in Section 2.1 do not provide a practical starting point for how to establish a Knowledge Management strategy and what viewpoints have to be taken into account. To give a more practical view of Knowledge Management, this chapter introduces from the literature two existing models used to set up and classify Knowledge Management strategies. It shows how to integrate both perspectives and define a new, comprehensive, and multidimensional model: the Knowledge Management Cube (KMC). The cube enables the modeling of Knowledge Management entities with its most important facets.

3.1 Elements of Knowledge Management

Knowledge Management cannot be realized by focusing on a single facet, since many aspects are involved in the creation of a strategy for building a Knowledge Management System. To focus, for example, on technology (e.g., setting up a corporate intranet) is insufficient, if co-workers are not willing to share their knowledge with each other. One the other hand, technology has the potential to enable effective Knowledge Management. An effective Knowledge Management strategy has thus to take a holistic view of the field. The model depicted in Figure 3.1 is often used to illustrate the different dimensions of Knowledge Management (cf. [Albrecht, 1993], p.227; [Schneider, 1996], p. 36; [Bullinger et al., 1997], p. 9). *Organization, People* and *Technology* are at the center of all Knowledge Management activities, supplemented by the corporate culture. If a Knowledge Management Strategy concentrates only on one aspect or pillar (e.g., just implementing an intranet) this Knowledge Management Strategy is likely to fail. The following sections discuss the dimensions of Knowledge Management in more detail.

3.1.1 Corporate Culture

Before the three pillars are described in detail, a clarification of corporate culture is provided. According to Schnyder [Schnyder, 1989], corporate culture is a social-cultural, immaterial, corporate-specific phenomenon. Corporate culture includes the values, norms, social contexts, knowledge and skills which are shared and accepted by most members of the organization.[2] Successful companies have created their own unique corporate cultures, which determine how companies think and behave (cf. [Nonaka & Takeuchi, 1995], p. 42). Probst et al. [Probst et al., 1997a] p. 360, stresses that effective Knowledge Management can only be introduced within a

1. This chapter is based on [Wolf & Decker & Abecker, 1999] and on [Wolf, 1999].

2. "Unternehmenskultur ist ein soziokulturelles, immaterielles unternehmungsspezifisches Phänomen, welches die Werthaltungen, Normen und Orientierungsmuster, das Wissen und die Fähigkeiten sowie die Sinnvermittlungspotentiale umfaßt, die von der Mehrzahl der Organisationsmitglieder geteilt und akzeptiert werden." [Schnyder, 1989], p. 61

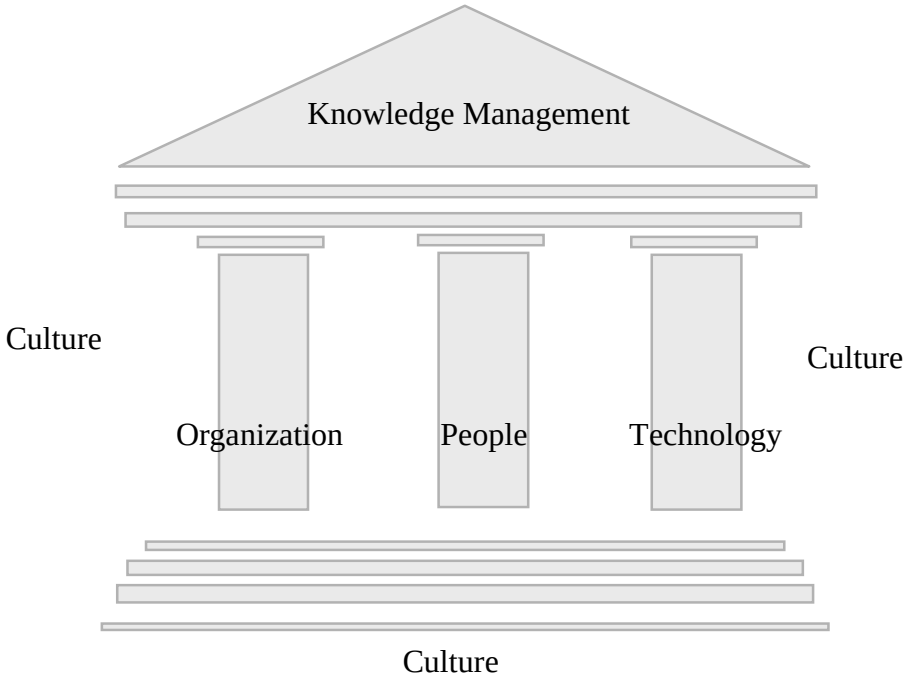


Fig. 3.1. Three Pillars of Knowledge Management

company if this company has an awareness of its own corporate culture, and that culture's influence on the handling of knowledge. As an example [Probst et al., 1997a], p.258 describes that in a highly quantitative-oriented corporate culture only financial figures and similar aspects are regarded as important. Less relevant knowledge areas may not be included in the corporate-wide knowledge dissemination.

One of the most influential factors on corporate culture is the history and environment of the enterprise. This history is documented in the common organizational memory, which represents the grown knowledge expressed through the corporate culture (cf. [Rehäuser & Krcmar, 1996] p. 16). The collective memory is built up over decades or even centuries. Heyl reports in [Hejl, 1991] the following phenomena: even today it is possible for psychologists, using depth interview techniques, to identify the frontiers of the Thirty-Years' War (1618-1648). The fears of that time are so deeply burned into collective memory that even after 350 years they influence the behavior of descendants (example provided by [Romhardt, 1997]). This gives evidence that a corporate culture can be ancient and that effort and time are required to effect shifts.

3.1.2 Organization

The first pillar *Organization* denotes all activities and tasks, in which the processes and structures inside an enterprise are important. To establish a holistic Knowledge Management strategy, structures and processes have to be designed such that it is possible to map tasks, responsibilities and competence within an enterprise in a transparent way. The task of the organization is to establish the connection between business processes and the asset *Knowledge*, which needs to be organized and disseminated (cf. [Bullinger et al., 1997], p. 9f; [ICD, 1997]). To achieve an improvement in productivity, the knowledge intensive processes especially must be supported and improved, e.g.,

Research and Development, Marketing, Design, Education (cf. [Davenport et al., 1996]). Thus, evaluation of these processes becomes more and more important. That means appropriate metrics have to be used that help measure the cost/benefit ratio (cf. [Bullinger et al., 1997], p. 9f]).

Davenport [Davenport, 1996] argues that Knowledge Management has to be incorporated in the organization's structure. Similar to the organizational sections that are responsible for functions such as *finance* and *human resources*, an organizational section is needed that will be responsible for the function of *Knowledge*: "*Knowledge won't be well-managed until some group within a firm has clear responsibility for the job.*" [Davenport, 1996]. As a consequence, Davenport demands the explicit implementation of Knowledge Managers. The tasks of these Knowledge Managers are Knowledge Acquisition, Knowledge Structuring, the introduction of knowledge-oriented information technology and the monitoring and control of knowledge utilization. It is not sufficient to implement Knowledge Management within the job descriptions of certain employees in the IT department, rather new positions must be created to explicitly support Knowledge Management. Knowledge Management must also be supported from above. In the following, some of these new Knowledge Management-related job positions and management roles are introduced, mostly taken from ([Probst et al., 1997a], p. 358ff):

- *Chief Knowledge Officer (CKO) and Knowledge Managers*. The CKO is already an established management role in some companies. Their task is to develop, to organize and to control the organizational knowledge base. They have to sensitize the overall organization. The CKO supervises the Knowledge Manager. The task of the Knowledge Managers is to operationally implement the goals developed in the knowledge strategy by the CKO. Knowledge Managers are often regarded as absolutely necessary to the implementation of a Knowledge Management strategy: "*Knowledge management requires knowledge managers.*" [Davenport, 1996].
- *Knowledge Broker* [Waddington, 1996]. A 350-page report based on a survey of 1,313 junior, middle and senior managers in the U.K., U.S., Australia, Hong Kong and Singapore reported that:
  - two-thirds of managers report tension with work colleagues, and loss of job satisfaction because of stress associated with information overload,[(1)]
  - one-third of managers suffer from ill health as a direct consequence of stress associated with information overload. This figure increases to 43% among senior managers.
  - almost two-thirds (62%) of managers testify that their personal relationships suffer as a direct result of information overload.
  - 43% of managers think important decisions are delayed, and the ability to make decisions is affected, as a result of having too much information, and
  - 44% of managers believe the cost of collating information exceeds its value to business.

As a consequence, new job descriptions emerge:

- *Knowledge Brokers* use new technologies (e.g., Inter- and Intranet) to obtain necessary information. Especially important for Knowledge Brokers is their meta-knowledge: they have to know where to look up necessary knowledge. This enables fast look-up and dissemination within an organization (cf. [Schüppel, 1996], p. 201]).

1. [Kirsh, 2000] examines the notion of 'information overload' in more detail and provides a definition as well as a classification.

- *Competence Field Managers* design, control, and develop an area of competence within an important knowledge area. The task of a Competence Field Manager is to build up a network of expertise within a knowledge field, and to group these experts together. This includes also the establishment of an appropriate infrastructure, containing news groups, conferences, Best-Practice-Workshops etc.
- *Boundary Spanners* establish connections between different areas of competence and detect previously unused knowledge assets. The exploitation of these knowledge assets is delegated to the appropriate Competence Field Manager (depending on the area). Boundary Spanners have international and interdisciplinary contacts and are identified as the people to talk to when issues involving more than one department are identified. Furthermore, they are the appropriate people to build up internal and external contacts.

3.1.3 People

Human beings are at the center of all economic activities. A goal-oriented human resource management is therefore essential, because in the end only human beings can own and use knowledge (cf. [Bullinger et al., 1997], p.9). Thus the knowledge (and the ability to access knowledge) of the employees constitutes the organizational knowledge base. If employees are not involved in Knowledge Management activities, if they are not motivated and their needs are not respected, then successful Knowledge Management is not possible.

To deploy Knowledge Management inside a company the stage must be set for knowledge sharing among employees, departments and domains. Also, an awareness must exist of how important an effective and efficient usage of the production factor *Knowledge* is - for the individual as well as for the company's success (cf. [ICD, 1997]). This is in strong contrast to statements like '*knowledge is power'* and '*who shares his knowledge, loses power and can be replaced'*. To counteract these opinions is a difficult task for leading managers. They have to convince the knowledge owners of the value and usefulness of their knowledge to the collaborators and that the knowledge is only useful if it is shared with others. The task is to develop experts in knowledge brokering, whose customers are the users of their knowledge, and to offer incentives for knowledge sharing and usage. To establish a system that supports knowledge sharing as part of the corporate culture, knowledge sharing must be rewarded with more than just knowledge owning. In the beginning of the establishment of a Knowledge Management strategy seeds have to be planted, e.g., by charismatic leaders, which convey Knowledge Management as something positive (cf. [Davenport, 1996]).

During the change from an industrial society to a knowledge society, new job areas are being created. Knowledge-based companies such as consulting and software companies are growing fast. Therefore, there is a huge change in working methods and content: knowledge-based companies employ knowledge workers. Typical representatives are e.g., consultants, software developers, lawyers, employees in research and development. Drucker [Drucker, 1988] detected this trend in 1988: "*The center of gravity in employment is moving fast from manual and clerical workers to knowledge workers who resist the Command-and-Control model that business took from the military 100 years ago.*" The break with the Command-and-Control model is realized by the implementation of new forms of organizations, usually accompanied with reengineering efforts: flat organizational hierarchies or a shift of competence from management to the knowledge worker, who is better able than management to make decisions because of his or her expertise, is required. This shows the dependency between workers and organizations in a holistic approach to Knowledge Management (cf. [Karner, 1996], p. 95).

3.1.4 Technology

Technology means information technology as well as communication technology and similar technology areas. The relationship to the previous section is illustrated by Tom Davenport [Davenport, 1996]: "*Effective Management of knowledge requires hybrid solutions of people and technology.*" Some tasks are better done by human beings, others are better done by technology. Using knowledge to make a decision is usually better effected by humans. Transformation and storage of information is usually better effected by an appropriate technology. However, looking up useful knowledge using technology still requires a huge effort by human beings if the knowledge is not formalized and is only represented in documents. But formalization of knowledge is also a costly process that requires skilled workers and a huge effort.

3.2 Elements of Knowledge Management II

In the following section, the 8 building blocks of Knowledge Management according to [Probst et al., 1997a] are described. Figure 3.2 depicts two areas: the outer areas (*Knowledge Goals, Knowledge Strategy Implementation, Knowledge Assessment*) are related to the classical managerial cycle. The inner area (comprising a network) consists of 6 building blocks: *Knowledge Identification, Knowledge Acquisition, Knowledge Development, Knowledge Sharing/ Dissemination, Knowledge Preservation, and Knowledge Utilization.* This dissection of the Knowledge Management process has several advantages: the complex process *Knowledge Management* itself becomes more manageable; the focus on single process steps allows division of responsibilities; and the structuring of the management process allows for easier assessment of the overall process; detection of problems in this process is more focused and detection of problems which interfere with the overall Knowledge Management process is simplified. The following sections describe the single building blocks in more detail according to [Probst et al., 1997a].

3.2.1 Defining Knowledge Goals

The definition of knowledge goals aims at guiding the whole Knowledge Management process. The other process steps should support the knowledge goals defined in this step. Depending on the level of detail, the following types of knowledge goals may be distinguished.

- *Normative Knowledge Goals* aim at the creation of a framework (e.g., at the creation of corporate culture that supports Knowledge Management).
- *Strategic Knowledge Goals* define the future knowledge demands of the company, by defining core knowledge areas.
- *Operative Knowledge Goals* are defined by addressing the normative and strategic knowledge goals at an operative level.

3.2.2 Knowledge Identification

During this process step an analysis of the knowledge available in a company is performed. Know-how, knowledge provider, information systems and personal networks need to be identified and to be made transparent, so that the knowledge can be accessed when needed. Often the problem is not

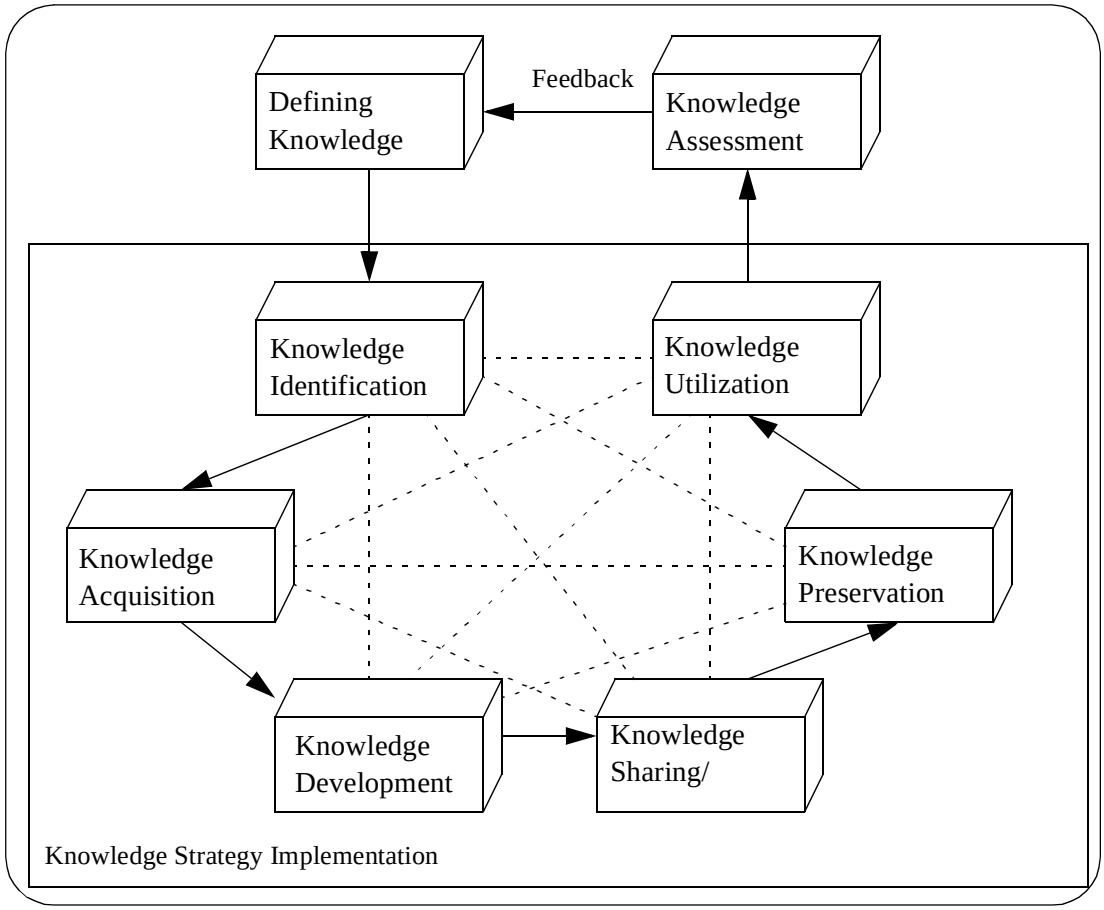


Fig. 3.2. Building Blocks of the Knowledge Management Process (taken from [Probst et al., 1997a])

so much the lack of information sources, but that information sources are not structured appropriately to obtain the relevant knowledge in time. This is made clear by complaints such as, '*if we knew, what we know*' (cf. [Davenport, 1997]).

For the localization of knowledge inside an organization, several tools may be used: popular tools are *knowledge maps*, which are graphical dictionaries of knowledge providers, knowledge sources, or applications (cf. [Eppler, 1997]). A non-graphical version of knowledge maps is *Yellow Pages for Expertise*, which helps to locate an appropriate expert in an enterprise (cf. [Probst et al., 1997a], p. 107). Knowledge providers, together with their domains of expertise, are collected in a dictionary which may be searched by electronic means.

However, knowledge maps and yellow pages help to find a knowledgeable expert inside an organization. It is desirable to have the knowledge itself in a machine-processable and deployable form (if experts leave or retire). Technology known from Knowledge Engineering (see Section 2.2.1) can be used to formalize this knowledge (cf. [Wielinga et al., 1997], [Milton et al., 1999]).

3.2.3 Knowledge Acquisition

Since Knowledge Acquisition within an organization deals first and foremost with knowledge acquired by human beings, the learning behavior of humans is investigated first. Human beings learn through the sense organs.

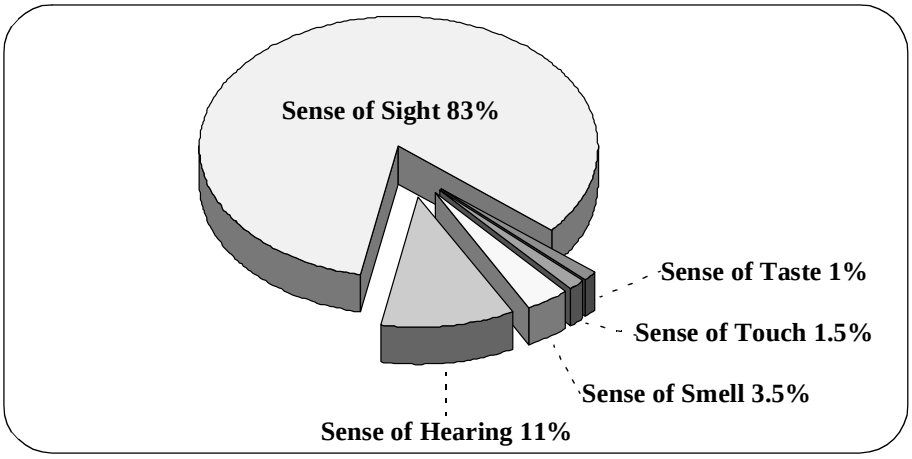


Fig. 3.3. Knowledge Acquisition with the Senses
(see [Böning, 1994])

Figure 3.3 illustrates the percentage for sense involved in the human knowledge acquisition, however, it doesn't say anything about long-term learning. There are numerous educational possibilities for acquiring knowledge, with different effectiveness regarding long-term learning.

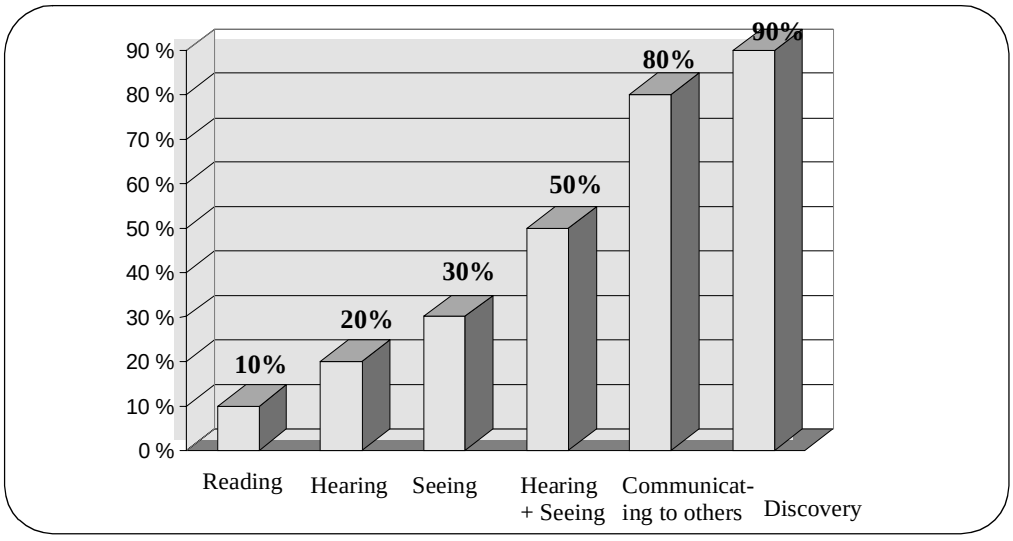


Fig. 3.4. Memorization Effect for Different Knowledge Acquisition Types
(taken from [Böning, 1994])

Figure 3.4 from [Böning, 1994] roughly depicts the effectiveness of selected learning possibilities. E.g., workshops can be classified as very effective learning events; classic lectures (*Hearing + Seeing*) are in the middle.

Another possibility to acquire knowledge is to purchase it - enterprises may buy required knowledge in external markets. The following sources can be distinguished:

1. *Single, external knowledge sources*. Examples for taping single knowledge sources are recruiting methods or hiring of external consultants.

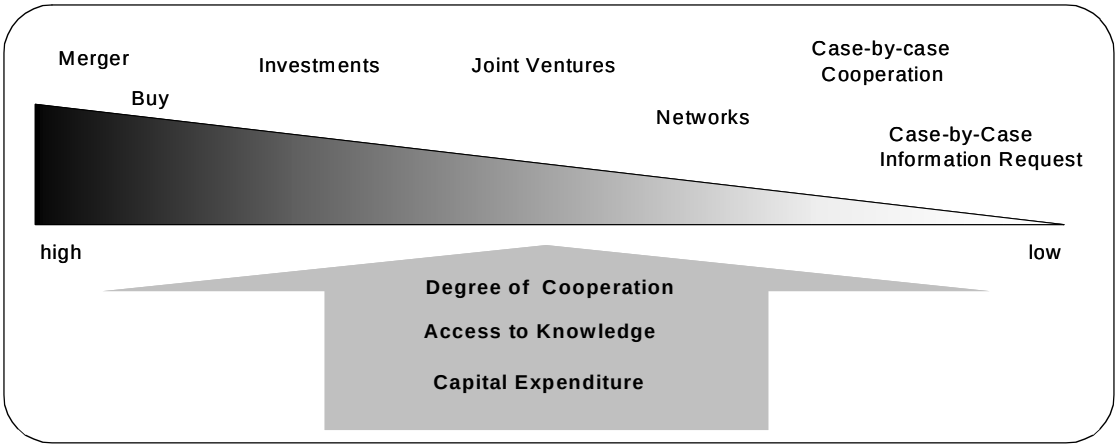


Fig. 3.5. Continuum of Cooperation
(see [Probst et al, 1997b]

2. *Knowledge of other organizations*. The usage of the knowledge bases of other companies helps to acquire competencies faster. Fig. 3.5. [Probst et al, 1997b] shows different possibilities for cooperation and their influence on the degree of cooperation, access to knowledge and necessary capital expenditure.(1)
3. *Knowledge of stakeholders*. According to [Bleicher, 1992] and [Probst et al., 1997a] stakeholders (groups with particular interests in an organization) can be useful knowledge sources. The knowledge of customers, suppliers and shareholders can be beneficial in acquiring new ideas and suggestions for improvement.
4. *Knowledge Products.* By buying *knowledge products* like patents, CD-ROMs and software an enterprise does not automatically acquire new knowledge benefits. The interaction between employees and these knowledge products generates new usable knowledge. Software has a high potential as a knowledge medium, since by using software complete business processes are often implemented.

The most important success factor is integration of the new knowledge into the existing knowledge base. New, acquired knowledge is often not accepted, however, and learning barriers appear [Schüppel, 1996]. Common also is the so-called '*Not-Invented-Here-Syndrome*' - if it was not developed inside the company, it is not accepted.

3.2.4 Knowledge Development

Elimination of identified knowledge gaps can either be done by knowledge acquisition (as described in the previous section) or by *knowledge development*. Knowledge development requires goal-oriented management, which enables the organization to produce knowledge which is not available on site or does not even exist. Since the organizational knowledge base has an individual and a collective dimension, this division is also inherent in the knowledge development procedures.

1. Only the case of voluntary or legal cooperation is discussed. Other possible forms may be e.g., espionage.

- Individual knowledge development is based on two components: *creativity* and the ability to systematically *solve* problems. However, a large part of the personal knowledge is neither obtained by innovation nor from explicit problem-solving techniques, but due to implicit abilities. Implicit, tacit knowledge can often be modeled and by this modeling new (explicit) knowledge is created.
- From the discussed individual development procedures, the collective differs in the sense that it is not the individual, but rather a team that is the focus of learning. This usually requires the creation of complementary abilities in the team as well as the definition of group targets. Openness and trust are at the forefront in particular, paired with healthy group dynamics, in building superiority of the collective knowledge development over the individual. An instrument particularly applicable for the improvement of work in projects is *lessons learned* [van Heijst et al., 1998]. After the termination of a project the team reflects on the project. The reflection makes explicit which critical experiences were gained and which experiences are useful for future teams with similar tasks. Furthermore, the reflection also helps team members to learn about their own performance. Lessons learned thus represents the essence of a collective learning process and can be a crucial factor for work improvement if represented in an accessible (usually electronic) form.

3.2.5 Knowledge Sharing and Dissemination

Nobody needs to (or is able to) know everything - it is usually sufficient for employees to have fast and simple access to the knowledge needed and that the knowledge distributions mechanisms work. Not only is the distribution process of knowledge to the place of work important in this context, but so also is a willingness to share knowledge with others. Knowledge sharing can be supported by different tools, which do not necessarily have an electronic nature.

Indispensable for a productive team is that its members share their knowledge; the individual is not only evaluated for his/her abilities in problem solving, but for their contributions to the team and its capabilities regarding cooperation. Virtual teams (and organizations) help to overcome the locality problem of skills being only available at a specific location. Skills are available world-wide, and knowledge sharing is usually handled by electronic means.

In teams where the members are located close to each other, knowledge dissemination can be simplified by appropriate *space management*. Usually office organization reflects the department or group organization. Space management promotes the organization of work places along the business process - the organization of the office locations is oriented towards the business process. By the relative proximity of the co-workers, who depend on each other, knowledge exchange is simplified [Probst et al, 1997b], p. 226.

To distribute knowledge effectively and efficiently requires an infrastructure of personnel and electronic networks. Two different approaches may be distinguished: the *pull* and the *push* strategies. With the *push* strategy a central place delivers the appropriate knowledge to the individual co-workers. The central place decides which multiplication concepts (e.g., training, learning-by-doing or e-learning) are used. The pull approach requires the employee to be motivated to procure the required knowledge by him- or herself, usually supported by the means to rapidly access relevant information. Computer networks and systems especially offer many possibilities to distribute data, information and knowledge. The construction of employee networks for certain topics helps to use and distribute required knowledge and to react quickly to new trends in the marketplace.

3.2.6 Knowledge Preservation

To reuse past experiences in the future and to generate new knowledge, a carefully established and goal-oriented knowledge preservation process is necessary. The knowledge preservation process has three steps [Probst et al, 1997b], p.289.

1. *Selection*. Large amounts of data, information and knowledge are generated every day, and many employees suffer from information overload. The crucial point is the selection and separation of the 'keep-worthy' knowledge. This selection step is a large challenge for an enterprise. The selection must be based on knowledge targets, which help to identify core areas in which information is stored.
2. *Storage*. Three different levels of storage are distinguished: a) the individual, b) the collective (or group), and c) the electronic or physical level. At the individual level human beings are the knowledge carriers, whose know-how must be retained. Death, retirement and changes to other enterprises create 'knowledge flow-off'. Through incentive schemes, know-how carriers can be held within the enterprise. Knowledge can be distributed redundantly to several persons by job rotation. The collective level covers the storage of experiences, which were gained in the group and knowledge, which was received through the procedural processes of the organization. Individual persons only store fragments of these collective experiences. The safest form of storage of data, information and knowledge is electronically in computer systems or on paper. However, it is very difficult to capture implicit knowledge and store it in electronic form. Also, indexing of knowledge is an important problem: stored files are of no use to anybody unless they can be readily located. Electronic storage usually offers a huge advantage here.
3. *Update*. Keeping knowledge up-to-date is a continuous process. Information systems containing old, invalid information are not useful but equally knowledge on the individual level needs to be updated. [Schüppel, 1996] defines the half-life period of knowledge as the time length in which the relevance of knowledge has shrunk to 50%. In general, the half-life for a knowledge period is dependent on the type of knowledge (e.g., knowledge about physics is valid much longer than knowledge about a particular software product).

Since much knowledge becomes increasingly less relevant over time, directed forgetting is necessary. According to Karner [Karner, 1996], p.116, learning is easier than dropping old habits and opinions, especially if they (as is often the case in Europe) are based on deep history.

3.2.7 Knowledge Utilization

Knowledge Utilization is a process for transferring knowledge into deployment. [Gueldenberg, 1998] distinguishes Knowledge Utilization from *Knowledge Realization*, which can either be direct (e.g., by acquiring the knowledge itself through such avenues as licenses or training) or indirect (acquiring Knowledge through products or services).

According to [Gueldenberg, 1998], knowledge is utilized in three different ways: *communication*, *decision*, and *action.*

New knowledge causes changes in communication behavior, e.g., in changed communication processes.

Knowledge utilization in actions is expressed e.g., in attitude change of the organization itself, usually brought about by individual learning effects. However, sometimes the organization as a whole learns, which brings about Knowledge Utilization in ways not directly reducible to a single individual.

Knowledge utilization involving decisions is the most basic knowledge utilization method - knowledge utilization involving communication and action usually requires a decision to be taken.

[Probst et al, 1997b] stress that Knowledge Utilization may be disabled by utilization barriers: e.g., mistrust of foreign knowledge or fear of embarrassment can hamper successful knowledge utilization.

3.2.8 Knowledge Assessment

Finally, *Knowledge Assessment* is the last ingredient in Probst's model necessary to close the management cycle depicted in Figure 3.2. Difficulties exist in measuring the performance gain by Knowledge Management. [Probst et al, 1997b] cites several companies that try to measure the performance gain caused by the introduction of Knowledge Management. However, [Probst et al, 1997b] stresses that there is no exact way to measure the gain, but still hint that knowledge assessment should be the foundation of knowledge controlling. Starting with a task definition and separating them in different parts, effects become measurable.

3.3 A Synthesis: The Knowledge Management Cube

The previous sections presented two models for classifying and structuring Knowledge Management. The three-pillar model (Section 3.1) classifies Knowledge Management according to the dimensions *Organization*, *People*, and *Technology*, which also represent the three main components of companies. The components, according to Probst, focus on the Knowledge Management process with the single elements as processes. The elements contain organizational, people-oriented and technological aspects. The combination of both models results in a two dimensional intermediary model. Every management activity involves generic, strategic and operative components, which impact on the Organization, People and Technology, as well as the steps in the Knowledge Management process chain. Thus, these steps are the third dimension of a complete Knowledge Management.

The synthesized model, the Knowledge Management Cube (KMC), is the result of the combination of its subparts and is illustrated in Figure 3.6. Every project aiming at Knowledge Management can be captured, analyzed and assessed with the KMC model. The KMC model is illustrated with an example: the implementation of an intranet feature which visualizes all employees of a department and their skills in a tree structure. The data is maintained by the employees themselves. In this case, all three dimensions of the KMC are required to describe the project. The project involves the structure of the organization and the employees, and requires a certain technology. The knowledge identification process step is particularly important, but *Knowledge Goals, Knowledge Assessment* and *Knowledge Preservation* are also required process steps. The project has to be integrated into the generic and strategic goals of the organization, and the operative management (from the determination of project goals to operative management aspects) determines the project's success or failure.

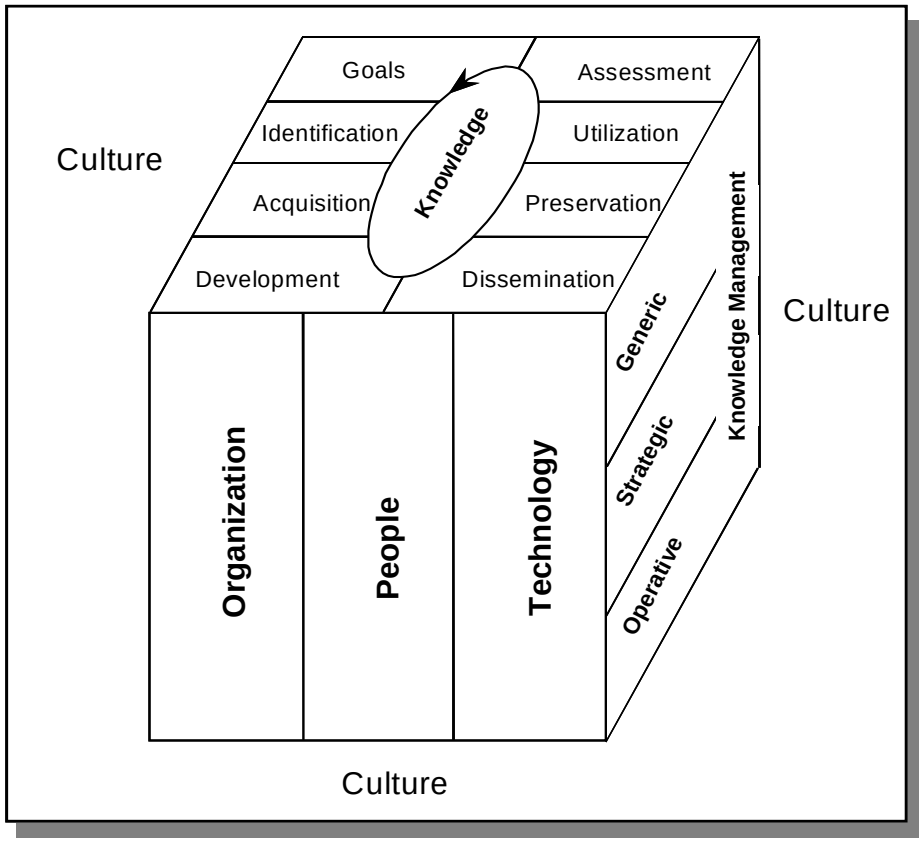


Fig. 3.6. The Knowledge Management Cube

The KMC model is used in this way to define all facets of Knowledge Management projects, and to define interfaces between different subparts of the projects. This helps to manage the complexity of introducing and building Knowledge Management systems within an organization. The following chapters concentrate on the information technologies part of the KMC model.(1) The KMC model is used in Section 10.1.1 to analyze the Ontobroker system, and to determine the different aspects that were involved in the $(KA)^2$ project, where Ontobroker was used as a enabling technology for Knowledge Management.

1. It turned out that companies have already adopted the KMC model in practice since its first publication - see e.g., http://www.pass-consulting.com/internet/html_d/presseservice/pdf_499/bestandteile_wmg.pdf (accessed December 2002).

Chapter 4 Organizational Memory Information Systems[1]

This chapter takes a closer look at IT support for Enterprise Knowledge Management (KM) and the notion of Organizational Memory Information Systems (OMIS), thus focusing on the technology aspects of the Knowledge Management Cube. The chapter provides several classifications of OMIS (e.g., a process-oriented vs. a product-oriented viewpoint). OMIS are contrasted to Knowledge Based Systems.

4.1 IT-Support for Knowledge Management

Managers have recognized that, for a number of reasons, effective development and management of an enterprise's organizational knowledge base will be a crucial success factor in the knowledge-intensive markets of the next century. Identification, acquisition, development, dissemination, utilization, and preservation of knowledge in the enterprise have been identified as basic KM activities (cf. [Davenport et al., 1996], [Probst et al., 1997a]).

IT support for KM is addressed in e.g., [Borghoff & Pareschi, 1998a], [Liebowitz & Wilcox, 1997], [Wolf & Reimer, 1996]. However, all such comprehensive volumes are characterized by an enormous heterogeneity of goals and techniques, which makes it almost impossible to identify a common technical core and philosophy. Even worse, in many articles it is not possible to see why the achievements described should be called “Knowledge Management” although exactly the same research topics were categorized as information systems, expert systems, CBR technology, CSCW systems, workflow engines, data warehouses, data mining, document management, etc., a short time ago. [Kühn & Abecker, 1998] proposed distinguishing two basic categories of IT contributions according to their main focus and approach:

- The *process-centered* view mainly understands KM as a social communication process, which can be improved though IT-means by various aspects of groupware systems. It is based on the observation that the most important knowledge source in an enterprise is the employees. Furthermore, solving really *wicked problems* [Conklin & Weil, 1997] is merely a process of achieving social commitment than one of problem solving. Basic techniques for this approach come from Computer-Supported Cooperative Work (CSCW) and from Workflow Management [Prinz & Syri et al., 1998], [Simone & Divitini, 1998].
- The *product-centered view* focuses on knowledge documents; their creation, storage, and reuse in computer-based *corporate memories*. The view is based on the explication, documentation, and formalization of knowledge as possession of a tangible resource. The user's individual knowledge development and usage shall be supported by presenting the right knowledge sources at the appropriate time. Basic techniques come from Document Management, Knowledge-Based Systems, and Information Systems [Kühn & Abecker, 1998], [van Heijst et al., 1998].

1. This chapter is based on [Abecker & Decker, 1999] and [Abecker, Decker & Kühn, 1998b]

The following identifies where the diversity of technical approaches originates. This results in the observation that there is no single goal and approach to Organizational Memory, but merely a bundle of motivations and technical aims. Requirements are identified, which should nevertheless (in the ideal case) be fulfilled by *Organizational Memory Information Systems (OMIS)*.

AI and especially expert systems have an important role for processing knowledge in Computer Science. Expert Systems, however, exhibit serious deficiencies in industrial practice. Analyzing the problems of expert systems has led to the development of OMIS approaches: instead of *fully* automating *specific* tasks in an enterprise (like diagnosis or configuration) by *completely new* IT *approaches* (problem-solving methods and formal knowledge-bases), the goals are now more focused on supporting the user in performing *arbitrary* processes leveraging and exploiting the knowledge already contained in manifold representations in the enterprise.

However, to realize such support the given representations are enhanced incrementally for further applications. And here, it turned out that the numerous AI techniques (ontologies, inference systems, CBR, etc.) are entirely applicable. The integration of product-centered and process-centered systems is viewed here as the abstract goal to be addressed by OMIS research and development. The following sections deal with:

- arguments that this diversity of approaches is not a problem for IT people, but is natural if one considers the diversity of reasons and aims of KM from the business perspective.
- arguments that the two above approaches are not contradictory, but complementary. Thus, a useful OMIS will emerge from a synergistic amalgamation of both approaches.
- essential elements of an OMIS definition and necessary conditions to develop a comprehensive technological approach.

4.2 Crucial Points for Realizing OMIS

Consideration is concentrated here on technical aspects of OMIS. Of course, for building and deployment of operational systems in practice, the non-technical aspects are specifically of crucial importance. For a thorough system-theoretic analysis of functions and properties of an OMIS, please refer to [Stein & Zwass, 1995] and [Dieng et al., 1998].

As a working definition of what an OMIS constitutes, this thesis proposes the following:

> *"An Organizational Memory Information System (OMIS) integrates basic techniques into a computer system, which - within the enterprise's business activities - continuously gathers, actualizes, and structures knowledge and information and provides it in different operative tasks in a context-sensitive, purposeful and active manner in order to improve cooperative, knowledge-intensive work processes."*

Figure 4.1 gives an impression of which existing enabling technologies may be used for building OMIS. The three steps given in Figure 4.1 are related to the the KMC model dimensions. Based on industrial studies (e.g., [Kühn & Abecker, 1998], [Tschaitschian et al., 1997]) it is possible to identify requirements on each of the three levels of any information or knowledge system (*capture*, *storage*, and *usage*). These conditions help to establish specific requirements for designing and building an OMIS.

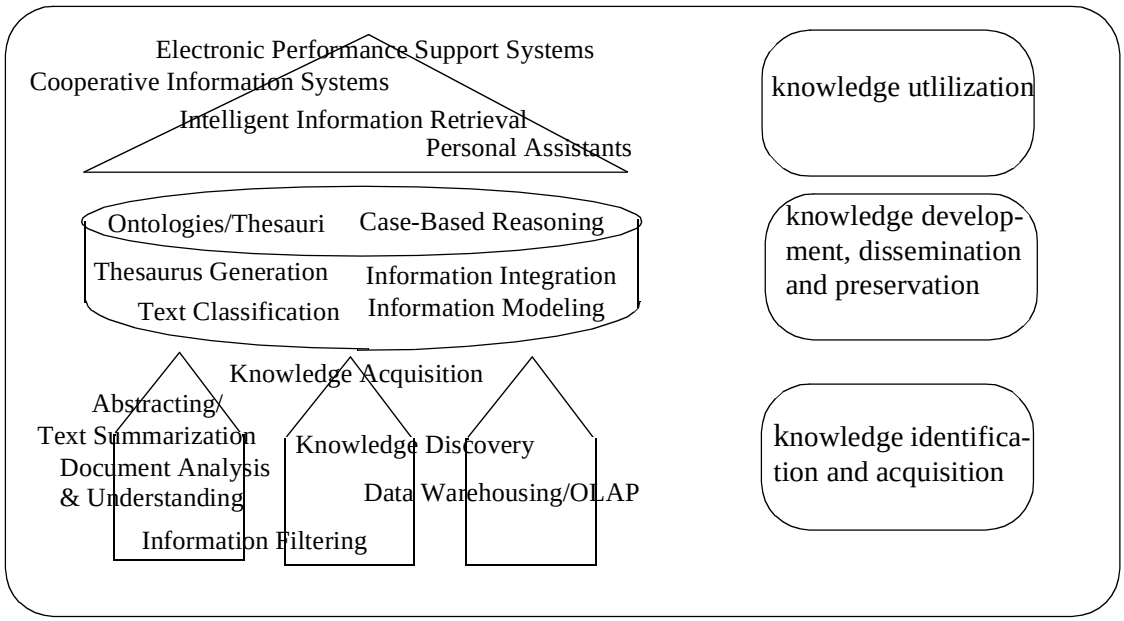


Fig. 4.1. Basic Technologies for Building an OMIS
(taken from [Abecker et al., 1998c])

All three levels must be able to cope with a huge variety of possible knowledge sources and formats, ranging from databases, to text or multimedia documents, to some formal knowledge representation. Especially for the knowledge repository at the core of the OMIS, coping with technical as well as conceptual heterogeneity is a demanding task. The following paragraph will discuss the three levels in more detail.

4.2.1 Knowledge Acquisition and Maintenance

Knowledge acquisition and maintenance are as crucial for the successful deployment of an OMIS as for any knowledge-based system (see [Davenport et al., 1996]). In an ideal case, an OMIS would be self-adaptive and self-organizing, gathering information within the usual business operations for later reuse without disturbing the normal flow of employee work. The Ontobroker approach presented in part II of this thesis deploys a distributed, non-centralized approach to knowledge acquisition.

In many best-practice projects in consulting companies [O'Leary, 1998b], as well as in the area of *Experience Factories* for learning software organizations [Basili et al., 1994], [Basili & Rombach., 1998], the role of a Knowledge- or Experience Manager has been proposed. Such a Knowledge Manager acts as a kind of knowledge clearinghouse and is responsible for preparing knowledge for reuse, bringing it into a useful form and adding appropriate metadata to ease later reuse. Though this makes sense for rather complex and valuable pieces of enterprise knowledge (as best practices are), it is quite an expensive strategy, especially in large and diverse organizations. Furthermore, it is not appropriate for many less complex, less important, and less ambitious knowledge acquisition and maintenance tasks. It is also not appropriate when confronted with huge amounts of data and information to be scanned and analyzed for useful knowledge. [Wiederhold, 1992] argues from a mediator perspective that a single center cannot deal with all the varieties of information that are

useful for corporate decision-making. Thus, Knowledge Acquisition would best take place in a distributed manner. The Ontobroker system, the main contribution of this thesis and presented in part III, performs Knowledge Acquisition in that way.

In such cases, acquisition tools and automated approaches are helpful. Consequently, the knowledge capture level in Figure 4.1 shows several input streams, which feed the OMIS with knowledge extracted from texts and from databases.

[O'Leary, 1998a] describes examples of specialized analysis tools and knowledge discovery agents constantly searching the enterprise internal and external web and knowledge sources in order to feed interesting input into the OMIS.

Another necessary part for knowledge acquisition in OMIS is a user-friendly system interface, which collects user feedback, comments, and memos in a context-sensitive and unobtrusive manner. Although there have been interesting results in document analysis and understanding (DAU) for more than a decade (see, e.g., [Dengel & Hinkelmann, 1996]), the application scenarios for methods such as document structure analysis, information extraction, and automated summarization are still quite limited. The Price Waterhouse systems show that already simple text analysis technology can produce useful results. Steps towards the analysis of Web pages have already been done (see, e.g., [Catarci et al., 1998]) and are currently known as "*Wrapper generation*" (cf. [Wiederhold & Genesereth, 1997]) or *Wrapper Learning* [Kushmerick et al., 1997].

Ontologies (as introduced in Section 2.2.2) are proposed as a means for retrieving and structuring knowledge. They provide: "*an explicit specification of a conceptualization,*" [Gruber, 1993] and are discussed in the literature as a means to support knowledge sharing and reuse [Farquhar et al., 1997]; [Fridman Noy & Hafner, 1997].

[Abecker et al., 1998c] describe three kinds of ontologies, which are useful for organizational memory information systems.

These three ontologies are:

- *Information ontology*: the information ontology describes the information meta-model, e.g., the structure, and format of the knowledge sources. This is the lowest level ontology.
- *Domain ontology:* the domain ontology is used to describe the content of the knowledge source.
- *Enterprise ontology:* the enterprise ontology is used in modeling business process. Its purpose is to model the knowledge needs in business processes to describe a process context, which enables active knowledge delivery.

Knowledge acquisition tools for Knowledge Management have to:

- support the creation and maintenance of these types of ontologies (especially the domain ontology), and
- support usage and structuring of knowledge according to these ontologies (especially the information and enterprise ontology).

Special tools often use just one type of ontology, mostly the domain ontology, because enterprise demands are frequently neglected and tools are commonly focused on one special input format. Also, knowledge acquisition tools are usually tailored for building and acquiring knowledge for AI applications. However, approaches to knowledge in AI often make the assumption that knowledge is completely formalizable. In a corporation this is usually not the case. As [Essers & Schreinemakers, 1996] state:

> *"An important difference between the approaches in philosophy and CKM (= Corporate Knowledge Management) to knowledge in this respect is that the responsibility of CKM cannot be restricted to knowledge that is of an incontestably scientific nature but must extend to all alleged knowledge a company accepts consciously or unconsciously as a capacity for corporate action."*

For the full formalization of knowledge, complex representation formalisms are necessary. This would lead to weak competencies and high maintenance and building costs for knowledge-based systems. For this reason many Knowledge Management approaches are document-centered, representing knowledge informally as texts or drawings.

4.2.2 Knowledge Integration

The core of an OMIS is a (possibly distributed) repository system, which must - technically viewed - grow out of the integration of several legacy systems (which usually cannot be replaced), enriched by information and metadata that allows retrieval of useful knowledge and documents in a given application situation. This repository is characterized by heterogeneity with respect to several dimensions.

Concerning *meaning and content* of knowledge, categories such as product and process related knowledge, reasons for decisions, individual competencies and skills, etc., can be found in an enterprise. Contemporary systems usually manage just one kind of this knowledge. Thus, the issue of *information integration* and *semi-structured data* as a means to capture and represent all the different kinds of information available in an organization becomes important.

Given a dynamic environment, the concept of *Modularized Mediation* becomes critical. [Wiederhold, 1992] remarks that a single center or mediator cannot deal with all the varieties of information that are useful for corporate decision-making. The consequence is that an OMIS will finally be distributed, possibly implemented as a dynamic set of mediators.

The power of KM comes exactly from the interplay and synergistic view of multiple kinds of knowledge in their usage and creation context. Looking at representation, text and hypertext documents, e-mails, graphics etc. are often already at the workplace and are much more comfortable and expressive for the human user than any formal knowledge representation. On the other hand, only formalized notions allow for inferences and precise retrieval. So it is beneficial to aim at a meaningful combination of formal and informal knowledge items. A good example of the beneficial synergy between several kinds of knowledge are *Issue-based Information Systems* (IBIS) [Kunz & Rittel, 1970]. The QuestMap tool [Buckingham Shum, 1998], an IBIS, enables embedding of design artifacts (like graphics, minutes of meetings, or requirement texts) into a graphical notation of the discussion and decision process of the design group which led to these artifacts. So, one can retrieve all pros and cons encountered in a discussion, or can trace all decisions influenced by some preconditions in the case that the preconditions change.

Also in the design area, additional, complementary views for business process modeling (BPM) were tested for their efficacy in analyzing and controlling document and knowledge creation, utilization, and flow. Extending *Business Process Modeling* (BPM) toward representing knowledge, documents, and expertise seems to be a promising approach. Chapter 5 shows the results of extending BPR and applying it to a case study involving industrial design.

4.2.3 Knowledge Retrieval

From the point of view of knowledge utilization, two features distinguish an OMIS from conventional information and knowledge systems:

(1) the active, context dependent supply of knowledge and information useful for the user in a given situation, and (2) the application independence of the OMIS knowledge base, which should be useful for manifold task-specific assistant systems. To this end, task specific OMIS exploitation and retrieval services can act as intelligent assistants to the user that:

- accompany the execution of tasks, and
- present relevant information that helps employees to do their work effectively.

Consequently, such an intelligent assistant must have some notion of the work process it supports and must be able to perform a precise content retrieval from the OMIS archive. First steps in this direction are user, group, or task specific knowledge push mechanisms, constantly filtering and actively delivering interesting OMIS contents to their subscribers. [Staab & Schnurr, 2000] present an approach, based on the Ontobroker toolset, to retrieve information based on ontologies associated with tasks in a business process.

Activeness and task independence are somewhat conflicting goals. Highly ‘intelligent’ approaches to active support require so much domain and task knowledge that they can hardly be built in an application-independent manner. [Kühn & Abecker, 1998] describe an early OMIS prototype, the KONUS system for conservation of crankshaft design rules. KONUS already actively suggests design decisions or generates a warning if the user violates stored rules. However, these services require a completely task and domain-specific architecture. Moreover, the services are based on a fairly expensive knowledge acquisition and modeling process.

In [Reimer, 1997], the EULE/2 prototype is presented, which supports employees at Swiss Life Corporation on the basis of a declarative, knowledge-rich formalization of business processes, business rules, and relevant laws and regulations. These formal models are employed for controlling workflows, for automated data gathering for open cases, and for semi-automatic consistency-checks for user decisions with relevant business rules and laws. In the case of violations, the system can actively warn the user and offer the relevant pieces of rules and legal texts. Though it is a specific business solution for Swiss Life Corporation, EULE/2 made an important contribution: it introduces some notion of business process modeling for controlling the KM process.

The DFKI KnowMore project [Abecker et al., 1998c] goes a step further and proposes conventional business process models (BPMs) as a first step to KM solutions. The process of building and analyzing BPMs is used as a vehicle to organize enterprise KM in the same stream of work. Knowledge-intensive tasks are identified as integral parts of business processes. Standard BPMs are extended by variables, which carry decisions from previous business activities, thus defining the kind of context and input for subsequent knowledge-intensive tasks. Workflow enactment controls the overall business process and thus gives the user a chance to activate specific task assistants as a hook to the conventional workflow engine. If a particular knowledge item is created within such a business process, the actual workflow context already defines its creation context, which might be automatically assessed for its later reuse in other business processes.

Cased Based Reasoning (CBR) can provide robust and effective techniques for retrieval. Metadata is added to knowledge resources. Queries are processed employing an appropriate measure, which estimates the similarity between query situation and stored experiences. An *experience-factory* approach along these ideas is described in [Althoff et al., 1998]. The advantage is the possibility of

incremental formalization: at first, cases can be described very roughly; more sophisticated meta data can be added if required by the application. Retrieval is based on an ontology for formulation of resource metadata.

The above-mentioned workflow-embedded activation model for knowledge retrieval agents is still in its very first stage. Agent-based approaches for exploiting the OMIS content seem definitely appropriate in order to tackle the manifold integration problems in a scalable manner. Since these agents have to integrate various, heterogeneous information sources, they are similar to wrapper/ mediator architectures as proposed by Wiederhold and others in the database area (cf. [Wiederhold, 1992], [Wiederhold & Genesereth, 1997]).

4.2.4 Differences between OMIS and other Information Systems

Heterogeneity of knowledge and data sources with a predominance of informal representations, and task-independence of the architecture are the main challenges for Organizational Memories. Ontobroker, as detailed in part II, primarily addresses the above challenges in the areas of intelligent assistants exploiting the OMIS repository. A major difference is also that OMIS also require knowledge-intensive, contextually-enriched information modeling at the core of the repository system, and learning and self-adaptive knowledge discovery agents for capturing knowledge and filling the archive.

Innovative approaches often grow out from the integration of known technologies at the borders of different technologies and dimensions, e.g., the Ontobroker system amalgamates formal and informal representations of data. [Borghoff & Pareschi, 1998b] observe that most approaches to KM concentrate either on storage and retrieval of documents (this is what this chapter introduced as the *product view* on KM), or on collaborative support for people *(the process view*). Next generation systems will come from the integration of both views via some knowledge-flow mechanism on the basis of some knowledge cartography. This view is consistent with [Nonaka & Takeuchi, 1995] analysis of kinds of enterprise knowledge and knowledge transformation processes; it is also reflected by O'Leary, whose analysis argued that KM essentially amounts to making connections [O'Leary, 1998c]: connecting people, connecting knowledge and people. Another point arises when building an enterprise-wide OMIS from several previously independent archives: since it is not realistic to expect an enterprise-wide, consistent organization schema and terminology, the questions of ontology mapping and ontology merging for organizing the overall system are crucial [Noy & Musen, 1999], [McGuinness et al., 2000], which also results in a set of *mediators*.

Since formal models are only built for stable and important knowledge (e.g., for the organization of an archive), and are usually very expensive to generate, most of the information inside an OMIS consists of informally represented knowledge in the form of documents. Interpretation of this informally represented knowledge is done by humans. As browsing a document collection is a time consuming activity, it is desirable to support the document delivery and retrieval task with task-oriented assistant systems.

To build task oriented assistant systems, it is necessary to develop techniques for adding information to documents to enable an intelligent information retrieval. These techniques require the usage and elicitation of formal knowledge. The elicitation of formal knowledge is one of the main tasks dealt with in Knowledge Engineering, which is mainly concerned with building Expert Systems. In the next section an investigation of the relationships between OMIS, Expert Systems and Knowledge engineering is undertaken.

4.2.5 OMIS and Expert Systems

Although OMIS and Expert Systems (or Knowledge Based Systems (KBS)) are aimed at the preservation and usage of knowledge, the following differences between an OMIS and a KBS exist: A KBS focuses on the solution to a single task, usually automated by a Problem Solving Method (see Section 2.2.3). This is not true for an OMIS: it supports at least a collection of different business processes and thus has to support different tasks. A KBS contains knowledge at a high level of formalization, whereas an OMIS consists of knowledge at different formalization levels. Examples include unstructured documents, semi-structured hypertext (cf. [Euzenat, 1996]), and formal knowledge bases. Typically, informally represented knowledge is usually much more important than formally represented knowledge inside an OMIS, and formalizing the knowledge represented informally is infeasible and too expensive. In Knowledge Management the final consumers of the knowledge are usually human beings, not the system itself. An OMIS integrates different kinds of knowledge (e.g., *Best Practices* and *Experiences* (cf. [Althoff et al., 1998], [Landes et al., 1998]). *Design Rationales* [Buckingham Shum, 1998] process knowledge at different levels of representation. Since a KBS's aim is to solve a single task, the knowledge requirements are very homogenous.

Groupware and knowledge dissemination techniques are often not part of a KBS, but are essential for an OMIS because the knowledge stored inside the system has to be communicated to the employees using the system. In addition an OMIS has to integrate many different pre-existing *system components* and legacy applications, which are selected for a specific Knowledge Management strategy.

A KBS can be part of an OMIS. Then it supports the knowledge-based solution of single business tasks. Furthermore, several techniques developed in Knowledge Engineering (KE) can contribute to a methodology for building an OMIS and supporting a *Knowledge Management Strategy.* For instance, the CommonKADS-Methodology [Schreiber et al., 1999] can help to identify and analyze a company's knowledge-intensive work processes. So we conclude that KE not only provides tools and methodologies for building KBSs based on PSMs, but also serves to a large extent as a general framework for building an OMIS.

Chapter 5 A Modeling Schema for Integrating Enterprise Processes and OMIS Support(1)

This chapter develops OMMM (Organizational Memory Modeling Method), an Enterprise Modeling Framework for building OMIS. OMMM is inspired by Problem Solving Methods (PSMs) used in AI to represent dynamic knowledge. Concrete notations based on UML are proposed for enterprise models. The modeling approach is compared to other approaches like CommonKADS and ARIS. The Modeling Framework was evaluated during a project aimed at a Knowledge Management system for industrial design.

5.1 Motivation

There are several possibilities for improving or partially automating business processes: this can be done by e.g., restructuring the business process, by applying standard software (e.g., SAP R/3), or by developing individual software components. To be able to model business goals and to analyze problems which occur within business processes, these processes include organizational structures and activities that have to be modeled, since larger organizations tend to have complex business processes, which are not easily understandable. Since the modeling domain, the enterprise, is usually complex, a schemata or meta-model is useful. Davis [Davis, 1993] summarizes the need for a meta-model (for the domain of software engineering) as follows:

> *"Because there are so many relationships that can exist between items under analysis and so many levels of possible detail, analysts need [...] a knowledge structure, which is just a structured collection of concepts and their interrelationships."*

Enterprise models serve three purposes: first, in the requirements-elicitation phase through modeling business processes (cf. [Kirikova & Bubenko, 1994]); second, to study the impact of a system on an organization (cf. [de Hoog et al., 1996]); and third, to model the software system to implement it (cf. [Jaeschke, 1996], [Decker et al., 1996]).

Looking into *Knowledge Based Systems* (KBS) now, a KBS is a specific kind of individual software component, aimed at improving business processes in an organization.

Taking modern approaches to knowledge engineering for KBS (like CommonKADS or MIKE [Angele et al., 1998]) into account, it should be possible to identify modeling primitives for enterprise modeling approaches, which help to relate a KBS and Organizational Memory to an enterprise model. This chapter integrates the primitives and develops the *Organizational Memory Modeling Method* (OMMM).

1. This chapter is an extended and revised version of [Decker et al., 1996], [Decker & Studer, 1998],[Decker et al., 1997], [Daniel & Decker et al., 1997].

Although there are many existing approaches for enterprise reference schemes (e.g., [de Hoog et al., 1996], [Kirikova & Bubenko, 1994], [Ramackers & Verrijn-Stuart, 1995], [Scheer, 1994], [Jaeschke, 1996]), none is completely appropriate for our purposes. Most approaches do not consider KBS technology as a possibility for improving an enterprise's processes or the enterprise model itself is not sufficiently elaborated. To be able to define an integrated framework including other possibilities for improving business processes, the standard views of an enterprise are determined (also common to other approaches) and a common notation for all views is suggested. To reach this goal, relevant views of an enterprise are defined and the integration of the enterprise modeling with the model for a Knowledge Based System is declared. The next sections starts with an analysis of the relevant models of CommonKADS, which provides the basis for determination of the missing pieces: the exact relationship between business process models and problem solving methods, and a graphical modeling language to define these models.

5.2 CommonKADS

CommonKADS [Schreiber et al., 1994] is a methodology for the development of KBS, which is the result of the Esprit-II project (P5248) KADS-II. CommonKADS supports most aspects of a KBS development project, including project management, organizational analysis, knowledge acquisition, conceptual modeling, user interaction, system integration, and design. The methodology is result-oriented rather than process-oriented. It describes KBS development from two perspectives:

- *Results perspective*: a set of models of different aspects of the KBS and its environment that are continuously improved during a project life cycle, and
- *Project management perspective*: a risk-driven generic spiral life-cycle model that can be configured into a process adapted to the particular project.

For the purposes of identifying the relationship between an organization and a KBS, OMMM focuses on the models. The models defined in CommonKADS to represent different aspects of the KBS are:

- *Organization model*: the CommonKADS organizational model captures an organization from five major perspectives: the activities, the structure, the processes, the power/authority, and the resources in the organization.
- *Task model*: the Task model shows the tasks carried out in the course of a particular process. If the organizational model indicates a particular function, which might usefully be automated, a task model can be produced, which provides a detailed description of the tasks which carry out that function. The task model may also identify the inputs and outputs of each process (or task), and the decomposition of tasks into more specific sub-tasks. Its purpose is to allow identification of tasks which could usefully be performed by an automated system, or by a program and a user working in conjunction.
- *Agent model*: the Agent model represents the capabilities required of the agents who perform a process, and constraints on their performance. The agent model represents all the agents who participate in a problem solving process.
- *Communication model*: the Communication model shows the communication required among agents during a process; it may also specify the form of messages, and specify who takes the initiative in a transaction. The communication model indicates all the transactions which take place between different agents.

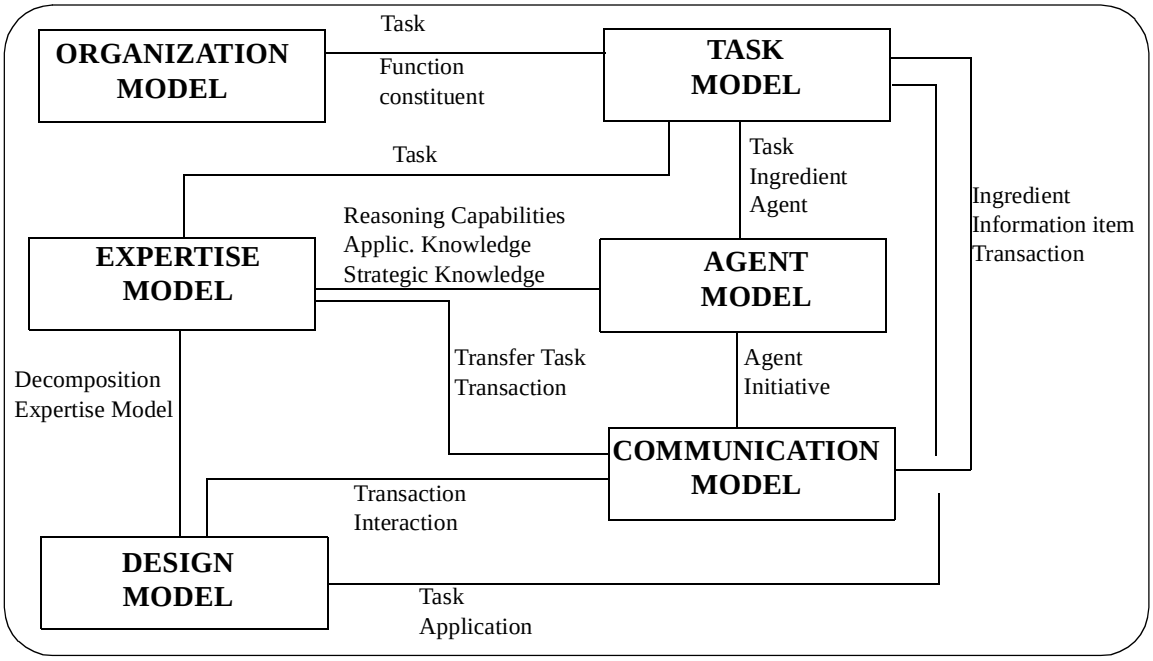


Fig. 5.1. The CommonKADS Model Overview
(taken from [de Hoog et al., 1994])

- *Expertise model*: the Expertise model describes the knowledge used by the KBS to solve its task. A distinguishing feature of the expertise model support in CommonKADS is the use of a library of generic modeling components, e.g., task specific problem solving methods (PSMs) and domain ontologies. This model is divided into three components:
  - declarative knowledge about the domain,
  - the inference processes required during problem solving, and
  - a task structure specifying the hierarchical decomposition and ordering of the inference processes.
- *Design model*: The Design model provides the link between the conceptual models (task, expertise, agent, communication) and the computer implementation. It describes the architecture and detailed functionality of the system to be implemented (The implementation of a design model is not directly supported by CommonKADS, since it is dependent on the particular implementation environment).

Figure 5.1 shows a high-level overview diagram of the models and their relationships (cf. [de Hoog et al., 1994]). For the purpose of defining a schema, which services as guidance for how to model the relationship between a KBS and the business processes of an organization, the next section takes a deeper look at the Organization Model, the Task Model and the Expertise Model.

5.2.1 CommonKADS Organization Model

The CommonKADS Organization Model [de Hoog et al., 1994b], [de Hoog et al., 1996] (see Figure 5.2) was developed to serve three main purposes:

- identification of promising areas for knowledge-based systems applications,
- identification of the impacts of knowledge-based systems on the organization, and

- help for the knowledge engineer to develop a 'feeling' for the organization.

The Organization Model consists of several constituents, e.g., *function*, *knowledge*, and *process*. However, no internal structure is defined for the constituents. Modeling tools that support the CommonKADS methodology represent the constituents of the Organization Model as plain text parts. Furthermore: "*the model is definitely not meant as a general model that can be used for organizational analysis [..]. Features that would normally be part of such a general model, but are most of the time irrelevant for KBS projects, are therefore missing*" [de Hoog et al., 1996].

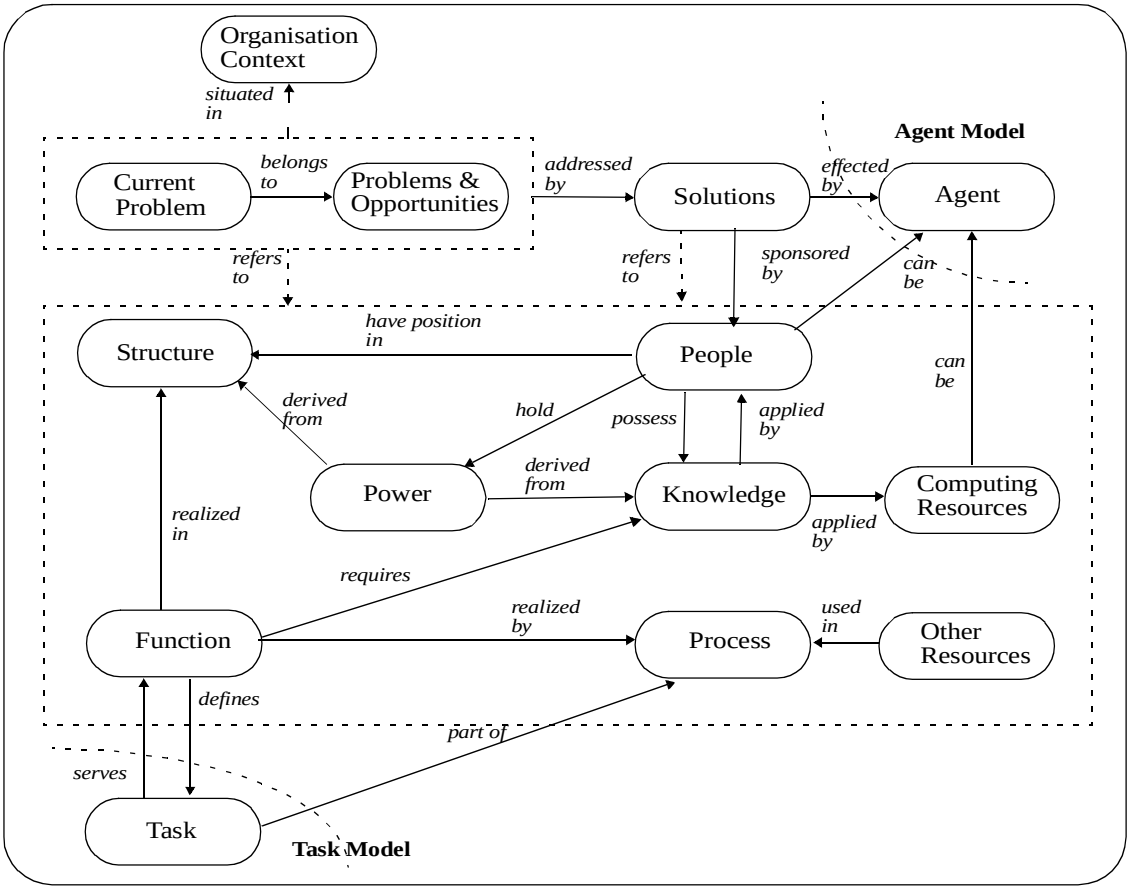


Fig. 5.2. The CommonKADS Organization Model (taken from [de Hoog et al., 1996])

Of particular interest are the process and knowledge constituents, since these constituents need to be connected to the final KBS. No special structure or formalism is suggested to model these constituents. It is just required that:

- the process constituent refers to the overall workflow of the organization's primary process. The major topics that are addressed in the process constituent are: "*what are the dependencies between the organizational functions that have been identified*", and "*how are the organizational functions realized in time ordered tasks*?"
- The knowledge constituent represents the general and high-level knowledge that might influence the definition of the current problem or (the feasibility of) solutions to the current problem. Example issues that are studied in the knowledge constituent are: "*what types of knowledge are present*", "*how is knowledge maintained*", and "*how important is knowledge for the organization?*"

More recently, [Schreiber et al., 1999] suggest worksheets for the elicitation of the CommonKADS models (see Figure 5.3 for an example). [Schreiber et al., 1999] also suggest a diagrammatic representation based on UML.

Organization Model		**Process Breakdown Worksheet OM-3**				
NO.	TASK	PER-FORMED BY	WHERE?	KNOWL-EDGE ASSET	INTEN-SIVE?	SIGNIFI-CANCE
Task identifier	Task name (some part of the process in OM-2)	A certain agent, either a human (see „People“ in OM-2) or a software system (see „Resource“ in OM-2)	Some location in the organization structure (see OM-2)	List of knowledge resources used by this task	Boolean indicating whether the task is considered knowledge-intensive?	Indication of how significant the task is considered to be (e.g., on a five point scale in terms of frequency, costs, resources or mission criticality

Fig. 5.3. Worksheet OM-3: Description of the Process in terms of the Tasks of which it is composed (taken from [Schreiber et al., 1999])

5.2.2 CommonKADS Task Model

The CommonKADS Task Model [Duursma et al., 1993] (see Figure 5.4) is a template used to describe functions of an organization in terms of tasks. The analysis of the tasks helps the knowledge engineer organize his or her view on an expert’s major tasks in a given area, and helps to set the scope of the KBS solution. In contrast to the Organization Model, the Task Model provides some more guidance to the knowledge engineer by providing more structure to the constituents. For example, the task constituent of the Task Model has 9 attributes, which provide information on what may be necessary for the tasks model to be useful. For the purposes of developing a language to represent and model a KBS within an organization model, the most relevant constituent is the task constituent itself. [Schreiber et al., 1999] suggest several different worksheets for knowledge elicitation.

5.2.3 CommonKADS Expertise Model

The *Expertise Model* [Wielinga, et al 1994] (see Figure 5.5) distinguishes between application-related knowledge and problem-solving knowledge. The application knowledge encompasses domain knowledge, inference knowledge, and task knowledge. The problem-solving knowledge consists of problem-solving methods and strategic knowledge.

- The *domain knowledge* captures knowledge about a particular domain in static terms. CommonKADS provides a modeling methodology based on extensible ontologies (aiming at conceptualizing and formalizing a domain by means of concepts, expressions and relations) and a domain model. A domain model describes knowledge on the part of the domain, from a specific point of view. For example, one can describe a causal model of a system, a behavioral model, structural decomposition, etc.

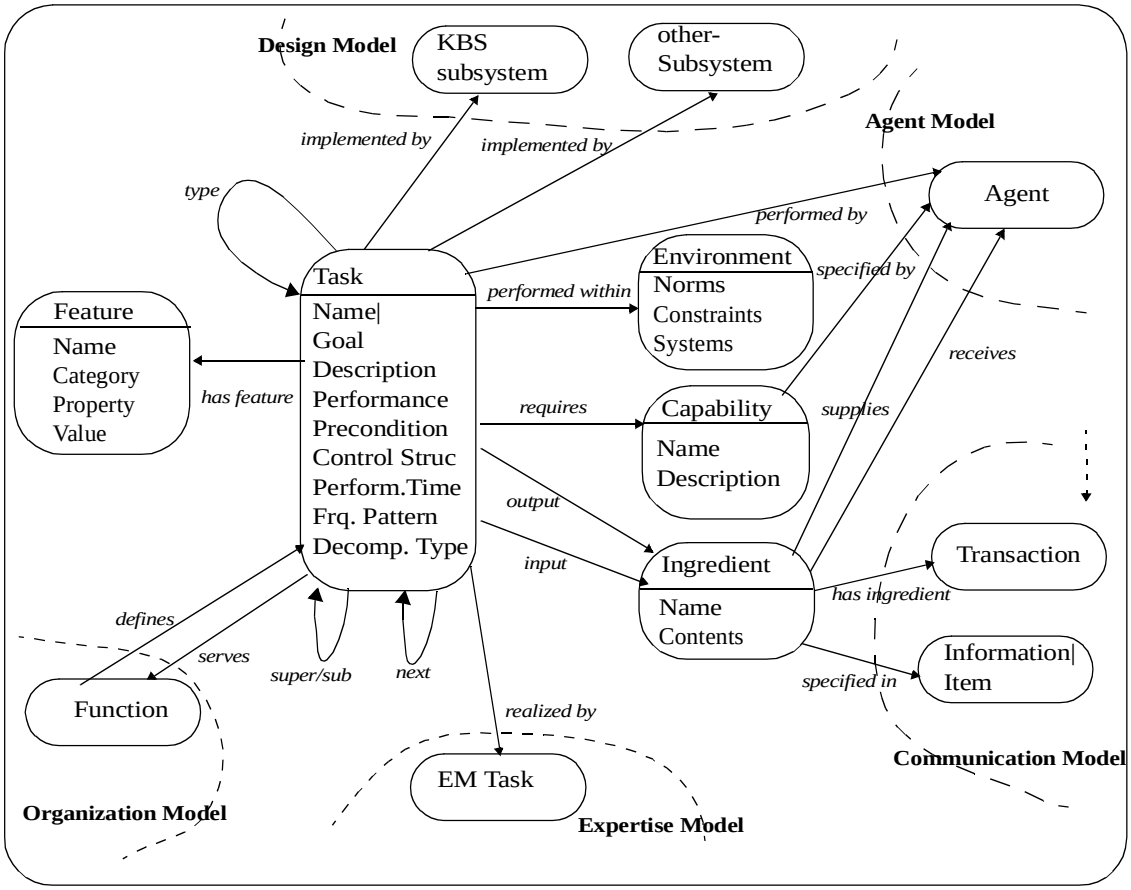


Fig. 5.4. The CommonKADS Task Model
(taken from [Duursma et al., 1993])

- *Inference knowledge* abstracts from the domain knowledge and describes the basic inferences applicable to a certain domain. An inference may operate on input data and may produce output data. Input and output data are provided using *roles*. Two kinds of roles are distinguished: *static* and *dynamic* roles. Static roles provide domain knowledge to the inferences and can only be used to provide input. Dynamic roles can provide input and output.
- *Task knowledge* contains the specification of the tasks. A task definition consists of a *goal*, *input and output roles*, a *task specification*, *subgoals* and related *subtasks*, and a *task control structure*. Three different kinds of tasks are distinguished: *composite tasks* (the tasks are decomposable into subtasks), *primitive tasks* (tasks that can be directly related to inferences), and *transfer tasks* (tasks of interaction with the world, i.e. the user).
- *Problem-solving methods* describe how to organize the domain, task and inference knowledge (see Section 2.2.3, in which an introduction to the notion of problem solving methods was given).
- *Strategic knowledge* describes what kind of model of an application is adequate for a given task environment.

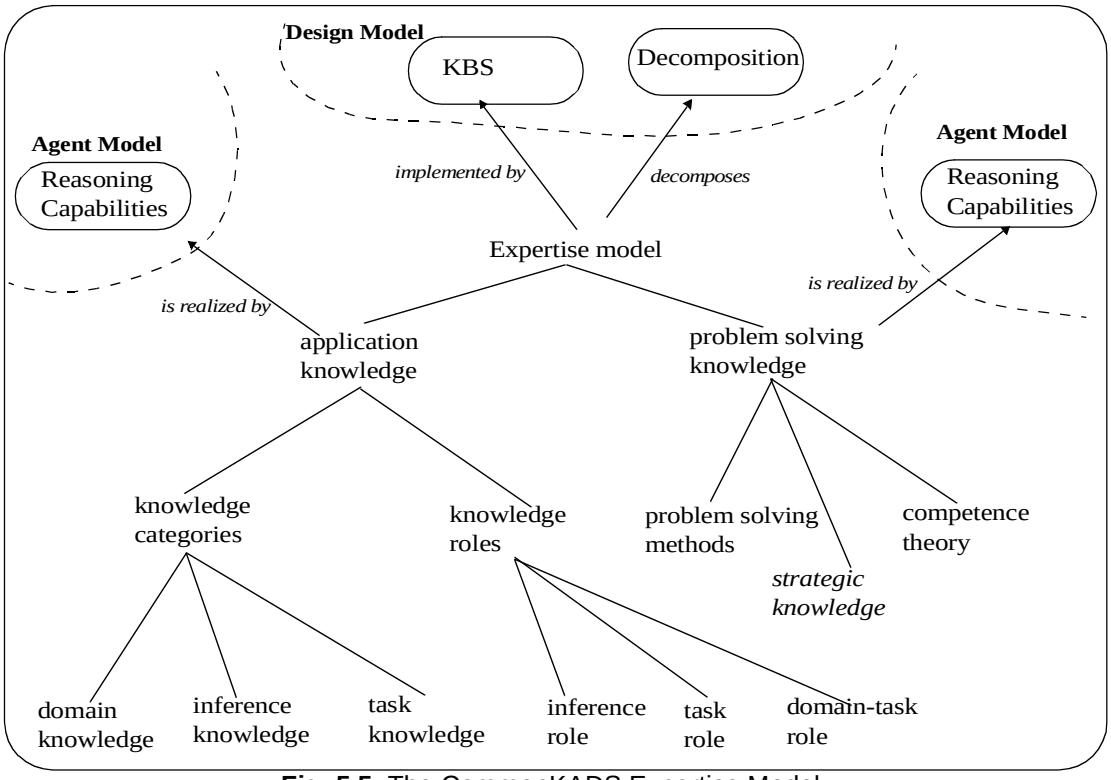


Fig. 5.5. The CommonKADS Expertise Model
(taken from [Wielinga, et al 1994])

The task and inference levels aim at modeling reasoning and problem solving. These levels are supposed to be as independent as possible from the domain: they should be described in a task specific way. For example, if the task and inference levels describe diagnosis of electric components, they should talk mainly about diagnosis and not (much) about electrical matters.

5.3 Relationship between Process Meta Models and Models for Knowledge Based Systems

This section examines process-modeling frameworks and determines the relationship between the notion of a business process and the notion of problem-solving methods (PSMs), as introduced in Section 5.2.3 and Section 2.2.3. In a PSM a (non-atomic) task is decomposed into multiple subtasks, which may be instantiated by other PSMs. A question needing to be answered is whether or not the kind of representation used to model PSMs is similar to the representation used for modeling business processes. If they are similar then the technology developed to represent, express, and process problem-solving methods can be reused for combining and exchanging business processes.

A review of the literature shows that other modeling frameworks have also adopted the decomposition approach used for PSMs: Figure 5.6, taken from [Marin, 2001] shows the meta-model of the Workflow Process Definition of the Workflow Management Coalition (WFMC).[(1)] A business or workflow process is defined in a *process definition*, which can be decomposed in sub-processes and activities. This directly corresponds to the tasks and methods decomposition of PSMs.

1. see http://www.wfmc.org/

Furthermore, the document [Marin, 2001] also defines data flow and control flow primitives - thus the workflow model of the WFMC could be regarded as an extension of PSMs modeling languages, and most of the technologies for composition of PSMs are applicable.

Other suggestions for a standardized business process exchange include the Business Process Modeling Language (BPML) proposed by the Business Process Management Initiative (BPMI.org) [Arkin & Agrawal, 2001]. The BPMI specification supports the decomposition of processes into simple and complex activities. A complex activity is decomposed into sub-activities, which again may consist of complex activities. Control-Flow is defined with complex activities, e.g., `sequence` or `repeats`. Data-Flow is defined using input/output messages - at its core, this model is also very similar to the structure of PSM modeling languages.

Another example of a process definition language is the Process Specification Language (PSL).(1) PSL is an interchange format aiming to help exchange process information automatically among a wide variety of manufacturing applications such as process modeling, process planning, scheduling, simulation, workflow, project management, and business process re-engineering tools. In contrast to the BPML specification and the WFMC's Workflow Process Definition, PSL is axiomatized using KIF. PSL possesses a sub-activity element, allowing for decomposition of activities into sub-

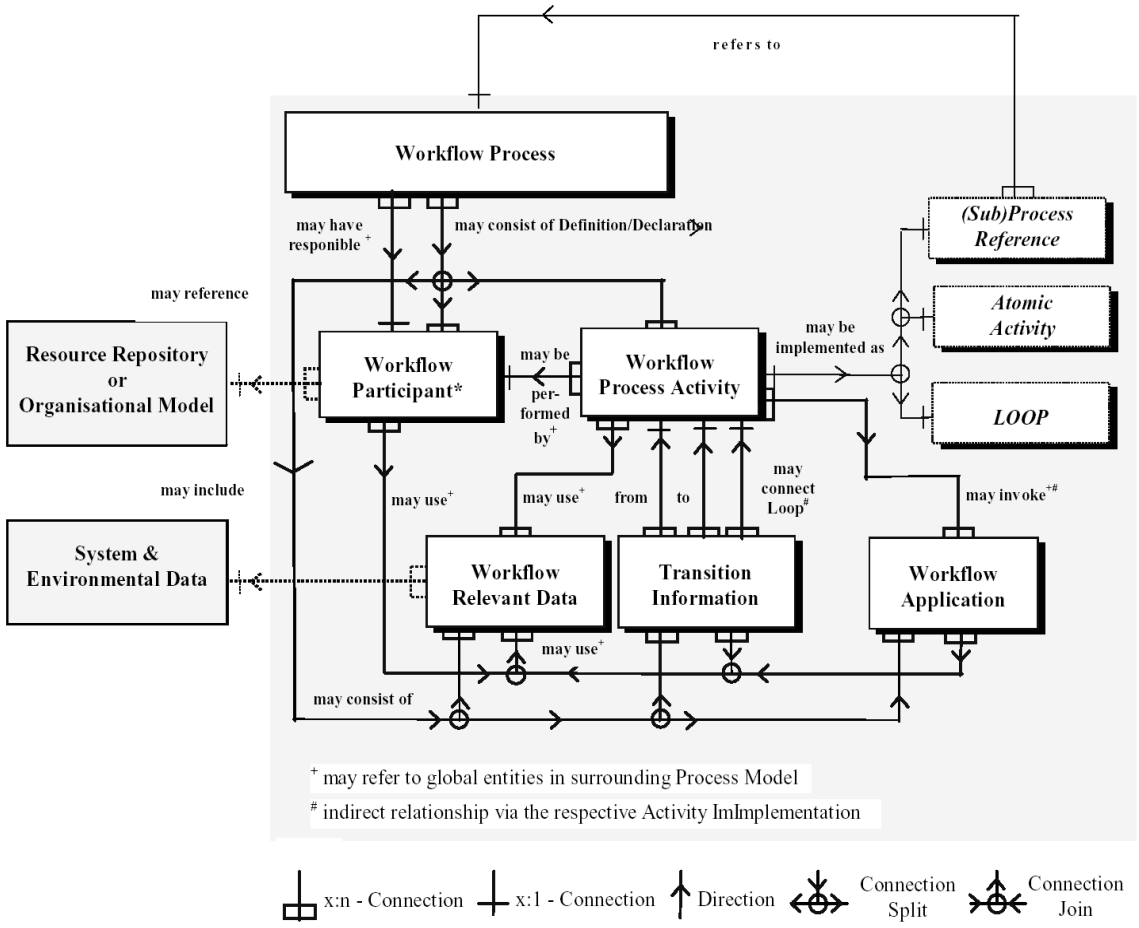


Fig. 5.6. Workflow Process Definition Meta-Model of the Workflow Management Coalition (taken from [Marin, 2001])

1. http://www.mel.nist.gov/psl/

activities. However, PSL allows specification of non-deterministic activities, which means the execution order is non-determined. This is clearly an extension to the model used to specify languages for PSMs, where usually a clear execution order is assumed. Data-flow is specified using a `participates-in` modeling primitive, which indicates that an object is involved in an activity. PSL is more expressive then current PSM specification languages - however, the elements data-flow, control-flow and task decomposition appear again.

5.4 Deriving the Schema of OMMM

A meta-model for OMMM, which mirrors the CommonKADS specification and provides a concrete diagrammatic syntax for the creation of models for business processes and PSMs, needs to fulfill the following requirements:

- The diagrammatic language should provide means to capture the static and dynamic aspects of business processes and problem solving methods. As argued in the previous section, the same modeling primitives should apply to both.
- The schema should closely integrate the various components of business-process models as well as expertise models.
- The language has to support modeling at various levels of details and with incremental formalization in mind. A Business-Process Model might be used just as a visualization and communication platform or as executable specifications for Workflow Management Systems. The ability to incrementally formalize a given model from a high level view to an executable specification reduces cost, since the model process can stop at an appropriate level of detail and formality. Incremental formalization has the advantage that the errors in the modeling process can be corrected early on, without much effort wasted if, at the beginning, a wrong direction was chosen.

It is generally accepted that for an operational description of a system three views are sufficient (see Figure 5.7 taken from [Ramackers & Verrijn-Stuart, 1995]). These three perspectives have

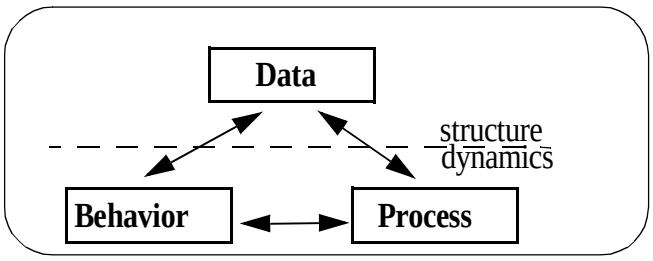


Fig. 5.7. Model Perspectives
(taken from [Ramackers & Verrijn-Stuart, 1995])

principled relevance for modeling: they are generally used to model static and dynamic information. Dynamic aspects can be identified in several parts of an enterprise (e.g., in the business processes and in the processes that are executed in a software system). Although the level of abstraction is different in these two processes and they are probably modeled in different layers of an enterprise model, the same notation can be used for both. A notation for modeling an enterprise should be (1) widely accepted, (2) useful for different types of software systems (e.g., information systems and knowledge based systems) and, (3) powerful enough to model all relevant aspects. Lastly, it should bridge the gap between the user world and the developer's world, so that models are usable for software development or adaptation, but can still be communicated to non-software developers.

UML,[1] created for software development, has also proved its usefulness in other areas: design of knowledge-based systems [Schreiber & Wielinga, 1993] and enterprise modeling ([Bauer et al., 1994], [Kaschek et al., 1995] (based on the OMT notation (cf. [Rumbaugh et al., 1991])). Furthermore, the diagram types defined in UML correspond closely to the different constituents of Figure 5.7. The data constituent in Figure 5.7 corresponds to the *static object diagrams* of UMT, the behavior constituent corresponds to the *statecharts* and the process constituent corresponds to *dataflow diagrams*, which resemble activity diagrams in UML or the *functional model* in OMT. In the OMMM approach statecharts are used for the behavior constituent and DFDs (dataflow diagrams) are used for the process constituent. Statecharts, as well as dataflow diagrams, account for incremental formalization: a simple statechart contains only nodes and (unlabeled) arcs between the nodes. By adding formal conditions the statecharts becomes executable.[2] The last requirement is the integration of the different viewpoints of the Organization Model and the Expertise Model. The domain model and the data view are unified, since for the organization the available data is what constitutes the input of the KBS. Furthermore, OMMM regards the tasks of the KBS as a refinement of the business process tasks contained in the process constituent of the CommonKADS Organization model. In the following, the schema for the integration of organizational modeling and knowledge engineering is presented. The schema is an *ontology* in itself, since it defines the concepts and relationships of the concepts that OMMM enables to be modeled. The explicit definition of the schema and the connection of the Organization and the Expertise Model are the main differences between the OMMM approach and CommonKADS.

5.5 Views of OMMM

The view presented first is the *Data View* of OMMM (see Figure 5.8), a view that is very common in enterprise modeling. It allows the modeling of documents, database schemata etc. used in an enterprise and in its business processes. CommonKADS does not explicitly model this view, but can be captured by the *Other Resources* constituent (see Figure 5.2) in the Organization model. The data view is also related to the domain knowledge captured by the Expertise model. OMMM deploys the UML class diagrams for modeling of the data view, which provides the necessary ontological modeling primitives to represent the domain model.

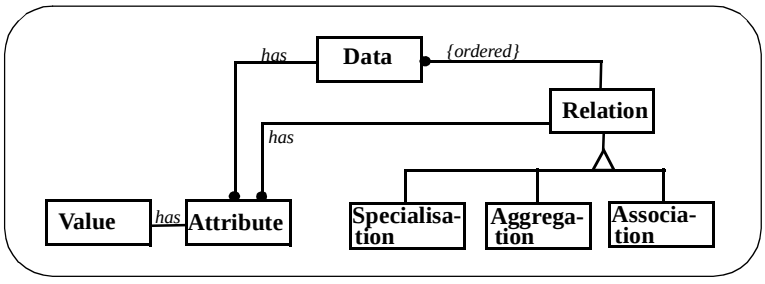


Fig. 5.8. Data View

1. See http://www.omg.org/technology/uml/ for downloadable specification documents of UML

2. Note that the metamodels (views) don't contain the modeling primitives of statecharts and data flow diagrams. They are left out deliberately, since they don't provide any additional insight into the structure of business processes. For OMMM purposes familiarity with statecharts and dataflow diagrams is assumed.

The *Process View* (see Figure 5.9) is another very common view of enterprises. In this view certain business processes are modeled using defined primitives, e.g., ARIS uses EPCs (event-driven process chains), which are claimed to be a kind of high-level petri net. In CommonKADS no further modeling primitives are suggested for this view.

OMMM uses the following modeling primitives for this view: OMMM distinguishes between business goals and processes. For each business goal, task decomposition is expressed through UML class diagrams. For the dynamic aspects (control-flow and dataflow), OMMM uses statecharts and dataflow graphs. Furthermore, OMMM distinguishes between three levels of detail in processes: *business level*, *job level* and *job-part level*. Business process (BP) tasks are done at a high-level view, they abstract from single positions, and typically a BP task describes a high level task of a department (e.g., sales). Job tasks are related to a particular position, and give a high level description of the tasks of an employee. One BP task may involve several employees, but a job task always involves just one employee. The next category are job-part tasks, job-part tasks decompose job tasks even further, and describe single functions necessary to do a job task. The dataflow diagrams are used to describe the dataflow inside the process, and statecharts are used to describe the state of the process. Each task is therefore identified within a state of the overall process. Note

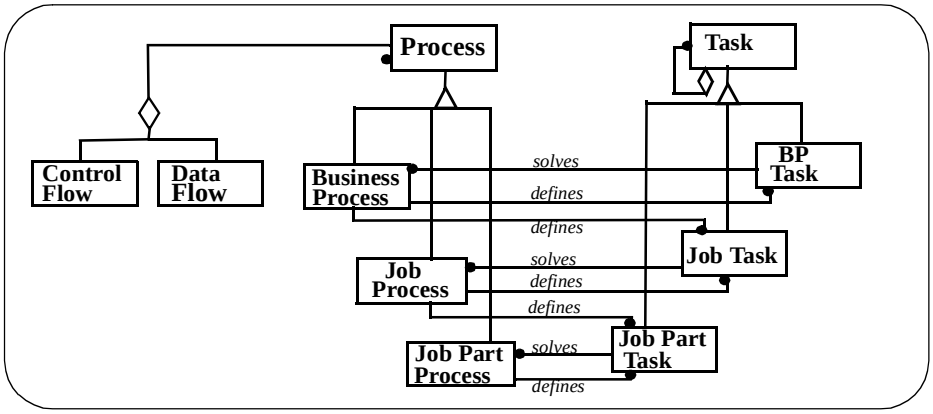


Fig. 5.9. ProcessView

that the OMMM schema does not contain the modeling primitives of statecharts and dataflow diagrams.These definitions described in the UML specification documents.(1)

The *Organizational Structure View* (Figure 5.10) is intended to capture the static organizational aspects of an enterprise, e.g., the structure of the organizational units. The first entity to model is the structure of the organizational units. Therefore, OMMM models a decomposition of the organizational unit class into smaller organizational units. Another relationship exists between organizational units and jobs or collection of jobs. A job is also related to a collection of tasks, which is a connection from the organizational structure view to the process view. An organizational unit is also related to a set of tasks (modeled in the process view) which it fulfills. OMMM differentiates between jobs and job places because both are important to take into account for the needs of employees (and pose possible problems to solve). To allow statements about larger enterprises, the job type and organization unit type class enable statements about collections of objects. ARIS supports this view through organizational charts; CommonKADS suggests ‘a tree-like diagram’ for the structural constituent. OMMM deploys UML class diagrams for modeling this view.

1. See http://www.omg.org/technology/uml/

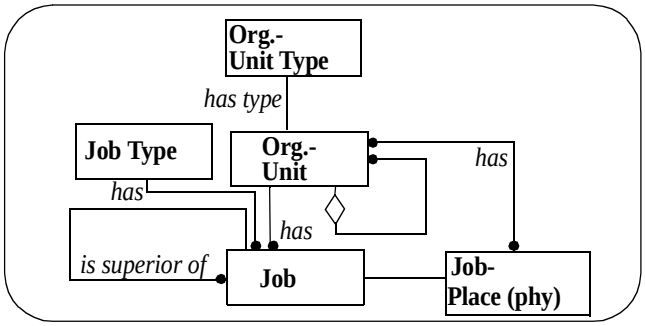


Fig. 5.10. Organizational Structure Schema

The *Staff View* (Figure 5.11) allows incorporating a human-centric view into the modeling process. This covers several aspects: e.g., the engineering of human-centric business processes as well as the design of ergonomic software products. For this purpose the two classes *Role* and *Ergonomic Requirements* are defined. Role objects capture the relationships between employees and their jobs. In some modeling approaches, the S*taff View* and the W*orking Tool View* are mixed together into a R*esource View.* This is not an appropriate choice: modeling should always be done with requirements of the staff in mind, since employees are primary knowledge carriers - unlike other resources. ARIS subsumes the staff view with the *Organization View* of ARIS, whereas CommonKADS has a *people* constituent, but does not define any further structure for it. OMMM regards this as a separate view, because from a KM perspective, the staff of an enterprise is the most important asset. OMMM uses UML class diagrams as the modeling language for this view.

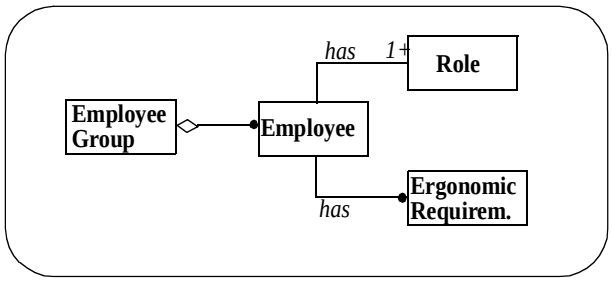


Fig. 5.11. Staff Schema

The *Working Tool View* (Figure 5.12) allows for modeling relations between a task and the tools that are required to complete the tasks. To design business processes one should know which tools used in the process to detect e.g., media breaks. Therefore OMMM requires additional attributes to characterize working tools. The type may be e.g., *computer* (general), *KBS* (more specific) or even a *typewriter.* The attribute *media* says something about the media used in or with that tool, e.g., e*lectronic documents* (for a computer system). So there exists a link from the *Working Tool* to the data view, which is used to model the documents and electronic data. With the attribute *use constraints*, restrictions of the working tool may be expressed; e.g., a computer system is not usable during system maintenance. The function attribute should give information about the role of the working tool in the business processes, e.g., give answers to the following questions: "*Is the tool essential? For which tasks is the tool usable?"*

ARIS captures this view in the different stages of the ‘ARIS house’ (Figure 5.16). However, KM depends heavily on the tools and techniques for building, maintaining, and using knowledge. Therefore a separate view was introduced. CommonKADS has a *Computing Resource* constituent, but defines no internal structure. OMMM uses UML class diagrams as the modeling language for this view.

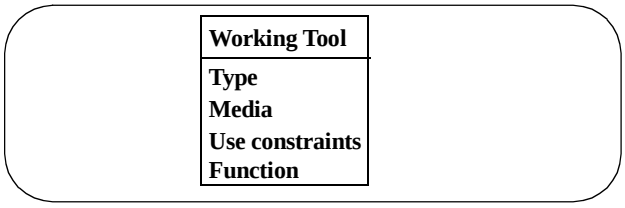


Fig. 5.12. Working Tool Schema

For the development of an information system as well as for a knowledge based system it is necessary to determine at which point in the work process the employee needs additional information to perform his/her task. The *Communication/Cooperation View* (Figure 5.13) describes communication and cooperation that occurs in a business process. The design of the Communication/Cooperation View is similar to known techniques for describing human/computer interaction (interaction diagrams) [Rumbaugh et al., 1991]. The communication/cooperation objects may be instances of the classes *employee*, *job*, *process* and *working tool* defined in the other views. These instances are the objects that communicate and cooperate with each other. This modeling decision provides the opportunity to model several levels of detail. The lower part of the diagram defines a relationship and corresponds to a simple link between two of these objects: the link is annotated with attributes, which make assertions about the owner, the contents etc. of the communication/cooperation. If two persons communicate/cooperate with each other, they do this usually within a task. To model the information flow, this task/process dependency can be noted in an attribute. ARIS does not possess a communication view, instead the communication aspects are regarded as part of the business processes. This has the drawback that communication which occurs aside from the business processes cannot be modeled. CommonKADS has its own model for this view, the *Communication Model* [Wielinga et al., 1993] that is primarily intended to capture the interaction between the staff and a KBS. As a notation OMMM uses UML class diagrams, or (equivalently) interaction diagrams.

These are the views that OMMM regards as usable for many types of systems. Because Knowledge Management is a very diverse field, other views on an enterprise may be important for different techniques used to implement Knowledge Management. In the following, the views of OMMM are proposed that are important for building KBSs as part of a general Knowledge Management strategy.

The *Source View* (Figure 5.14) models relevant sources for the knowledge elicitation process of building Knowledge Management systems. It is one of the model constituents necessary for the development of a knowledge based system. The Source View supports the planning of the knowledge elicitation process, where e.g., different staff members have to be interviewed. Therefore, OMMM distinguishes between sources of knowledge and the knowledge itself. Sources

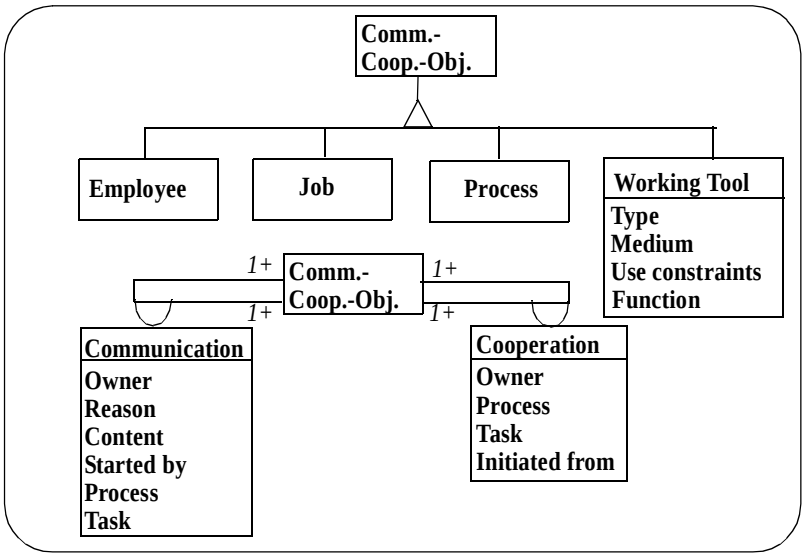


Fig. 5.13. Communication/Cooperation Schema

may be e.g., employees (who are active) but also books, which are passive. After taking the decision about what kind of knowledge is relevant for a business process improvement, it needs to be determined which knowledge should be elicited and modeled and what sources are relevant for the knowledge. The knowledge can be further classified: "*Which field does it belong to? Is it directly available and understandable (how much effort has the knowledge engineer to put into the elicitation)?"* The answers to these questions are modeled using the different attributes.

ARIS does not support the notion of knowledge, and thus does not support this kind of modeling. OMMM uses UML class diagrams as the modeling language for this view.

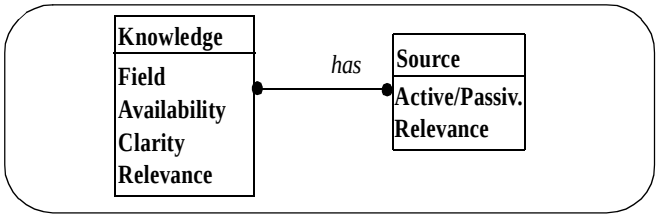


Fig. 5.14. Source Schema

The *Expertise View* of OMMM (see Figure 5.15, part c) is similar to the structure model of MIKE [Angele et al., 1998] and the model of expertise in CommonKADS: a task is solved by a problem-solving method, which needs domain knowledge. The problem-solving method defines a task decomposition ('*divide and conquer'*) of the original task. The PSM also defines data, the control-flow and dataflow among the subtasks. The Expertise View is a special view: it is the only one which contains all different model views presented in Figure 5.7. This is because it represents a complete description of a knowledge based system, which requires all three model components.

ARIS does not have the notion of KBS and PSMs. As in the Process View, class diagrams, statecharts and dataflow diagrams are used: dataflow diagrams describe the dataflow, and statecharts are used to describe the state of the process (the control-flow). Each task is therefore identified within a state of the overall process.

Several connections exist between these views, most of which are standard. The most important connections are those between the process view, the expertise view and the data view. These connections capture where a KBS may support an employee at the job-part task level. At this level an employee works on a closed task, in which his knowledge mainly determines how to solve the task. This is the point where a KBS or Organizational Memory becomes important. The tasks performed by problem-solving methods are just subtasks of the task the employee performs. The

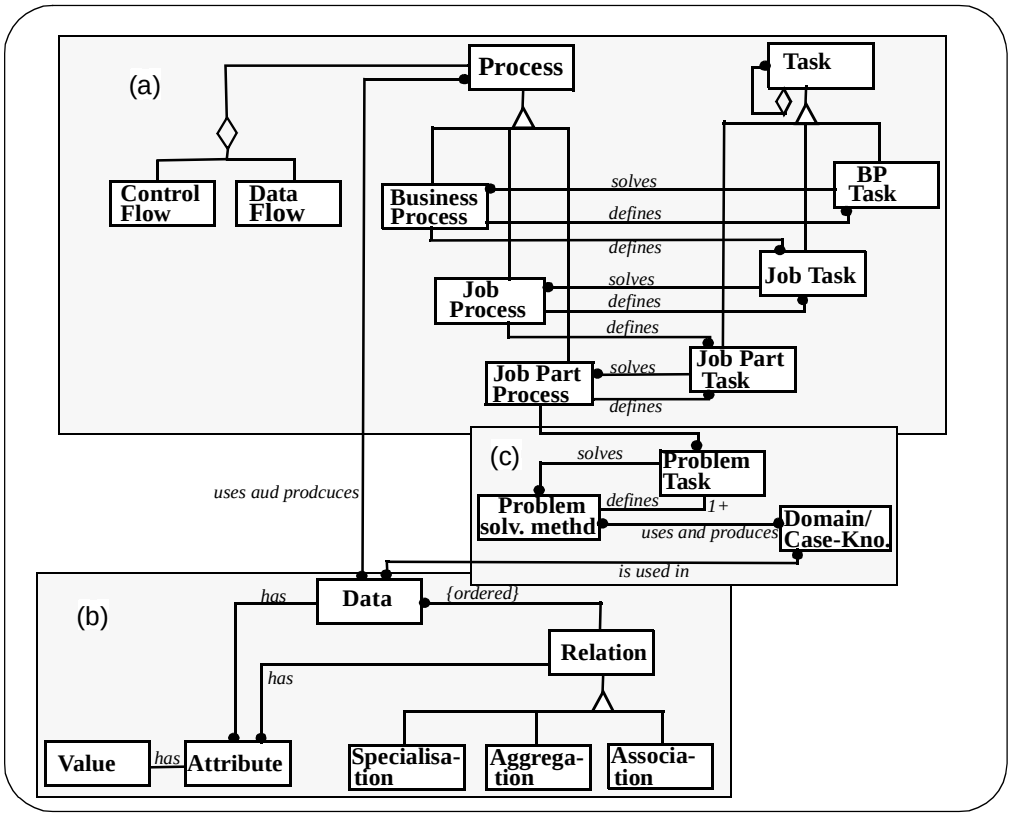


Fig. 5.15. Core Relations between (a) Process View, (b) Data View and (c) Expertise View of OMMM

connection suggests how enterprise-modeling tools are able to support the creation of KBS: based on a particular task (at the job-part level) the enterprise modeling tool can connect to a modeling tool for expertise models.

5.6 Related Work in Enterprise Modeling

A modeling approach (including tool support) for business process reengineering and description of information systems is ARIS (*Architecture of Integrated Information Systems* [Scheer, 1994]). The architecture or basic orientation frame of ARIS is given by two dimensions orthogonal to each other (see Figure 5.16). In one dimension, views on the object worlds to be modeled are distinguished.

- In the *Organization View* the relations between enterprise units and their classification into the organizational hierarchy are modeled.

- The *Data View* describes objects, their attributes and inter-object relations. Furthermore, the data view contains events that may initiate and control processes.
- The *Function View* embodies functions, which are part of processes and determined through the creation and change of objects and events. A complex function can be decomposed into more elementary ones.

Another dimension in ARIS is the level of implementation, e.g., *requirements definition level*, *design specification level*, and *implementation description level*. The starting point is the managerial/economic description of the enterprise domain, the requirements definition. These concepts are formulated in business terms, but are strongly influenced by technical possibilities. The resulting models of this first level are noted in semiformal diagrams. The design specification is also modeled semiformally, but uses terms of the envisioned information systems solution (i.e., it may speak about modules and transactions). The last part consists of the physical implementation description of the upper levels.

For modeling, several diagrammatic notions are proposed: e.g., Event-driven Process Chain (EPC) for modeling processes, Entity Relationship diagrams (ER) for data modeling. Shortcomings of ARIS for building Knowledge Management systems are: ARIS has no notion of knowledge, so it is not possible to identify knowledge assets and furthermore, the distribution of tasks between human beings and computers cannot be modeled explicitly.

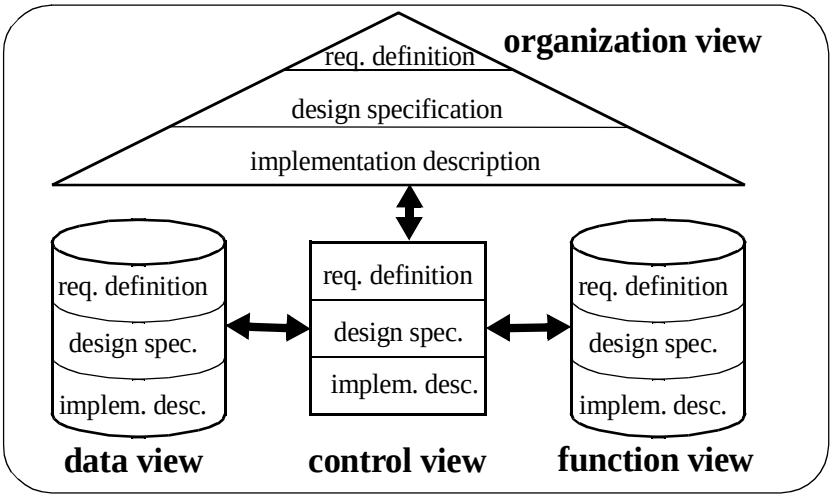


Fig. 5.16. The ARIS house

Apart from approaches like ARIS and CommonKADS, the presented approach is also related to the Enterprise Ontology presented in [Uschold et al., 1998]. The Enterprise Ontology is a collection of terms and definitions relevant to enterprises, and thus has aims similar to OMMM. However, OMMM is aimed at business modeling from a knowledge perspective: to identify the sources relevant for business processes and to model the relevant knowledge processes, even on and below the employee level. Consequently, the source view and the communication view are not present in the Enterprise Ontology. Also, the expertise view and the detailed description of processes are not present in the Enterprise Ontology. The Enterprise Ontology also does not provide a graphical representation for the defined vocabulary.

The reference scheme defined in [Kangassalo et al., 1995] is similar to the one presented here: they define a reference scheme with a non-standard notation for business modeling and show an operationalization with high level petri nets. OMMM focuses on the standard notation UML and the interface between knowledge engineering (especially problem-solving methods) and business process modeling.

In [Kirikova & Bubenko, 1994] the notion of an *Enterprise Model* is introduced. Such an Enterprise Model is composed of several sub-models: objectives model, activities and usage model, actors model, concept model, and information systems requirements model. In that way, the Enterprise Model aims at capturing all aspects that are relevant when developing an information system in a business context, i.e. it defines a meta-level framework, which specifies the type of knowledge which has to be modeled within each of the sub-models.The OMMM approach can be interpreted as a concrete instance of such a meta-model, i.e. as a proposal of how to represent such sub-models and their relationships.

A meta-model approach for modeling business processes is described in [Jarke et al., 1997]. Jarke et al. propose the definition of a language meta model, which can be used to describe different views on business processes. Their proposal for a meta language aims at modeling quality-oriented business processes and puts emphasis on supporting the negotiation process needed to achieve coherent views. On the other hand, their approach does not consider the development of a KBS and does not pay much attention to the persons working in an organization.

The *Brahms-Framework* (cf. [Clancey et al., 1996]) aims at the informal modeling of scenarios in a situational way: every activity of an employee is collected and described in a so-called work frame. A work frame contains a semi-formal description of an activity, which can be further analyzed. Regarding the level of detail, Brahms' models are between cognitive models and business process models. The main differences to the OMMM approach are that OMMM doesn't focus on informal aspects of an enterprise: instead OMMM models several views of an enterprise aiming at a smooth transition from business process models to problem-solving methods using a standard graphical notation from software engineering. Also Brahms is not especially directed to the development of Knowledge Management Systems.

A related approach to OMMM is presented in [Kingston et al., 1997]. The CommonKADS Task Model is extended by IDEF3 and the CommonKADS Communication Model is extended by Role Activity Diagrams. [Kingston et al., 1997] does not provide an integrated modeling schema for business processes and knowledge based systems - IDEF3 is only used on the task level, not for the expertise model. It seems possible though to extend the coverage of IDEF3 to the expertise model.

INCOME/Star (cf. [Oberweis et al., 1994]) is an environment for the cooperative development of distributed information systems based on high-level Petri nets (Nested Relation/Transition nets (NR/T-nets)). NR/T-nets allow for the modeling of concurrent processes and related complex structured objects in distributed business applications. INCOME/Star supports a stepwise top down approach for Entity/Relationship based object modeling, and posseses an evolutionary process model for information system development; ProMISE. In [Weitz, 1999] INCOME/Star is extended with XML/SGML Petri nets. XML-Petri nets can be interpreted as an extension of XML to represent the dynamics of Workflow Management, and enable the description of change of (parts of) documents within a business process. INCOME/Star is related to the OMMM approach as follows: Petri nets capture the dataflow and control flow structure of processes. OMMM relies on the separate description of control-flow and dataflow, since integration of PSMs and business process models is simplified by using the same representation mechanisms for both.

It is scarcely surprising that the entities and concepts dealt with in other modeling approaches (e.g., the Toronto Virtual Enterprise (TOVE) project [Fox et al., 1996]) are very similar to OMMM. It would be highly surprising if this was not the case. However, none of the approaches has paid attention to the following: integration of the development of KBS in a business process reengineering methodology for enterprises. OMMM achieved the intregration by defining a common model for enterprise modeling and knowledge engineering.

5.7 An Application: ERBUS

5.7.1 Introduction

The section describes a case study of the OMMM approach. OMMM was deployed in the WORKS (*Work Oriented Design of Knowledge Systems*) project. The project started with the goal of building a KBS called ERBUS (*Ergonomie-Beratungs und Unterstützungs-System*) to support industrial designers in answering questions regarding ergonomic issues. The motivation for ERBUS was that a large amount of ergonomic knowledge existed but this knowledge was often neglected by designers of industrial products. However, during this project it turned out that a KBS was not suitable: large portions of knowledge could only be represented informally, knowledge was created through lessons learned, and many tasks in the design process had to be supported and integrated to avoid media breaks (e.g., data from documents had to be retyped into the support system) and to encourage the designer to use the system. Furthermore it was not clear at the beginning which tasks needed support.

Therefore ERBUS was redesigned as a Knowledge Management system, incorporating many techniques for information and knowledge representation and delivery. The design process and the documents and data that a designer produces were modeled. The created model helped to identify two tasks that may be supported by knowledge based systems and several other tasks that may be supported otherwise. The model also helped to connect the documents created during the design process to the problem solving methods of the knowledge based system. In the following the derived models are described, but the implemented system is only sketched out. More details on the implemented ERBUS system are available in [Hoffmann et al., 1999].

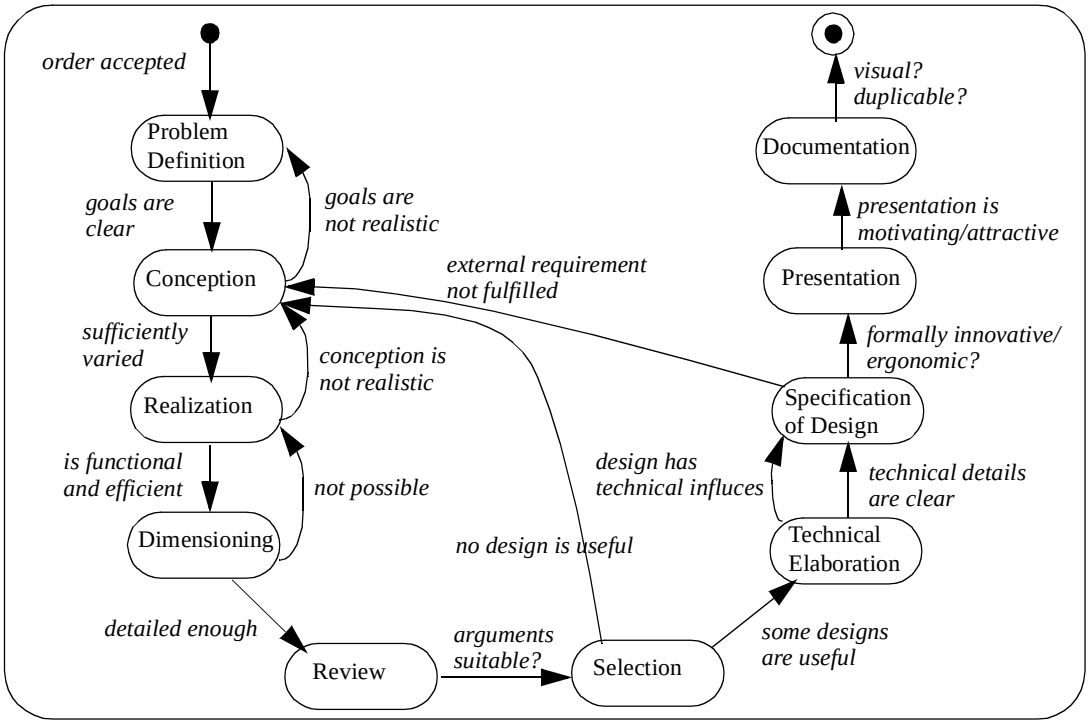


Fig. 5.18. Control Flow Specification of the Design Process

5.7.2 Modeling the Design Process of Industrial Designer

[Kühn & Abecker, 1998] state that for designing a Knowledge Management system, the first question that needs to be asked is: *"What are the tasks to be supported?"* To provide an answer to this question the process of industrial design was modeled and analyzed for support needs in design enterprises. The results were used in the requirements elicitation of the overall system. The design process was described using the OMMM approach, with the views described in Section 5.5. Because only small companies were examined, only a small subset of the views was modeled . The project was restricted to the modeling of the processes and started with the task decomposition (Figure 5.17) of the design process. The control flow of the process was represented using a statechart (Figure 5.18). A dataflow diagram was created to capture the dataflow between the

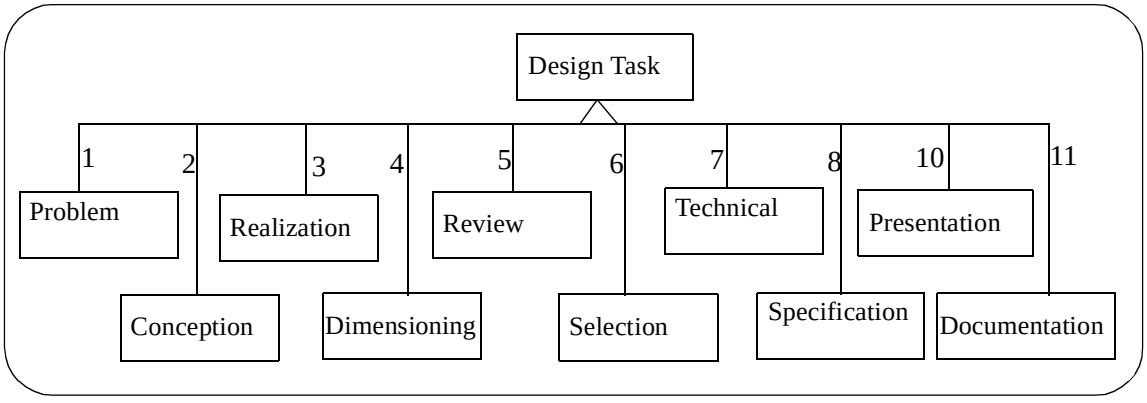


Fig. 5.17. Task Decomposition of the Design Process

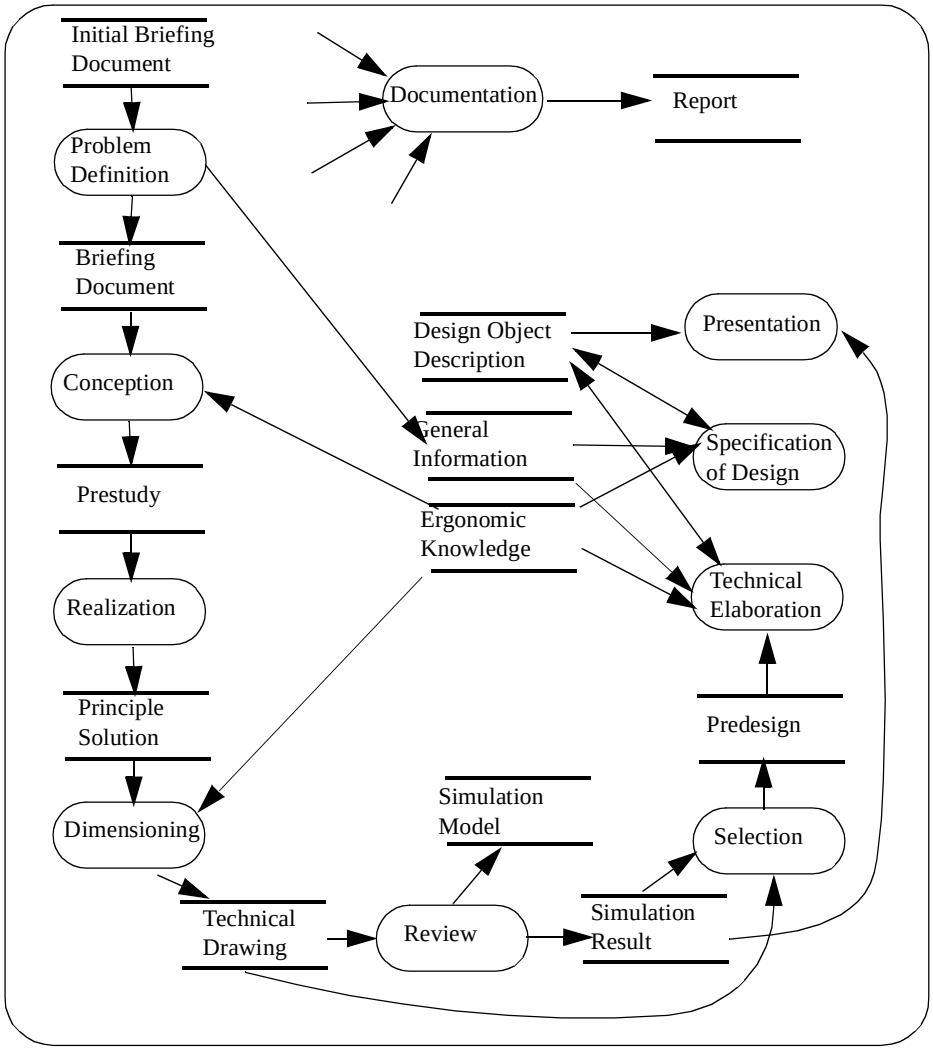


Fig. 5.19. Data Flow Specification of the Design Process

subtasks (Figure 5.19). The design process is creative and highly cyclic, however, it was possible to identify certain behavior patterns. Depending on the type of overall design task some process steps within the process model may be left out or may require less emphasis.

5.7.3 Identifying Tasks to Support

Using the data model and the process model, possibilities for supporting the design task may be identified. Realizing these possibilities introduced new data objects (which are necessary in the new process) or new process steps.

Throughout stages 1 and 2, support should be given for the elaboration of briefing and requirements documentation as well as for conceptualization. Support may be provided by referencing design documents, collected in former projects and possibly reusable in a new project. Similar properties and problem solutions need to be identified and delivered by ERBUS. At the same time, ergonomic issues (problems) are identified and solutions proposed based on the reference designs and rules

established for the ergonomic consultation. Eventually, a list of requirements is drafted, containing properties of the design artifact and their respective values. Created documents are managed by a document management system and are available for use in future projects.

The designer's required knowledge and skills are particularly complex during stages 4 - 8. In some areas, such as ergonomics or production technology and construction, specific information is difficult to obtain and probably not contained in ERBUS. Therefore, consulting an expert is necessary and is supported by the system.

During tasks 4 to 6, detailed information and refined knowledge for the elaboration of the tasks is required. The information is derived from the available reference designs and from the properties defined in stages 1 and 2.

The technical elaboration is usually a rather cumbersome task, since the overall structure is defined but the details are still open. Since components and requirements of the design object (the goal) are well known in stages 7 and 8, it is possible to support the elaboration with a KBS for parametric design (cf. [Motta, 1999]).

Support throughout stages 1, 2, 9 and 10 was provided mostly in the form of provision of information and processing of documents (especially old designs). The need for ergonomic consultation arises in stage 1 and 2 in the context of the definition of requirements.

During all other stages the need for ergonomic knowledge may emerge. This knowledge is complex and in part highly sophisticated. A comprehensive processing, which focuses on practical needs, covers most of the knowledge required by the designer. The demand for the knowledge-based component depends upon the depth of consultation desired during the various stages.

The industrial design process, in general, is very sensitive to input from the outside world. Therefore, means of support - ranging from information research to consultancy - should be adaptable to changes in the real world.

A task not directly modeled in Figure 5.17 is the maintenance and revision of the design object, which occurs after completion of the design process. The collection of the major design decisions during the whole design process together with documentation helps to avoid doing work twice.

5.7.4 Experiences and Assessment

OMMM has proven to be very useful for the modeling of the design processes. It was particularly useful to communicate the modeled processes to domain experts. Initially OMMM was aiming to enable the domain experts to model their processes themselves. This approach turned out not to be feasible; the domain experts (industrial designer) were not able to represent their work processes with a formal language, and the graphical language (UML) that OMMM deploys was still too formal.(1) The representation and process elicitation had to be done by a knowledge engineer, who conducted a guided interview of the domain experts.

After the processes were elicited, the OMMM models were useful to identify the tasks to support the general flow of ERBUS.

1. The designer reacted with the following statement: "*We don't want to press our creativity into small boxes.*"

5.8 Summary

This chapter presented the OMMM (Organizational Memory Modeling Method) approach, an Enterprise Modeling Framework for building OMIS and defining a Knowledge Management strategy. OMMM was inspired by Problem Solving Methods (PSMs) used in AI to represent dynamic knowledge, and arguments were presented that the PSM notion is indeed useful to represent business processes. OMMM separates staff resources from other resources, since the staff is the primary knowledge carrier. Concrete notations based on UML were given for enterprise models. The modeling approach was compared to other approaches like CommonKADS and ARIS. OMMM was evaluated during a project aiming at a Knowledge Management system for industrial design.

PART II KNOWLEDGE MANAGEMENT WITH ONTOBROKER

“This 'telephone' has too many shortcomings to be seriously considered as a means of communication. The device is inherently of no value to us.”

Western Union internal memo, 1876.

Chapter 6 Introduction to Ontobroker

6.1 Motivation

Before the Web and the Internet were widely deployed, information was created, stored and distributed only by a limited number of people and institutions. Print media played a major role as an information distributor. The progress in cheap and reliable storage and network technologies changed this picture dramatically. Figure 6.1 shows the development of the costs per gigabyte and

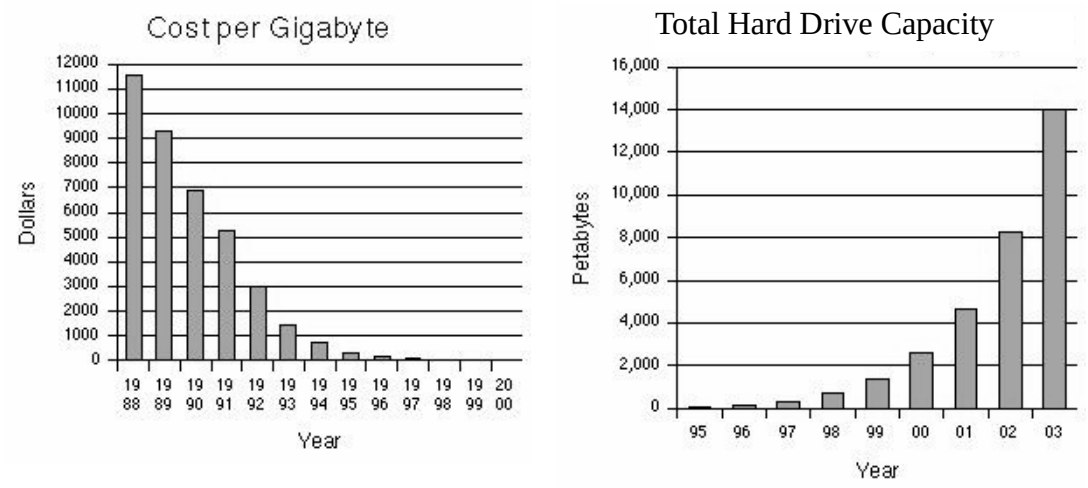


Fig. 6.1. Drive Cost per Gigabyte and Capacity Shipped (IDC reports.)
(Source: http://www.sims.berkeley.edu/research/projects/how-much-info/)

projected shipped capacity of hard drives; the costs per megabyte are approaching zero and the shipped storage capacity has increased dramatically.

The invention of the World Wide Web by Tim Berners-Lee in 1989 [Berners-Lee, 1999] triggered the growth of the Internet. Figure 6.2 shows the increase of host computers connected to the Internet in recent years. Both factors, the progress in network and storage technology, changed the landscape of information providers dramatically. As a consequence, information may be published by a much larger number of people and institutions at a much lower cost than was possible a couple of years ago. As an example, [Odlyzko, 1997] claims that the cost of publishing in an electronic journal is about $75 per paper, compared to the $2,000-4,000 per paper that print journals require. This has led to an increase in the information available on the Internet and especially on the World Wide Web.

Taking a closer look at the information available on the Web, it is possible to distinguish between two kinds of Web content. One, the *surface* Web, is what everybody knows as the ‘Web’ - Web content that consists of static, publicly available Web pages. While an exact measurement of the surface Web is impossible, a study performed by NEC Research [Lawrence & Giles, 1999] in 1999 estimates the ‘surface’ Web to have 1.5 billion Web pages, an 88 percent increase from 1998. This suggest 1.9 million Web pages are created each day. A more recent study estimated the surface Web

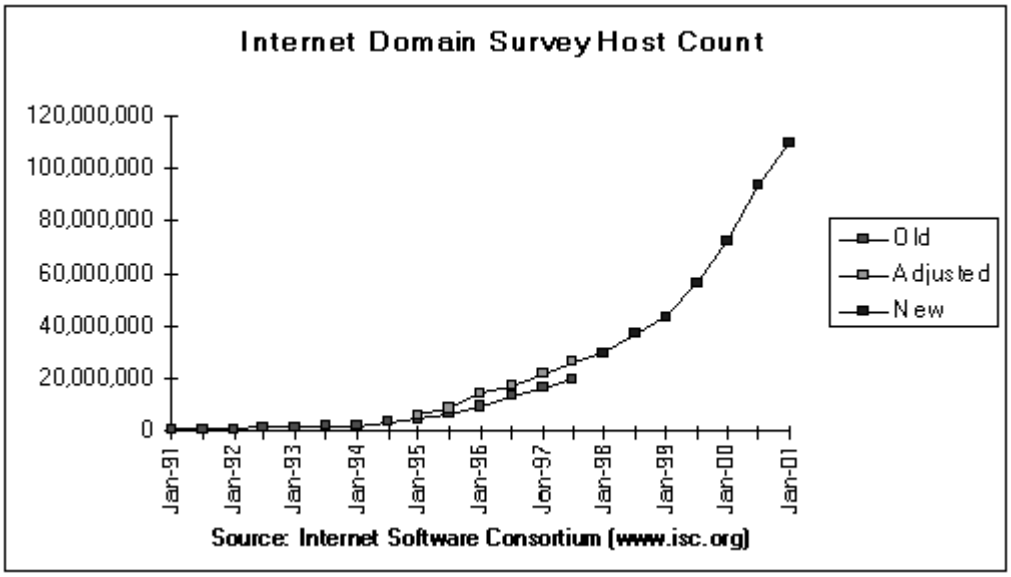


Fig. 6.2. Number of Hosts found on the Internet

to consist of approximately 2.5 billion documents in October 2000, with a rate of growth of 7.3 million pages per day (see [Lyman et al., 2000], citing a study of Cyveilance(1)). Extrapolating these numbers would mean the Web consisted of approximately 5 billion documents in October 2001. IDC expects the number to reach 8 billion in 2002.(2)

Another group of Web content is called the 'deep' Web, and it consists of specialized Web-accessible databases and dynamic Web sites. It is estimated that the information available on the deep Web is 400 to 550 times larger than the information on the 'surface' Web [Lyman et al., 2000].(3)

To provide access to this amount of information, indexes are necessary. Search engines are trying to index the information, however [Lawrence & Giles, 1999] reported in 1999 that the coverage by the largest search engine of the publicly indexable Web is 16%, and the combined coverage of all search engines is just 42%. Since then the capability of search engines has grown considerably (see Figure 6.3 for a graph showing the growth of the indexes of some popular search engines). Google recently reported indexing 1.6 billion pages - which is so far the largest index available. However, this would be just 32% of the indexable Web and just 0.01% of the 'deep' Web. Combined coverage measures are not available at this time, but given the unavoidable overlap between the search engines it is not expected to be much higher than the 42% reported by [Lawrence & Giles, 1999] in 1999.

Navigating amongst this amount of information is a challenging task, which search engines try to overcome by implementing a relevancy ordering. For instance, the search engine Google deploys a strategy called *PageRank* (cf. [Brin & Page, 1998], [Page et al., 1998], [Haveliwala, 1999]), leading to the 'most important' page for a given query. PageRank uses the Web's link structure as an

1. http://www.cyveillance.com/web/us/downloads/Sizing_the_Internet.pdf

2. see http://www.thestandard.com/article/0,1902,12329,00.html

3. The estimate might be too large - creating and maintaining the data necessary for the deep Web is a costly process [Wiederhold, 2002, personal communication].

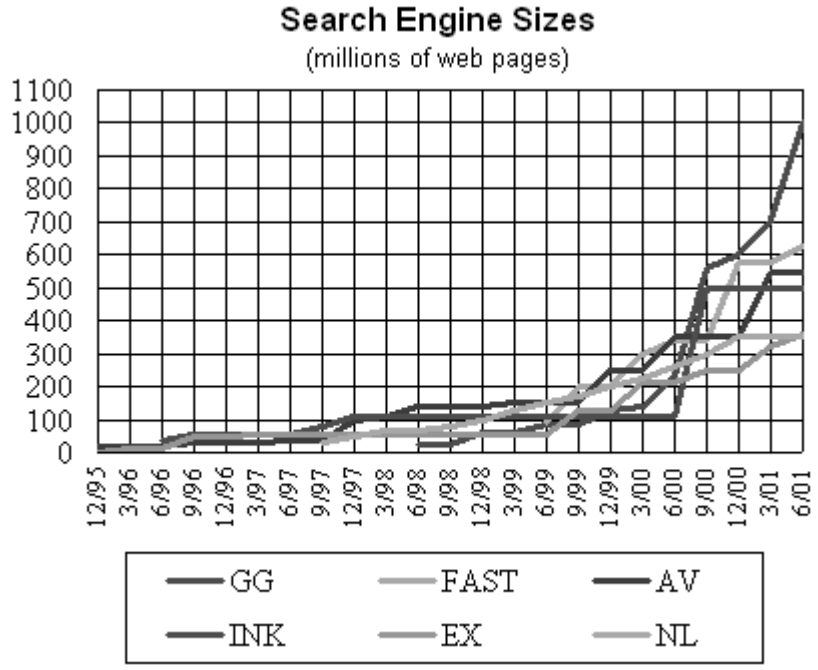


Fig. 6.3. Sizes of Popular Search Engines Over Time based on the Number of Indexed Pages (GG = Google, AV = AltaVista, INK = Inktomi, EX = Exite, NL = Northern Light).
Source: http://www.searchenginewatch.com/reports/sizes.html

indicator of an individual page's value. Google interprets a link from page A to page B as a vote, by page A, for page B. Votes cast by pages that are themselves ‘important’ weigh more heavily and help to make other pages ‘important’.

Important, high-quality sites receive a higher PageRank, which Google remembers each time it conducts a search. Google combines PageRank with text-matching techniques to find pages that are both important and relevant to a particular query. However, Google’s definition of relevance might not be always the most suitable one. As an example, the result set of the query *Stefan Decker* given to Google showed the following: the first result links to the homepage of the author, the first 19 results relate somehow to the author of this thesis. From the first 100 results, only 16 do *not* relate to the author of this thesis.(1) However, we are aware of other persons with the name *Stefan Decker*.(2) These persons are nearly impossible to find.

A more reasonable query result would return all homepages of persons with name *Stefan Decker*, sorted according to some relevance ordering. Just by using keyword-matching technology it is not possible to achieve a listing of all homepages, since background knowledge is necessary. E.g. it would be necessary to state that the query is supposed to return only homepages of persons with the

1. Queries performed in September 2001. Since Google constantly updates it’s index, changes are unavoidable.

2. A small selection of pages which mention other persons with the name *Stefan Decker* is: http://www.lstm.uni-erlangen.de/ma2/sdecker.html, http://home.t-online.de/home/StefanDecker/, http://www.schlangenfan.de/Uber_mich/uber_mich.html, http://www.invent-computing.de/english/company/team.html, http://www.bull.co.at/text_navi/frames_btm_kon_at.html, http://web.intru.de/schwalben-club/index1.htm, http://www.item.uni-bremen.de/aktuell/kohlfahrt/kohlfahrt99.html,... A recent query using a database which contains all phone numbers in Germany resulted in 246 hits for *Stefan Decker*.

name *Stefan Decker*. But a query to Google using the terms *Person Stefan Decker Homepage* does not return the desired results, since often in homepages the word *Person* does not appear (often even the word *Homepage* is not present in a homepage).

These problems not only arise on the Internet but also in corporate intranets. The possibility to ask more focused queries, resulting in exactly the right answers, would speed up search processes considerably. However, Web pages are not database tables.

An even more important aspect of intranets is the possibility of corporate portals. Portals integrate heterogeneous, distributed information sources of an enterprise at a central place. They present transactional data and other forms of personalized content, and act as a nexus of control, distribution and integration for information, content, applications and business processes among corporations and their employees, customers and business partners. Integrating different data sources and providing a uniform interface to these data sources is a major challenge. Text retrieval based search engines cannot help with this task. Other approaches are necessary.

In summary, text-based search engines have the following disadvantages.

- To retrieve a document, the keywords mentioned in the query have to be present in the document that is searched. Background knowledge capabilities, such as synonyms (e.g., human = person) or the recognition of homonyms, are typically not provided.
- Centralized search engines do not keep up with the growth of the Web. No search engine indexes more than 32% of the surface Web and less then 0.01% of the 'deep' Web.
- Current information seekers are very likely to find information about a specific topic (since on 1.6 Billion Web pages it is very likely that a few will match the expectations of the searcher). However, they are unlikely to find a specific Web page, since there is a real chance that it is not indexed by the search engine or does not have a high ranking.
- Ranking of documents according to a certain metrics might not provide the intended result and still return too many non-intended pages. This leads to the situation that although just 32% of the Web is indexed, information seekers get too many results.
- Retrieval results are not focused answers, but a collection of document URLs, where the user has to look up the desired information by him- or herself. Queries like: '*give me the research topics of all members of the research group*' are not possible, since the information is usually distributed on several pages.
- The keyword-document search metaphor is not sufficient for providing access to heterogenously distributed information sources, e.g., to provide access to a company's information databases.

6.2 The Architecture of Ontobroker

This section describes the architecture of Ontobroker, which is resolving the drawbacks of pure text retrieval deployed by other search engines.

6.2.1 The Approach

Instead of using text-based approaches to information search and retrieval to get exact answers, it is more useful to query the Web like a database. Annotation of static information is a way to provide additional information, which helps to find other information. To maximize the usefulness of the provided information and to simplify access to it means it is worthwhile to provide information in HTML format as well as in a machine-readable way.

Given published information, e.g., research interests or collaboration information, it should be possible to ask queries like: "*What are the research interests of all members of the OntoAgents research group?*" or "*Who collaborates with Stefan Decker?*", despite the fact that there is no *single* page on the Web that lists all the research interests or all the collaboration partners of Stefan Decker. The Web might contain the information, but it is usually distributed, scattered among many sources. The Ontobroker system enables exact query answering and overcomes the issues mentioned. In the following an overview of the Ontobroker system is given:

The database community has suggested semi-structured data as a data format suitable for information integration and exchange. However, to publish just the data in this format is not sufficient, since for a query-answering agent it is necessary to understand the semantics of the data.

Ontology based metadata is proposed as a means for specifying the semantics of, retrieving and using data from the Web (cf. [Boll et al., 1998], [Sheth & Klas, 1998]). It is unlikely that there will be a common ontology for the whole population of the WWW and every subject. This leads to the metaphor of a *news group* or d*omain specific ontology* [Kashyap & Shet, 1996], [Mena et al., 1998] to define the terminology for a group of people who share a common view on a specific domain.

Using ontologies for information retrieval has certain advantages over simple keyword-based access methods; an ontology provides a shared vocabulary for expressing information about the contents of (multimedia) documents. In addition, it includes axioms for specifying relationships between concepts. Such an ontology may then in turn be used to formulate semantic queries and to deliver exactly the information we are interested in. Furthermore, the axioms of the ontology provide a means for deriving information which has been specified only implicitly. These advantages come with the price of having to provide information and semantics in an explicitly formal and machine understandable manner. Since a large portion of the WWW is formulated using HTML or XML, which does not possess the means to expresses semantics, the following questions need to be answered:

- *How can ontologies and associated metadata be represented (in a sufficiently formal way) in the WWW?* This question will be answered by implementing ways to represent data, metadata, and ontologies on the Web.
- *How can information be extracted and maintained in the WWW?* Possibilities include wrappers that extract information regarding a certain ontology.
- *How can we query the data and use it for reasoning, and what inferences are possible?* Query interfaces and an inference engine that takes advantage of the defined ontologies will be implemented.

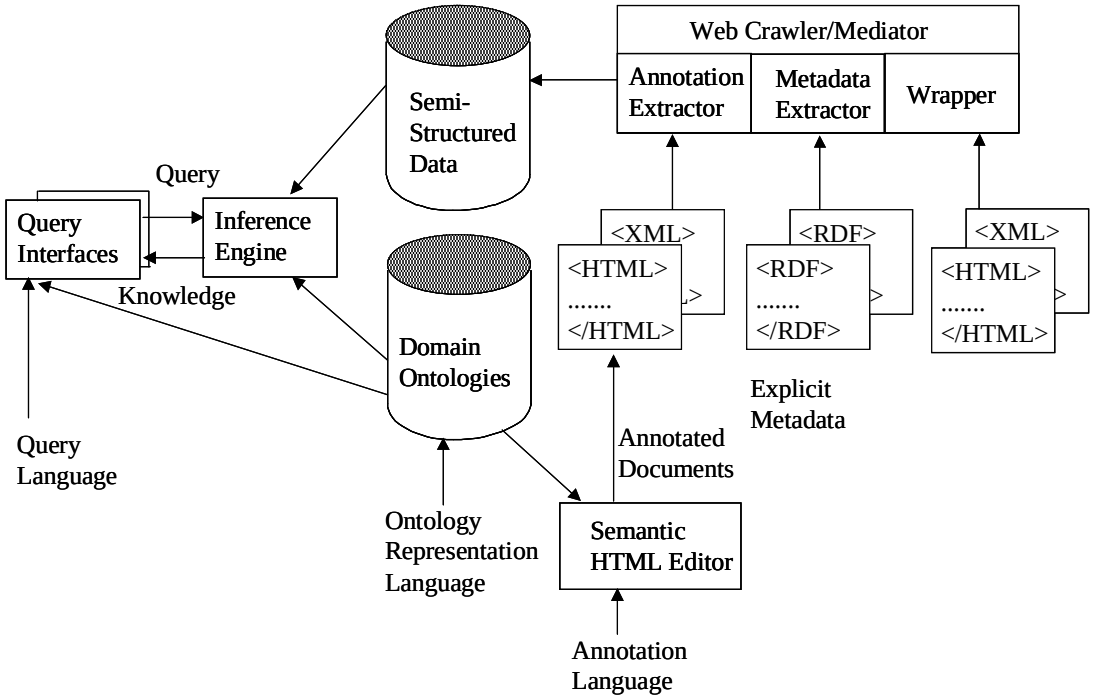


Fig. 6.4. Ontobroker Architecture

6.2.2 The System

Ontobroker (cf. [Fensel et al., 1998a], [Decker et al., 1999]) combines techniques from Logic Programming and Deductive Object-Oriented Databases, Semi-Structured Data, Hypertext Information Systems, Artificial Intelligence, and Information Visualization to improve access to heterogeneous, distributed information sources with a special focus on the World Wide Web or corporate intranets. The central idea is to provide an ontology as a way to organize distributed creation of data and to provide a query interface, which uses the ontology as a guidance to formulate queries about the data. According to the metadata classification of [Kashyap & Shet, 1996], Ontobroker deploys *domain-specific metadata* that is *content-descriptive* and utilizes a domain specific ontology. Additionally, the metadata represented and deployed is also *direct content-based*, thus allowing semantics-based access to Web information. In addition, the reasoning service provides a means for deriving information which has been specified only implicitly in the Web sources.

The architecture of Ontobroker is depicted in Figure 6.4, and consists of the following parts:

- The central part of the system are the *Ontologies*, since the ontologies influence the creation of the data as well as the inference engine, which is used for reasoning about the collected data, and the query interface, which is used for the communication with the system by the user. For expressing ontologies a *Representation and Query Language* is needed.
- The ontologies are guiding data creation and querying. Ontobroker concentrates on three data sources:
  - Annotated HTML and XML-documents, which are enriched with ontology-based annotations. For annotating HTML documents Ontobroker provides a language, HTML-A. A specialized tool, the *Ontology enhanced HTML Editor OntoPad*, supports the annotation

of hypertext with ontology based metadata by providing a user interface for generating metadata. The HTML-A approach was generalized and extended for XML documents to *XAL* (XML Annotation Language). The documents are collected by a crawler and the annotation data is extracted. The parts of Ontobroker are presented in detail in Chapter 8.

- Explicitly represented metadata in the Resource Description Framework (RDF) format.
- Wrapper-based data. Wrappers are used to connect legacy systems to the Ontobroker system.

- The *Inference Engine SiLRI* (see Chapter 7) is able to reason with the ontology and the metadata in the repository. The inference mechanism of the inference engine is based on a minimal model of a set of Horn clauses with a negation based on closed-world semantics. However, the language for representing ontologies is syntactically richer than pure Horn logic. First, ideas of [Lloyd & Topor, 1984] are used to extend Horn logic syntactically without requiring a new inference mechanism. Second, languages with richer epistemological primitives than predicate logic are provided. Frame logic [Kifer et al., 1995] is used as the representation language for ontologies. It incorporates objects, relations, attributes, classes, and subclass-of and element-of relationships.
- The *Query Interface* (see Chapter 9) enables ontology-guided composing of queries. The Query Interface hides the complexity of the Representation and Query Language and allows for generation of queries about the metadata collected from the distributed sources and enables the interactive formulation of queries while browsing the ontology and selecting the terms constituting the query.

The strength of Ontobroker is the close coupling of informal, semiformal, and formal information. The annotation languages enable the incremental generation of metadata within informal text documents, based on formal ontologies.

All parts of the architecture have been implemented and were deployed in a case study - the Knowledge Annotation Initiative of the Knowledge Acquisition Community ($(KA)^2$).

Chapter 7 SiLRI: The Representation Language and System

This chapter introduces the design and implementation of SiLRI (Simple Logic-based RDF Inference Engine), which implements the representation language used to represent ontologies in Ontobroker. The representation language is a variant of F-Logic, enhanced with the capability to provide syntactical expressive rules bodies based on full first-order logic. The inference engine of SiLRI is based on Horn logic with a closed world negation in its core. The SiLRI representation language is compiled from Horn logic with negation. The SiLRI language can be regarded as a domain specific representation language. In this chapter a method is provided for the construction of domain specific representation languages and their compilation back to available inference engines. As an example of building a new domain specific language an illustration of the combination of Chronolog is given, a language for linear temporal logic and F-Logic, which results in a language called *C-F-Logic,* capable of expressing the dynamic behaviour of objects over time. In the forthcoming chapters F-logic is used to represent the ontologies and the instance data within Ontobroker. Furthermore, an example of an ontology is provided, along with a new translation procedure realizing a variant of the Lloyd-Topor transformation.

7.1 Introduction

One of the goals of the Ontobroker system is to enable inferencing with the information from different Web sites. Inference capabilities are useful for the integration of information - especially as the ontology provides domain knowledge, which can be exploited for responding to queries. A simple example of the usefulness of inferencing is the symmetry of certain properties, e.g., `cooperatesWith`. The property `cooperatesWith` is usually a symmetric property, i.e., if person A cooperates with person B, then person B also cooperates with person A. But just from the fact that A cooperates with B it is not possible to conclude that B cooperates with A. The specification of explit domain knowledge about the `cooperatesWith` relation helps here. A flexible way to exploit the domain knowledge about `cooperatesWith` for answering queries requires that this property be modeled as symmetric within an ontology, and uses an inference engine to draw the conclusions using the declarations in the ontology and the available information from the Web. Inference engines are able to handle much more sophisticated definitions (e.g., transitivity of a relation).

A representation language for Ontologies has to fulfill the following requirements: it should be *computational* and *epistemologically* effective. Computational effectiveness guarantees that there is an effective computational environment that can reason with ontologies. Epistemological effectiveness guarantees that the modeling process is efficient and effective, since the primitives of the language allows the entities of the domain to be expressed. Different logic-based representation languages and inference engines have been described in the knowledge representation and reasoning literature. In the following sections a brief review of existing approaches will be presented and their usability for inferencing with ontologies and metadata assessed.

7.1.1 Higher-Order Logics

Higher-order logics have the greatest expressive power among all known logics. In mathematics it is often necessary to talk about properties and relations as well as about entities, since this ability is required for certain reasoning and modeling tasks. For instance, to uniquely characterize one relation as the transitive closure of another relation higher-order logics are required. By higher-order languages, logicians usually mean a language in which variables are allowed to appear in places where normally predicate and/or function symbols appear.

However, there are two different facets of higher-order languages: a higher-order syntax and a higher-order semantics. For the same higher-order syntax different definitions of the semantics are possible[(1)]. These semantics differ in the interpretation of their higher-order quantifiers. Since in the knowledge representation literature some confusion about these distinctions exists, the matter is investigated here in more depth.

- One possible interpretation of second-order quantifiers is the set of all relations, i.e., all subsets of the first-order universe (also called the *power set* of the universe). This interpretation is usually taken for second order logics by mathematicians. Unfortunately, no logical calculus can be complete with respect to this semantics, since the power set of a countable universe is uncountable. Many properties of the integers are uncountable - which cannot be checked by any procedure. For inference engines this results in unpleasant behavior: there are true statements, which are unprovable (This is a consequence of Goedel's Incompleteness Theorem [Kerber, 1994]), and thus these engines are not very useful in situations in which completeness is required.
- Another possible interpretation of second order quantifiers is the set of all definable properties, i.e., those which could be defined using an expression of the language (*Henkin semantics for HOL*). This is only a subset of the power set, and not further investigated here.
- Yet another possible interpretation of second-order quantifiers is the set of all named properties. The named properties are the properties which have been assigned to predicate names by the interpretation. This is an even smaller subset of the power set. The resulting logic is equivalent to First-Order Logic in terms of semantics. The translation is also possible for higher-order logic in general (cf. [Goedel, 1965]).

[Chen et al., 1993] exploit the different semantics to provide a first-order semantics for a higher-order logic-programming language, resulting in a syntactically rich language which is not restricted by the computational problems of higher order logics.

Higher-order languages with higher-order semantics are often not acceptable as knowledge representation languages, since no complete inference procedures exist. Fortunately often a first order semantics is sufficient for real world applications - the resulting language provides the syntactic expressiveness of higher-order-logics, while sound and complete inference procedures can still be used.

1. See http://suo.ieee.org/email/msg00345.html for an email from Pat Hayes explaining the different semantics to the Standard Upper Ontology Mailing list.

7.1.2 Full First-Order Logic-Based Inference Engines

Classical full first-order logic (FOL) (cf. [Ebbinghaus, 1994], [Enderton, 1992]) is one of the most well-known logics for knowledge representation tasks. Inferencing with FOL-axioms requires a fully-fledged automated theorem prover (cf. SETHEO [Ibens, 1998], OTTER,[1] SPASS [Weidenbach, 1997]). For FOL sound and complete inference procedures exist, but FOL is still semi-decidable and inferencing is computationally not tractable for large amounts of data and axioms (see [Goubault, 1994] for an in-depth complexity analysis). This is reflected by TPTP[2] (*Thousands of Problems for Theorem Prover*, see [Sutcliffe & Suttner, 1998]), a collection of test-cases for theorem provers, used by the automated theorem proving community. The TPTP library contains many small, tricky problems, and theorem provers trying to solve these problems often fail [Sutcliffe, 2001].

In an environment like the Web, these programs would not scale up for handling large amounts of data. Besides, using full first-order theorem prover would require the maintenance of consistency throughout the Web, which is considered to be impossible, since the Web accommodates many different viewpoints and opinions.

An alternative is to use subsets of FOL with more suitable properties.

7.1.3 Description Logic

Description Logics (DLs) allow specifying a terminological hierarchy using a restricted set of first order formulae. They usually have good computational properties (often decidable and tractable, and often with equally good average computational complexity), but the inference services restricted to classification and subsumption. Given a set of formulae describing classes, the classifier associated with a certain description logic will place them in a hierarchy, and given an instance description, the classifier will determine the most specific classes to which the particular instance belongs. From a modeling point of view, DL corresponds to monadic/dyadic FOL, restricted to formulae with three variables (cf. [Borgida, 1996]). This restriction suggests that modeling is syntactically bound. Available modern systems include FACT [Horrocks et al., 1999] and RACER [Haarslev & Möller, 2001].

Description Logics are strong in their intentional specification of ontologies (cf. [Brachman et al., 1991]), but don’t provide solid support for rules and other kind of reasoning, especially for a large volume of instance data.

Another possibility for KR tasks is languages based on Horn-logic, which is another fragment of FOL.

7.1.4 Logic Programming and Deductive Databases

Horn-logic and Datalog (Horn-logic only with 0-ary function symbols) was studied in the area of deductive databases and logic programming and a number of efficient evaluation strategies are available. Evaluation strategies can be distinguished by bottom-up and top-down variations. Top-down evaluation is usually used by Prolog systems (e.g., SWI-Prolog[3]). However, top-down

1. see http://www-unix.mcs.anl.gov/AR/otter/#doc
2. http://www.cs.miami.edu/~tptp/
3. http://www.swi.psy.uva.nl/projects/SWI-Prolog/

evaluation has certain disadvantages; it emphasizes the *programming aspect*, since a user of the system always needs to keep the evaluation strategy in mind. For instance, the prolog clause '`p(X,Z) :- p(X,Y), p(Y,Z)`' causes a Prolog system to loop.

Bottom-up evaluation strategies known from deductive databases (e.g., naive or semi-naive evaluation) do better; they always terminate when the set of formulae has a finite model. Goal directed bottom-up evaluation strategies (e.g., based on tabling or dynamic filtering) terminate when the model of the formulae necessary to answer the query is finite.

Logic programming has been suggested as a means to Knowledge Representation (c.f. [Baral & Gelfond, 1994], [Brewka & Dix, 1998], [Baader et al., 1991]). The research has focused on extending logic programs with capabilities for non-monotonic reasoning (e.g., default negation, explicit negation, preferences) and disjunction [Brewka & Dix, 1998].

Since Horn-logic based KR is more practical than KR based on FOL and the present interest is in other inference services than just classification and subsumption, the current approach uses Horn-logic. The notation used in [Lloyd, 1987] and [Brewka & Dix, 1998] is adopted.

However, for knowledge representation tasks, it is not only the underlying logical formalism that is important, but also the *epistemological primitives* that help to represent e.g., a domain ontology. Results in object-oriented deductive databases (cf. F-Logic [Kifer et al., 1995]) provide representation primitives that can be used for knowledge representation tasks. F-Logic aims at the representation of objects, classes and relationships between objects.

7.2 F-Logic in SiLRI

Frame Logic (abbr. F-logic) was introduced in [Kifer et al., 1995] as:

> *"A novel formalism that accounts in a clean and declarative fashion for most of the structural aspects of object oriented and framebased languages. These features include object identity, complex objects, inheritance, polymorphic types, query methods, encapsulation, and others. In a sense F-logic stands in the same relationship to the object oriented paradigm as classical predicate calculus stands to relational programming. F-logic has a model theoretic semantics and a sound and complete resolution-based proof theory. A small number of fundamental concepts that come from object-oriented programming have direct representation in F-logic; other, secondary aspects of this paradigm are easily modeled as well."*

Usually, ontologies are defined via concepts or classes, is-a relationships, attributes, further relationships, and axioms. Therefore, an adequate language for defining the ontology has to provide modeling primitives for these concepts. Frame-Logic [Kifer et al., 1995] already provides such modeling primitives and integrates them into a logical framework providing a Horn logic subset. Furthermore, in contrast to most Description Logics, expressing the ontology in Frame-Logic allows queries that directly use parts of the ontology as first class citizens. That is, not only instances and their values but also concept and attribute names can be provided as answers via variable substitutions.

F-logic has clear roots in object-oriented databases and logic programming. Existing systems (e.g., FLORID [Frohn et al., 1997] and FLORA [Yang & Kifer, 2000]) are tailored towards object-oriented logic programming, incorporating metaphors from PROLOG.

However, defining ontologies is a modeling and specification task, and a programming task. Defining an ontology is not programming, and the knowledge engineer should neither worry about the particular order in which literals are executed nor about other idiosyncrasies usually present in a progamming language.

So starting from the original F-logic specification, F-logic was modified, so that it supported the modeling of ontologies. In the next section the syntax of the SiLRI[(1)]-version of F-Logic is introduced.

7.2.1 Syntax of F-Logic in SiLRI[(2)]

For a detailed description of F-logic [Kifer et al., 1995] should be consulted. F-logic is described here to the extend necessary for following sections.

From a syntactic point of view F-logic is a superset of first-order logic. To define the language we introduce the F-logic alphabet consisting of the set of predicate symbols ***P***, function symbols ***F*** and variables ***V***. An *id-term* is a first-order term composed of function symbols in ***F*** and variables in ***V***. The set of all ground id-terms is denoted by ***U(F)***, also called Herbrand Universe, which is the set of all objects.

Let o, m, r, c, c_1, c_2, t, V_i, T_j id-terms. A *molecule* in F-logic is one of the following expressions:

- $o{:}c$ (*instanceOf assertion*) (Object o is an instance of the class c).
- $c_1{::}c_2$ (*subclassOf assertion*) (Class c_1 is a subclass of the class c_2).
- $o[m@v_n,...,v_n \rightarrow r]$ (*functional method application*) This expression denotes that the result of the application of the single-valued method m with the arguments $v_1, \ldots, v_n$ on the object o is r. n (the number of parameters) might be zero - in this case the '@' sign is omitted. Instead of the simple arrow '$\rightarrow$' the symbol '*$\rightarrow$' might be used. In the latter case the result is said to be *inheritable*, and the value acts as a default value.
- $o[m@v_n,...,v_n \twoheadrightarrow \{r_1,...,r_m\}]$ (*set-valued method application)* This expression denotes that the result of the application of the set-valued method m with the arguments $v_1, \ldots, v_n$ on the object o is the set $\{r_1, \ldots, r_m\}$. n (the number of parameters) might be zero - in this case the '@' sign is omitted. If m=1 it is also permissible to omit the parentheses in the result specification. Instead of the simple arrow '$\twoheadrightarrow$' the symbol '*$\twoheadrightarrow$' might be used. In the latter case the result is said to be *inheritable*.
- $c[m@V_1,...,V_n \Rightarrow (T_1,...,T_m)]$ (*single-valued signature-molecule)* In the signature molecue the T_j are id-terms that represent types of the results returned by the method m when m is invoked on an object of class c with arguments of types V_i. The result of the method must be an instance of all the T_j. n might be zero - in this case the parentheses may be omitted.
- $c[m@V_1,...,V_n \Rightarrow\!\!> (T_1,...,T_m)]$ (*set-valued signature-molecule)* This case is analoguous to the above.
- Inside the square brackets it is permissible to have multiple method application, separated by an ";". Every id-term can itself be an object, upon which other objects are applied.
- $p(F_1,...,F_n)$ (*predicate molecule)* A predicate molecule (P-molecule) corresponds to an atom in First-Order predicate logic. The predicate symbol p is an element of P.

1. SiLRI is an acronym for *Simple Logic-based RDF Interpreter*
2. For a complete EBNF specification of the syntax of SiLRI's F-Logic variant, see appendix A.

Example: The following expression defines a complex object with id ***r1*** as an instance of the class ***Researcher*** with the two attributes ***authorOf*** and ***cooperatesWith***. Both attributes are set-valued, but only one value is defined. The value of the ***cooperatesWith*** attribute is a complex object with id ***r2*** and an attrribute ***authorOf***.

$$r1{:}Researcher[authorOf \twoheadrightarrow article1; cooperatesWith \twoheadrightarrow r2[authorOf \twoheadrightarrow b1]]$$

The complex objects expressions can be flattened. Then the expression above is equivalent to:

$$r1{:}Researcher \wedge r1[authorOf \twoheadrightarrow article1] \wedge r1[cooperatesWith \twoheadrightarrow r2] \wedge r2[authorOf \twoheadrightarrow b1].$$

From the molecules, complex expressions may be built. We distinguish between the following complex expressions: *facts*, *rules*, *double rules*, and *queries*. Facts are ground molecules. A rule consists of a head, the implication sign <-, and the body. The head is a conjunction of elementary expressions (connected using AND). The body is a complex formula built from elementary expressions and the usual predicate logic connectives (implies: ->, implied by: <-, equivalent: <->, AND, OR, and NOT. Variables are introduced in front of the head (with a FORALL-quantifier) or anywhere in the body (using EXISTS and FORALL-quantifiers). A double rule is an expression of the form: head <-> body, where the head and body are conjunctions of molecules. So full first-order formulae may appear as bodies in F-logic rules and require variables to be explicitly introduced - in contrast to other available F-logics dialects. So SiLRI is also (syntactically) more expressive than Prolog, since Prolog does not allow expressive rule bodies**.** Since negation in rules is not part of Horn logic, appropriate semantics for the negation need to be found. Semantic issues will be discussed in the next section.

Ontologies, as introduced in Section 2.2.2, consist mainly of three parts:

- The concept hierarchy defines the subclass relationship between different classes,
- Attribute definitions for classes, and
- A set of rules defining relationships between different concepts and attributes.

Table 7.1 shows an example for each of the thee parts of an ontology in SiLRI's F-logic variant - a simple class hierarchy, where classes like `Person` and `Researcher` are defined, attribute definitions (like `name` or `email`), and axioms defining, e.g., the symmetry of `cooperatesWith` for instances of class `Researcher`.

7.3 Semantics of F-Logic

[Kifer et al., 1995] give a *direct semantic characterization* of F-Logic in terms of *F-structures*. However, for Ontobroker this approach was discarded in favor of a translation semantics, i.e., a translation from F-logic to logic programs has been provided. Then the usual semantic characterizations of logic programs (cf. [Brewka & Dix, 1999]) are directly applicable. Since more expressive rule bodies than are usually available in logic programming systems have been allowed, a multi-step translation is applied here. Figure 7.1 presents the different steps of the translation procedure: initially, the SiLRI-F-Logic specification is reduced to a general logic program (cf. [Lloyd, 1987]). In the next step a Lloyd-Topor transformation [Lloyd & Topor, 1984] is applied, which results in a normal logic program. The end result can be interpreted by logic-programming systems. In the following the two steps are investigated in more detail. In Ontobroker only the F-logic translation is used. In the following section the F-logic translation is presented and generalized to the translation approach by providing a methodology for building domain-specific knowledge

Class Hierarchy

```
Object[].
   Person :: Object.
      Employee :: Person.
         AcademicStaff :: Employee.
            Researcher :: AcademicStaff.
               PhDStudent :: Researcher.
      Student :: Person.
         PhDStudent :: Student.
   Publication :: Object.
      Book :: Publication.
      Article :: Publication.
         TechnicalReport :: Article.
         JournalArticle :: Article.
      Journal :: Publication.
```

Attribute Declarations

```
Person[name => STRING; email => STRING; phone => STRING;
         publication => Publication; editor => Book].
Employee[employeeNo => STRING].
AcademicStaff[supervises => PhDStudent].
Researcher[cooperatesWith => Researcher].
Student[studentID => NUMBER].
PhDStudent[supervisor => AcademicStaff].
Publication[author => Person; title => STRING;
         year => NUMBER; abstract => STRING].
Book[editor => Person].
TechnicalReport[series => NUMBER; number => NUMBER].
JournalArticle[journal => Journal; firstPage => NUMBER;
         lastPage => NUMBER].
Journal[editor => Person; volume => NUMBER; number => NUMBER;
         containsArticle => JournalArticle].
```

Axiom Declarations

```
FORALL Pers1, Pers2
      Pers1:Researcher[cooperatesWith -> Pers2] <->
            Pers2:Researcher[cooperatesWith = Pers1].
FORALL Pers1, Publ1
      Publ1:Publication[author -> Pers1] <->
            Pers1:Person[publication -> Publ1].
FORALL Pers1, Publ1
      Publ1:Book[editor -> Pers1] <->
            Pers1:Person[editor -> Publ1].
FORALL Pers1, Pers2
      Pers1:PhDStudent[supervisor -> Pers2] <->
            Pers2:AcademicStaff[supervises = Pers1].
FORALL Publ1, Publ2
      Publ2:Journal[containsArticle -> Publ1] <->
            Publ1:JournalArticle[journal -> Publ2].
```

Tab. 7.1. Part of $(KA)^2$-Ontology

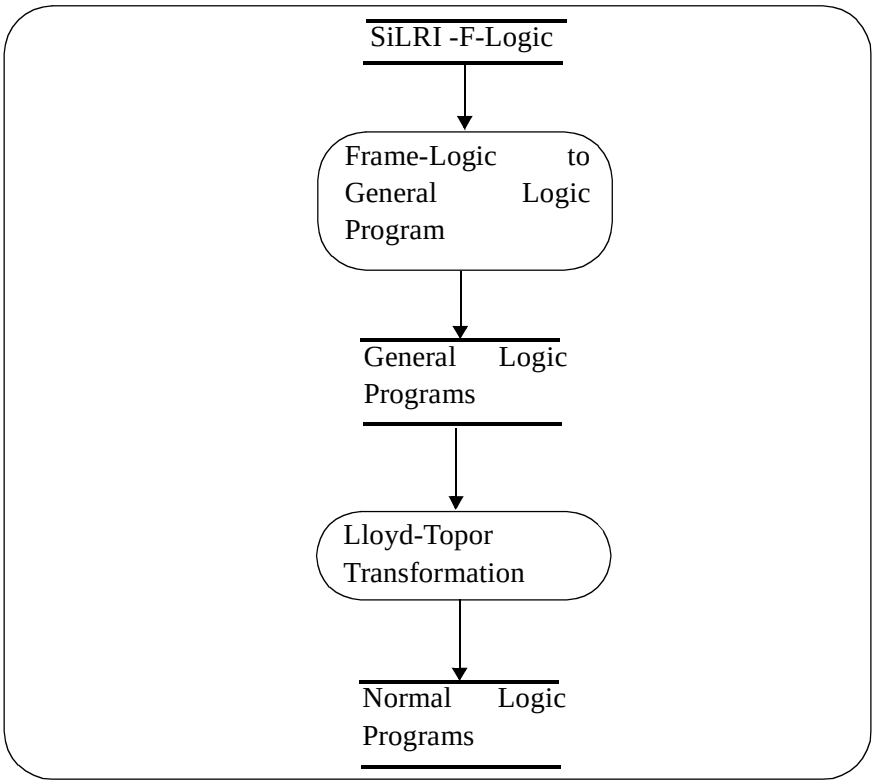


Fig. 7.1. Multi-step translation of SiLRI-F-Logic

representation languages on top of deductive databases. The presented approach helps transfer the Ontobroker approach to inferencing and reasoning into other areas. As an example, a specific language is constructed that supports reasoning for changing objects.

7.3.1 Constructing Domain-Specific Logic Languages[1]

A domain-specific language (DSL) can be viewed as a language dedicated to a particular domain or problem. It provides appropriate built-in abstractions and notations. Domain specific ontologies and domain specific languages, while aiming at similar goals, are different: a domain specific ontology consists of a class hierarchy, attribute definitions and axioms, and the means to define instances of classes defined in the ontology.

A domain-specific language (cf. [Wille & Ramming, 1999]) is usually unrelated to classes and instances. Because of appropriate abstractions, notations and declarative formulations, a domain specific language program is more concise and readable than a generic language. Hence, development time is shortened and maintenance is improved. In this section the problem of constructing domain-specific logic languages is examined.

Logic-based knowledge and data representation and specification has a long tradition in AI and databases (cf. [Hayes, 1977], [Fensel, 1995], [Bringsjord, 1998]), so for knowledge acquisition and engineering tasks many representation formalisms are available. However, most do not support the elicitation and representation of domain specific knowledge. Using the syntax of First Order Predicate Logic (FOL) for knowledge representation tasks is comparable to using a Turing Machine

1. This section is an extended version of [Decker, 1998].

for programming tasks: it is an adequate formalism for studying theoretical properties, but does not support real-life tasks. Of course, a Turing Machine is powerful enough from a *principle* point of view, but not from a *practical* one: finding the right encoding is often too expensive. E.g., human programmers tend to be much more productive in high-level programming languages like Java, although the expressive power is exactly the same as that of a 6502 assembler.

One of the reasons why finding the right encoding is expensive is that the FOL does not support thinking in domain terms. Instead, the domain terms must be translated into the terms of the representation language (in the case of First-Order Logic - relations). To establish a communication between a knowledge engineer and a domain expert, the representation must always be translated back to the domain terms that the expert is familiar with. The translation problem is fortunately a matter of the syntax of the representation language. Therefore, similarly to the development of high-level programing languages, the syntax of first-order logic is used here as a low-level language for building domain-specific declarative languages (DSDLs). Domain specific representation languages are compiled into FOL.

Particularly for Horn-logic, with its closed world semantics, a number of efficient systems exist (e.g. CORAL [Ramakrishnan et al., 1994], XSB [Rao et al., 1997], ADITI [Vaghani et al., 1996], ConceptBase [Jarke et al., 1997], LogicBase [Hammer et al., 1997] among others), which can be used as a *virtual machine* for inferences and execution, obtaining formal and operational semantics for domain-specific languages free of charge. The expressive power of the logics used in the above-mentioned systems is well understood and several semantics are defined (for an overview see e.g., [Brewka & Dix, 1999], [Lloyd, 1987]). Describing this approach using Brachmans stratification for knowledge representations [Brachman, 1979], the *implementation level* consists of data structures and algorithms used in the deductive database or logic programming system, the *logic level* is Horn-logic with negation, and the *epistemological level* is realized through adequate domain dependent representation primitives by compilation at the lower levels. The translation approach for logic languages is similar to approaches in compiler construction: usually an initial language is not compiled directly into a target language, but through several intermediate stages and languages. The intermediate stages help to bridge the conceptual gap between the initial language and the target language (e.g., the first C++ compiler was nothing more than a preprocessor for C programs). The direct implementation of the semantics used in a specific domain leads to an inflexible system: a small change in the input language results in changes in the whole inference system.

How this idea is exploited in knowledge representation is shown below, leading to special purpose knowledge acquisition, representation and inference systems that are easy to build, to maintain and to adjust to different tasks.

A domain-specific declarative language should support the elicitation and representation of knowledge of a particular domain. So, the domain to be represented determines the semantics of the language, and for compilation reasons the domain concepts have to be translated into the usual FOL language (see Figure 7.2 for an illustration).

The following methodology is used:

1. The first step is *clarification* of the concepts used in the specific domain. For object oriented systems such concepts can be e.g., classes, instances and methods or special individuals.
2. The next step is the *mapping* of the identified concepts to the usual first order semantics.The concepts of the domain have to be expressed through predicates, functions, constants or formulae.

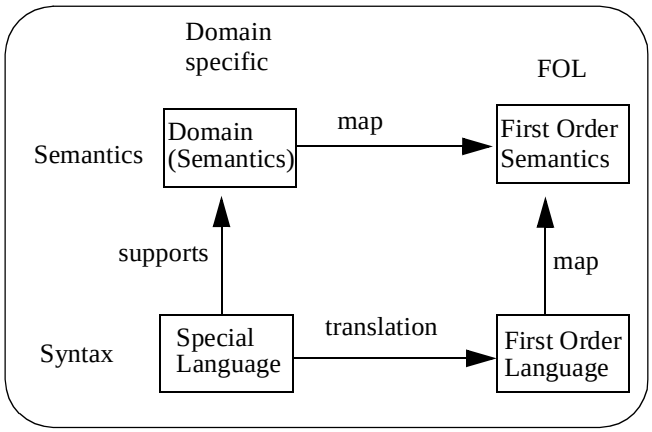


Fig. 7.2. Relationship between DSDL and FOL

3. The most important and most difficult step is to *identify syntactic constructs* that support the elicitation and representation of knowledge in an appropriate way. Several possibilities exist: special operators can be identified, such that relations from the domain are now expressed with operators. If these operators have built-in semantics, these semantics can be captured with appropriate axioms. Other domain concepts can be captured via special function symbols. Some useful principles are illustrated later on.
4. Using the identified mapping from the domain-specific semantics to the FOL semantics, an appropriate translation from the DSDL syntax to the FOL syntax can be defined. What was identified as a relation in the domain analysis is translated into a predicate. Special constants from the domain language are translated into special constants in the target language. This translation is needed in both directions: the set of formulae is initially (before reasoning takes place) translated from the domain syntax to the syntax of the deductive database or logic programming system, and the inference results (usually variable substitutions) have to be translated back into the domain syntax.

The general approach is illustrated through a reengineering of two special representation languages: Frame-Logic, useful for modeling in an object oriented style, and Chronolog (cf. e.g., [Liu & Orgun, 1996], [Orgun & Ma, 1996]), that realizes Linear Temporal Logic and allows for modeling of states and state changes. Although both languages have a well defined model and proof theory and implemented inference engines exist, inference engines for these languages can also be obtained, as shown, via a translation approach and the use of one of the inference engines for standard horn logic. Therefore, the languages can be adapted to different purposes, they can be combined, and techniques available for deductive databases are thus directly applicable.

7.3.1.1 Frame-Logic

F-logic has already been introduced in Section 7.2.1. In this section a reingineering of F-logic is performed by starting from the domain-specific notions (e.g., subclass), defining FOL equivalents and finally a supporting syntax.

The process begins with an analysis of the parts of the F-logic semantic structure. Therefore, translations for the molecular expressions of F-Logic are given. This is followed by an analysis of some of the most important semantic concepts used in F-logic:

- *Subclassing*: is an important concept in object-oriented languages (see [Stefik & Bobrow, 1986]). Properties of subclassing include transitivity and reflexivity. Subclassing is captured by a relationship.
- *Class membership*: allows for identification of which element of the universe of discourse is a member (or instance) of another element. Again, class membership is a relation within the FOL universe.
- *Methods*: identifies when an element of the universe of discourse can be interpreted as a method of other elements and what result is returned on invocation of the method. A method is a mapping between a set of parameters and a result, and may be best represented by a relation, which associates objects, parameters, and results.
- *Method definitions*: identifies which classes understand which methods, and what the class of the result is. This concept is again a relation.
- *Method inheritance*: in object oriented systems, methods defined in classes are inherited by the instances of the classes. Thus, a method definition is usable for all instances of a class, but it is still possible to overwrite it. This aspect of F-logic shows some of the limitations of the current approach and will be discussed later on.

Object Oriented		**First Order**
C::D	$\Rightarrow$	$sub(C, D)$
O:C	$\Rightarrow$	$instance(O, C)$
$O[M$->>V]	$\Rightarrow$	method(O,M,V)
$C[M$=>>D]	$\Rightarrow$	methodtype(C,M,D)
O:$C[M$->>V:D]	$\Rightarrow$	$instance(O, C) \wedge$ method(O,M,V) $\wedge$ $instance(V, D)$
Axioms:		
Reflexivity:	$\forall X \quad sub(X, X)$	
Transitivity I:	$\forall X, Y, Z \quad sub(X, Z) \leftarrow sub(X, Y) \wedge sub(Y, Z)$	
Transitivity II:	$\forall X, Y, Z \quad instance(X, Z) \leftarrow instance(X, Y) \wedge sub(Y, Z)$	
Acyclicity:	$\forall X, Y \quad (X = Y) \leftarrow sub(X, Y) \wedge sub(Y, Z)$	

Tab. 7.2. Translation Schema for F-logic

The next step is to map these object-oriented semantic concepts to first-order relations: In these cases all concepts are mapped to relations of the target semantics. Subclasses, class membership and methods are identified through special relations of the usual first order interpretations. Table 7.2 contains a list of syntactic translations and necessary axioms. Translation not only has to adopt the relational structure of the concepts, but also the domain specific properties of the domain, e.g. transitivity of the subclass relationship. Therefore, it is not only necessary to do the syntactic translation, but also to introduce the axioms that capture the domain specific properties of the constructs. The syntactic translation schemes and some of the axioms are also shown in table 7.2.

There are, of course, several possibilities to support the identified concepts syntactically in a domain specific language. The developers of F-logic have adopted notations that are similar to other languages: ***C :: D*** for subclassing, meaning that class ***C*** is a subclass of class ***D***. ***O : C*** for class

membership, meaning that ***O*** is an instance of class ***C***. ***O[M->V]*** for method invocation, meaning that the instance ***O*** has a method ***M*** with value ***V***, and ***C[M=>D]*** for method definitions, meaning that all instances of the class ***C*** understand the method ***M*** and the class of the result returned is ***D***. Furthermore, to allow for a more flexible handling, several syntactic constructs can be combined, e.g., ***O:C[M->>V:D]*** is an allowable construct. An example F-logic specification, defining a simple traffic light, is depicted in Figure 7.3.

```
Sign :: Object. Color :: Object.
Traffic_Light::Sign[status=>Color].
green : Color.
amber : Color.
red : Color.
```

Fig. 7.3. Part of a F-logic Traffic Specification

Here a part of a traffic specification is shown, defining three classes (Sign, Traffic_Light and Color), one method (status) of the class Traffic_Light and three instances (the colors). It is possible to specify rules about the objects, but it is very cumbersome to specify the dynamic behavior and change of objects (e.g., the change of the color of the traffic light) without any possibilities to express the change of states. The rules example shown translates only a part of F-logic, but most of the remaining rules can be handled in a similar way. In Figure 7.4 the translation is depicted.

```
sub(Sign,Object). sub(Color,Object).
sub(Traffic_Light,Sign).
methodtype(Traffic_Light,status,Color).
instance(green,Color).
instance(amber,Color).
instance(red,Color).
```

Fig. 7.4. Translation of the F-logic Traffic Specification

7.3.1.2 Chronolog

Temporal logic has been widely used as a formalism for program specification and verification. Chronolog [Orgun, 1996] is a language based on linear-time temporal logic. Semantic concepts are the initial state and the next state. The semantic structures of Chronolog are described as temporal Herbrand models, such that every element of the temporal Herbrand model is true in exactly one moment in time.

The example depicted in Figure 7.5 is taken from [Orgun & Ma, 1996]. The program specifies the simulation of a traffic light. At the initial stage the color of the traffic light is green. The clauses then define the value of the color in the next states. The series goes green, amber, red, green, amber, red,

green and so on. This example can be queried for the color of the light in the n-th point of time. Note that in Figure 7.5 no objects are defined, since Chronolog provides no means to specify objects and relationships between objects, as is possible in F-logic..

```
first light(green).
next light(amber) <- light(green).
next light(red) <- light(amber).
next light(green) <- light(red).
```

Fig. 7.5. Chronolog Example Specifying the Dynamics of a Traffic Light (taken from [Orgun & Ma, 1996])

The semantic analysis shows that every relation from the temporal Herbrand model is extended with an extra argument. This argument contains a counter that stands for the moment in time in which this fact is valid. The syntactic support for temporal specification is achieved through two temporal operators: *first* and *next*, which refer to the initial and the next moment in time respectively. An atom may have a prefix of the form *first next*, *next*, or no prefix at all, meaning that the atom is valid in all moments of time.

The syntactic translation is done as follows: every atom is extended with an extra argument. This extra argument is used as a time counter and the first-next sequence is coded into the extra argument according to table 7.3: the first operator is associated with a constant **O**, symbolizing the initial point of time. The unary function symbol ***s*** (for successor) is used to encode the next operator, e.g. the term ***s(0)*** is associated with the sequence *first next.*

Chronolog	First Order Logic
first light(green).	light(green,0).
next light(amber) <- light(green).	light(amber,s(X)) <- light(green,X).
next light(red) <- light(amber).	light(red,s(X)) <- light(amber,X).
next light(green) <- light(red).	light(green,s(X)) <- light(red,X).

Tab. 7.3. Translation Schema for the Chronolog Example

7.3.1.3 C-F-Logic: An Amalgamation of F-logic and Chronolog

Taking these two translation tables as a starting point, both approaches - F-logic and Chronolog - can be amalgamated. The resulting language C-F-Logic, allows declarative and executable specifications about objects changing their properties over time. The example in Figure 7.6 combines the two specifications about the traffic domain (Figure 7.3 and Figure 7.5). The result is a language similar to StateLog [May et al., 1997]. However, it is not necessary to build a special purpose inference engine. Instead, a translation is made into the usual deductive database formalisms. For performing this kind of translation, the different translation tables have to be

combined. The combination is effected by a two step process: in the first step the temporal

```
Sign :: Object. Color :: Object.
Traffic_Light :: Sign[status=>Color].
green : Color. amber : Color.
red : Color.
tl : Traffic_Light.
first tl[status->green].
next tl[status->amber]<-tl[status->green].
next tl[status->red]<-tl[status->amber].
next tl[status->green]<-tl[status->red].
```

Fig. 7.6. Combining F-logic and Chronolog

annotations are ignored and only the translations, as given in table 7.2, are performed. The result is an ordinary Chronolog program, so that the second translation table (Figure 7.3) is applicable. The result of the final translation (without the axioms specifying the behavior) is depicted in Figure 7.7

```
sub(Sign,Object,X).
sub(Color,Object,X).
sub(Traffic_Light,Sign,X).
methodtype(Traffic_Light,status,Color,X).
instance(green,Color,X).
instance(amber,Color,X).
instance(red,Color,X).
instance(tl, Traffic_Light,X).
method(tl,status,green,0).
method(tl,status,amber,s(X))<-
method(tl,status,green,X).
method(tl,status,red,s(X))<- method(tl,status,amber,X).
method(tl,status,green,s(X))<- method(tl,status,red,X).
```

Fig. 7.7. Translation Of The Combined Specification

Interdependencies

There are several interdependencies between the F-logic part and the Chronolog part in the combined language. Firstly, a complex F-logic molecule may be translated into a set of FOL-atoms (see Figure 7.2 for an example). In this case the temporal annotations are distributed over the atoms.

A second dependency is the occurrence of the well known *frame problem* [McCarthy & Hayes, 1969]*:* introducing the notion of objects (frames) and states one not only has to specify which properties of an object change over time, but also which properties don't change. A desirable behaviour is that properties do not change if there is no explicit specification that says otherwise.

This frame problem can be solved using the non-monotonic behavior of negation in logic programming: for every predicate ***p*** occurring in the translation of F-logic (see Figure 7.2) two new predicate symbols *direct_p* and *inherited_p* are introduced. In the bodies of all inference rules ***p*** is substituted by *inherited_p* and in the heads of all inference rules ***p*** is substituted by *direct_p*. For every predicate ***p*** additional inferences rules as shown in Figure 7.8 are added to the final result of the compilation process. These rules handle the frame problem as follows: if there is a direct

specification of the arguments for ***p*** in a certain moment of time, only the first rule is applicable - the direct specification of the values take precedence. If there is not a direct specification of the arguments for ***p***, the second rule is applicable. Then the value is copied from the previous point of time, given that no direct specification of the value is present The rules take care that the values of the attributes are 'inherited' over time....

$$\forall \vec{X}, T \quad \text{inherited\_p}(\vec{X}, T) \leftarrow \text{direct\_p}(\vec{X}, T)$$

$$\forall \vec{X}, T \quad \text{inherited\_p}(\vec{X}, s(T)) \leftarrow \text{inherited\_p}(\vec{X}, T) \wedge \neg\text{direct\_p}(\vec{X}, s(T))$$

Fig. 7.8. Rules for handling the Frame Problem

7.3.1.4 Assessment of the Approach

It is well known that definite clauses with function symbols are a *turing-complete* representation formalism. So every computable function can in principle be compiled into definite clauses. This means that we can always build a meta-interpreter that computes the desired semantics for an appropriate compilation result of the domain. However, this is usually not a good idea in practice: meta-interpretation introduces an additional level of inefficiency. So a direct compilation is favored, without the introduction of meta-interpretation. The limits of a direct compilation of some domain specific languages are exactly defined by the semantic properties of the target language. In this case the language is Horn-logic with negation. The semantic properties of the domain have to be compatible with the semantic properties of clauses. In practice this means semantic concepts of a domain that need the computation of e.g., multiple fixpoints are not directly compilable into Horn clauses.

An example is the multiple inheritance part in F-logic. Multiple inheritance requires the computation of multiple fixpoints (cf. [Kandzia, 1997] for further details). Nonmultiple inheritance can be handled with rules similar to those depicted in Figure 7.8 The only change is the omission of the temporal argument.

Of course, other authors have noticed that their logical systems are often easy to translate into first order logic. For instance Kifer, Lausen, and Wu argue in [Kifer et al., 1995] in favor of a direct semantics for F-logic: "*The syntax of a programming language is of paramount importance, as it shapes the way programmers approach and solve problems. However, syntax without semantics is not conducive to programming. Third, a direct semantics for a logic suggests ways of defining proof theories tailored to that logic. Such proof theories are likely to be a better basis for implementation than the general-purpose classical proof theory.*" While the first argument is not disputed, the second and third are highly debatable: if using a modeling language requires the understanding of its formal semantics, then the language is not useful. How many Cobol, C, C++, Java or even Prolog programmers understand a formal semantics of the language they use? In spite of this, they are performing many useful tasks with the languages. Furthermore, building a specialized inference engine for a special semantics requires an extraordinary effort. It may be true that when optimizing this special inference engine, the performance of the language might be superior to using an inference engine for general proof theory. However, usually no-one spends the effort, whereas the general inference engines are already highly optimized. The delayed development of even one of the basic optimization techniques known in the area of deductive databases to be usable for F-logic supports this claim [Schlepphorst, 1997].

Domain-specific languages support thinking in domain terms, instead of thinking in terms of the representation language: when modeling in an object-oriented style, an object-oriented language supports thinking in terms of classes, objects, and methods instead of relations and functions. Although the examples given are applicable in many domains, and hence not fully domain dependent, they show how to build more specific languages. A methodology for defining domain-specific declarative language has been sketched and two existing knowledge representation languages 'reengineered'. These languages, as shown, can be combined to obtain (relatively easily) a richer one. This approach is similar to approaches used in compiler design where several intermediate languages are used to bridge the conceptual gap between the start language and the target language.

7.3.2 Optimized Lloyd-Topor Transformation

The Lloyd-Topor transformation takes a general logic program (i.e., a set of clauses with First-Order bodies) and translates it into a normal logic program (logic programs with bodies consisting of a conjunction of literals). Figure 7.10 illustrates the differences in readability between a general logic

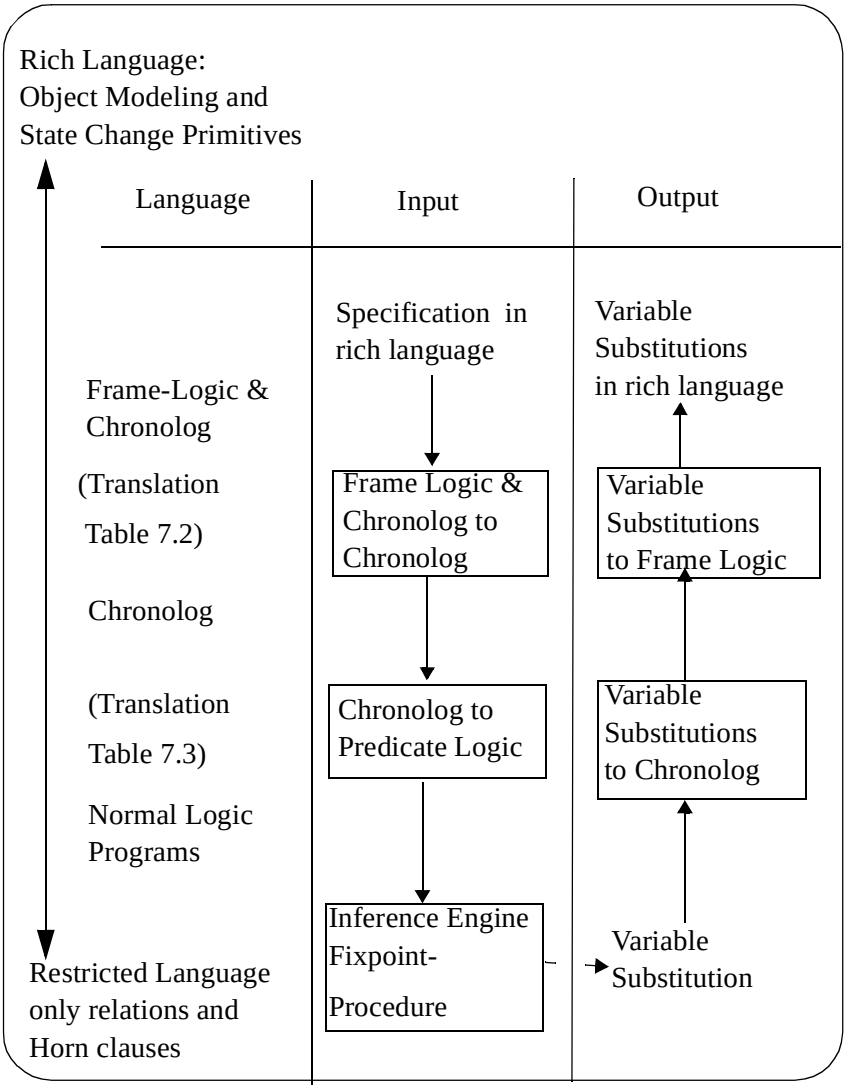


Fig. 7.9. Process and Languages

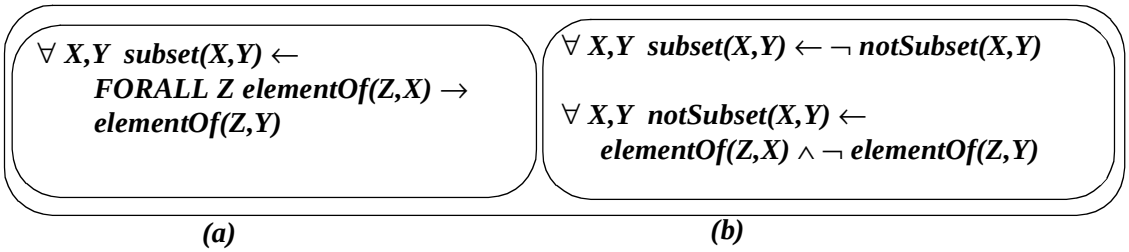


Fig. 7.10. Specification of the Subset-relation as (a) a General Logic Program and (b) as a Normal Logic Program

program and a normal logic program by specifying the subset relationship. The formula in Figure 7.10 (a) reads like the specification of the subset relationship: *X is subset of Y if every element in X is also an element of Y.* Part (b) represents the same statement as a normal logic program, using a double negation. Part (b) is much more difficult to understand, but equivalent to part (a), and fortunately can be generated automatically from the formula in part (a) by the Lloyd-Topor transformation. As noted in [Decker, 1994] the set of rules generated by the Lloyd-Topor transformation published in [Lloyd & Topor, 1984] and [Lloyd, 1987] has efficiency problems. The problems are caused by the rules responsible for translating the disjunction. The rewriting rule for the treatment of disjunction in [Lloyd & Topor, 1984] is defined as follows:

$$\frac{A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge (V \vee W) \wedge W_{i+1} \wedge \ldots \wedge W_m}{\begin{array}{l} A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge V \wedge W_{i+1} \wedge \ldots \wedge W_m \\ A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge W \wedge W_{i+1} \wedge \ldots \wedge W_m \end{array}}$$

To get all answer substitutions for the atom A, the rule engine has to evaluate the literals in $W_1 \wedge \ldots \wedge W_{i-1} \wedge W_{i+1} \wedge \ldots \wedge W_m$ twice - once for each rule. To save the extra effort, a different treatment is desirable. Instead of duplicating all literals, a new predicate symbol can be introduced to generate a new rule that contains only the elements of the disjunction. So for Ontobroker the rewrite rules are changed to:

Replace $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge (V_1 \vee \ldots \vee V_o) \wedge W_{i+1} \wedge \ldots \wedge W_m$

by $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge p(x_1, \ldots, x_n) \wedge W_{i+1} \wedge \ldots \wedge W_m$

$$p(x_1, \ldots, x_n) \leftarrow V_1$$

$$\ldots$$

$$p(x_1, \ldots, x_n) \leftarrow V_o$$

where **p** is a new, previously unused predicate symbol and the $x_1, \ldots, x_n$ are the free variables that occur in $(V_1 \vee \ldots \vee V_m)$.

In Appendix B an optimized Lloyd-Topor transformation based on rewrite rules is defined. A procedural algorithm is also given, which implements the rewrite rules and is more suitable for applications. The worst-case complexity of the transformation algorithm is exponential with respect to the length of the formula. The treatment of logical equivalence is responsible for the exponential complexity, which results in a duplication of the input formula. However, input formulae are usually small, and logical equivalence is not very common in logical formula. Not taking logical

equivalence into account, the complexity of the algorithm is linear to the size of the input formula, thus also the average case complexity is very close to linear. The average case complexity of the algorithm induced by the rewriting rules defined in [Lloyd & Topor, 1984] is much worse, since duplication occurs also in the case of disjunction, a common occurrence in predicate logic formulae.

7.4 Selecting an Evaluation Procedure and Semantics

As sketched in Section 7.3, a translation approach to F-Logic is being followed which results in a normal logic program. Normal logic programs are executable with standard techniques from logic programming. To be able to evaluate normal logic programs we need to select an appropriate semantics. Several possibilities exist. The most important types of semantics and evaluation procedures are sketched, and the requirements that finally led to the selection of the evaluation procedure of SiLRI are clarified.

The generally accepted semantics for horn logic programs is based on least Herbrand models, defined by a fixpoint construction. The usual definition of least fixpoints works only for definite logic programs, i.e., Horn clauses without negation. However, for some applications (e.g., expressing universal quantifiers in rule bodies) non-monotonic negation is needed (cf. [Brewka & Dix, 1999]). Several possible semantics for Horn clauses with negation exist:

- *Perfect Model semantics* [Przymusinski, 1988a]. The perfect model semantics require a partition of logic programs in strata such that there is no recursion over negation. F-logic specifications almost always result in a low number of predicates. As a consequence, specifications which involve negation are almost always non-stratifiable.
- *Stable model semantics* [Geldfond & Lifschitz, 1988]. Computing the stable model semantics has an exponential complexity and is not goal-directed. Even if facts are deleted or added to the database, which are not related to a query by means of used predicate symbols, the results of the query may change [Brewka & Dix, 1999]. In a dynamic, changing environment this is not a desirable property, since query results may be changed by events unrelated to the query.
- *Well-founded semantics* [van Gelder et al., 1991]. The well-founded semantics has an intuitive behavior based on three value models, providing a semantics for every logic program. The worst-case complexity is polynomial for function symbol-free languages.

The well-founded semantics is the most appealing semantics for our purposes. Therefore, this option has been selected for SiLRI.

7.5 Implementation of SiLRI

The implementation of SiLRI is based on JAVA and consists of two core parts: (1) the F-logic preprocessor, which translates a given ontology or set of instance definitions into normal logic programs, and (2) the inference core, which performs a bottom-up evaluation of the resulting normal logic program by a variant of dynamic filtering adapted to the well-founded semantics.. The F-logic parser was developed using the Java Compiler Compiler (JavaCC[1]), which accepts LL(k) grammars. Starting from a set of F-logic rules, based on the abstract Syntax Tree generated by the JavaCC F-logic parser the F-logic transformation rules are applied, resulting in a general logic program. In a second step the Lloyd-Topor transformation is applied - the final output is a normal

1. See http://www.webgain.com/products/java_cc/

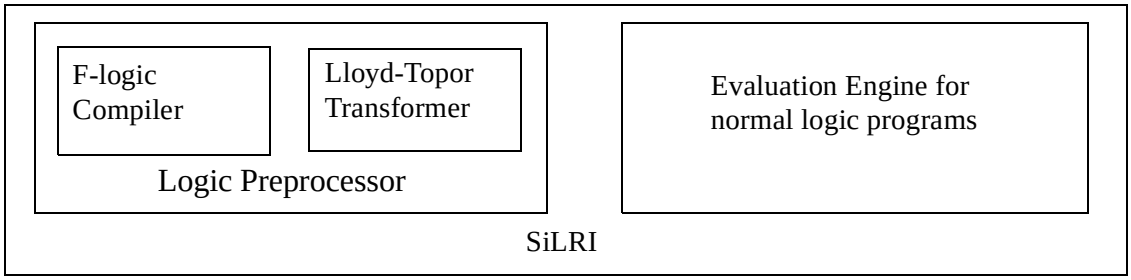


Fig. 7.11. SiLRI's Architecture

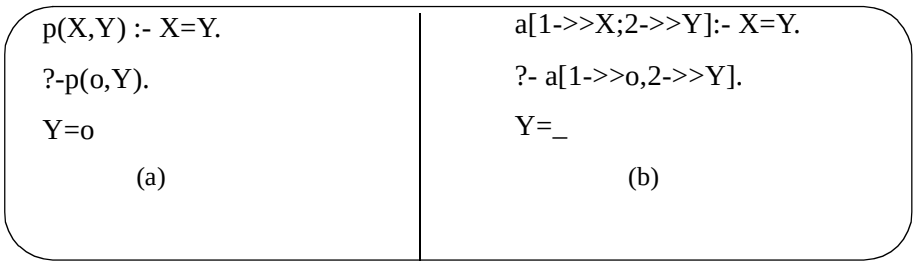


Fig. 7.12. Counter-Intuitive Results in F-logic

logic program. The current implementation uses an inference core based on the KARL specification language (see [Angele, 1993]). It is also possible to use other inference engines. XSB [Freire et al., 1997] has been successfully used as an alternative implementation after performance improvements had been made, induced by the author's work [Johnson, 1999].

7.6 Critical Evaluation of F-Logic as a Representation and Query Language for the Web

Frame-Logic, as defined in [Kifer et al., 1995] has some obstacles when considered as an ontology language for the Web:

- Frame-Logic sometimes has a counter-intuitive behavior: Figure 7.12 illustrates an example. Case (a), of Figure 7.12 formalized in Horn logic, returns a variable binding ***Y=o*** for the query, since in the process of query evaluation the variable *Y* gets bound to the value of variable *X*, which was bound to **o** by the query. In Figure 7.12 (b) the query answer is just an uninstantiated variable, although the query and rule are similar to the query and rule in case (a), and the same intuitions apply. The reason for this behavior is that the rule a[1->>X;2->>Y] :- X=Y. :of the rule (b) in Figure 7.12 is semantically equivalent to two independent rules: a[1->>X] :- X=Y $\wedge$ a[2->>Y] :- X=Y. Thus variable '*X*' is not bound to ***o***.
- F-logic has a fixed conceptual model. In science, engineering and modeling a plethora of modeling languages and styles exists, and the Web is the place where all these different languages come together. A rule language aiming to integrate data from different sources needs mechanisms that allow for representation and reason with multiple conceptual models and semantics. F-logic does not have this flexibility.
- F-logic makes a distinction between set-valued and scalar-valued methods. However, often a finer-grained distinction is necessary - e.g., UML allows for definition of numeric cardinality constraints to be associated with relations. There is no defined way to specify cardinality constraints within F-logic.

- Not all information sources on the Web are reliable and trustworthy. Thus, any system aiming to reason with data from the Web needs to distinguish between different data sources. F-logic does not have a built-in mechanism to represent the source of data.(1)
- There is no defined way to unambiguously construct IDs for objects. Without such a mechanism it is likely that the same ID gets used for different objects. This is not a problem as long as the different information sources are separated. But when these information sources are published on a Web, there is a possibility that some applications will be required to integrate both sources. When combining information sources it becomes very difficult to figure out when two object IDs denote the same object or not.

1. However, it is possible to encode the source of data with a parameter of a method, e.g., *stefan[name@(“http://www-db.stanford.edu/~stefan”) -> “Stefan Decker”]*.

Chapter 8 Document Annotation

This chapter presents the *information provider* aspects of Ontobroker. Itr introduces the concept of *metadata* and identifies the class of metadata that Ontobroker deploys. *HTML-A* is the annotation approach that was developed within Ontobroker for the ontology-based annotation of HTML documents. Then, this chapter presents an annotation tool for HTML-A: *OntoPad* is an WYSIWYG HTML-Editor, which supports the creation of ontology-based metadata. Finally, HTML-A is extended and generalized to *XAL* - the XML Annotation Language, and an implementation of XAL is presented.

8.1 Metadata

Metadata - *Data about Data* - is nothing new. Llibrarians have generated metadata for centuries (e.g., the Dewey Decimal Classification[1] or the machine readable MARC standard[2]) to enable users of libraries to easily find relevant literature. Metadata was also used for quite a while in software systems, e.g., the table definitions inside a relational database and file information stored inside a file system is metadata. Table 8.1 (taken with modifications from [Kashyap & Shet, 1996]) provides a classification of different types of metadata. The metadata currently interested in for the Ontobroker system is *domain specific metadata*, based on ontologies for documents available on the Web, especially for XML and HTML documents. The annotation of documents with semantical information (metadata) can be done in two ways:

1. The metadata can be *separated* from the content, e.g., located at the beginning or at the end of a document, or in a separate file.
2. The metadata can the *integrated* into the content of the document. The benefit of this method is that metadata information is not repeated and redundancy is avoided. If a separate representation is required (e.g., when storing or querying the data is necessary) it is feasible to extract such metadata from the documents.

The most common way to annotate HTML documents according to the first method is with HTML-`<META>`-Tags. `<META>`-Tags enable the attachment of arbitary attribute-value pairs. One common use is the representation of DublinCore[3]-based metadata in HTML. Figure 8.1[4] depicts an example of how Dublin Core based metadata is representable using the `<META>`-Tags. However, the `<META>`-Tags do not go beyond simple attribute-value pairs, associated with complete HTML documents. No extension which would enable the representation of ontology-based, domain specific metadata is proposed.

Ontobroker follows the second approach: the integration of metadata into the content of the document. The following section presents the annotation language *HTML-A* and supporting Ontology-based annotation tools, and later extends the approach to XML by presenting *XAL*, the XML Annotation Language.

1. cf. http://www.oclc.org/dewey/
2. see http://www.loc.gov/marc/
3. cf. http://www.dublincore.org
4. taken from http://www.ou.edu/cas/slis/courses/LIS5990A/slis5990/authortools/dublincore.htm

<table>
<tr><td>Content-independent metadata</td><td colspan="3">Content Dependent metadata</td></tr>
<tr><td></td><td colspan="3">This type of metadata depends on the content of the document it is associated with. Examples are the size of a document, max-colors, number-of-rows, number-of-columns of an image. There is a further categorization of this kind of metadata:</td></tr>
<tr><td></td><td>Direct Content-based metadata</td><td colspan="2">Content-descriptive metadata</td></tr>
<tr><td></td><td></td><td colspan="2">This metadata describes the contents of a document without direct utilization of the contents of the document. An example is textual annotations describing the contents of an image. This type comes in two flavors:</td></tr>
<tr><td></td><td></td><td>Domain-independent metadata</td><td>Domain Specific metadata</td></tr>
<tr><td>This type of information does not depend on the content of the document with which it is associated. Examples are location, modification-date of a document and type-of-sensor used to record a photographic image. There is no information content captured by such metadata but these might still be useful for retrieval of documents from their actual physical locations and for checking whether the information is up-to-date or not.</td><td>This type of metadata is based directly on the contents of a document. A popular example of this is full-text indices based on the text of the documents. Inverted tree and document vectors are further examples.</td><td>These metadata capture information present in the document independent of the application or subject domain of the information. Examples of these are the C/C++ parse trees and HTML/SGML document type definitions.</td><td>Metadata of this type is described in a manner specific to the application or subject domain of information. Issues of vocabulary become very important in this case as the terms have to be chosen in a domain specific manner. Examples of such metadata are relief landcover from the GIS domain. In the case of structured data, the database schema is also an example. Another interesting example is domain specific ontologies, terms that may be used as vocabulary to construct metadata specific to that domain.</td></tr>
</table>

Tab. 8.1. Metadata Classification (from [Kashyap & Shet, 1996] with Modifications)

8.2 HTML-A

HTML-A (*HTML plus semantic Annotations*) is an extension of HTML [Ragett et al., 1998] with the goal of providing the ability to integrate ontology-based metadata within the visible part of HTML. HTML-A is already presented in detail in [Erdmann, 2001], thus only the language will be presented here to the extent necessary for presentation of the enhanced HTML editor and for the successor of HTML-A, the XML Annotation Language (XAL).

8.2.1 Design of HTML-A

HTML [Ragett et al., 1998] is the most widely used document description language on the Web. HTML is standardized and is one of the pillars (among HTTP [Fielding et al., 1999] and URL/URIs [Berners-Lee et al., 1998]) of the World Wide Web. HTML aims to provide the primitives that

enable users to describe the structure and in part also the layout of hypertext documents. Besides the already mentioned META-Tag, HTML does not provide the means to describe the semantics of documents. HTML-A enables users to include ontology based metadata without redundancy in documents.

HTML-A is related to Don E. Knuth's idea of *literate programming* [Knuth, 1984]. However, usually *literate programming* is used to extract human-readable comments out of computer-understandable program code. HTML-A works the other way round: the goal is to extract formal, computer-understandable data out of human-readable text documents. This approach enables the incremental formalization of documents and does not interfere with Web browsers.

8.2.2 Syntax of HTML-A

HTML-A provides three different epistemological primitives to annotate ontological information in Web documents:

- Objects may be defined as *instances* of classes. Objects are identified by URIs.
- Value of an object's *attributes* may be set.
- *Relationships* between two or more objects may be established.

```
<html>
<head>
<title> Web Management: Dublin Core </title>
<META NAME="DC.Title" CONTENT="Web Document Management">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#title">
<META NAME="DC.Title.Alternative" CONTENT="LIS 5990 Summer 2000">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#title">
<META NAME="DC.Creator.PersonalName" CONTENT="Koehler, Wallace">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#creator">
<META NAME="DC.Creator.PersonalName.Address" CONTENT="wkoehler@ou.edu">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#creator">
<META NAME="DC.Subject" CONTENT="(SCHEME=LCSH) Dublin Core, metadata, resource discovery">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#subject">
<META NAME="DC.Description" CONTENT="A web-based graduate course on bibliographic management of the WWW">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#description">
<META NAME="DC.Publisher" CONTENT="School of Library and Information Studies, University of Oklahoma">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#publisher">
<META NAME="DC.Date" CONTENT="(SCHEME=ISO8601) 2000-05-01">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#date">
<META NAME="DC.Type" CONTENT="Text.Index">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#type">
<META NAME="DC.Format" CONTENT="(SCHEME=IMT) text/html">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#format">
<LINK REL=SCHEMA.imt HREF="http://sunsite.auc.dk/RFC/rfc/rfc2046.html">
<META NAME="DC.Identifier" CONTENT="http://www.ou.edu/">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#identifier">
<META NAME="DC.Language" CONTENT="(SCHEME=ISO639-1) en">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#language">
<META NAME="DC.Coverage" CONTENT="metadata">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#coverage">
<META NAME="DC.Rights" CONTENT="Copyright Wallace Koehler 2000 All Rights Reserved">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#rights">
<META NAME="DC.Date.X-MetadataLastModified" CONTENT="(SCHEME=ISO8601) 2000-02-05">
<LINK REL=SCHEMA.dc HREF="http://purl.org/metadata/dublin_core_elements#date">
</head>
```

Fig. 8.1. META Tags representing Dublin Core Metadata

All three primitives are expressed by using an extended version of a frequent HTML tag, the anchor tag:

*<a ...> ... *

The anchor tag is usually used to define named locations within a Web page and hypertext links to other locations in the WWW. Thus, it contains the *name* and *href* attributes to fulfill these purposes, respectively. For annotating a Web page another attribute was added to the syntax of the anchor tag, namely the *onto* attribute. All three attributes may contain information which describes objects relevant to the Ontobroker.

Typically a provider of information first defines an object by stating which class of the ontology the object is an instance of. For example, if the researcher Richard Benjamins (see Figure 8.2 for his homepage and the corresponding HTML source) defines an object representing himself in HTML-A as an instance-of the class *Researcher*, he needs to express this in HTML-A as follows:

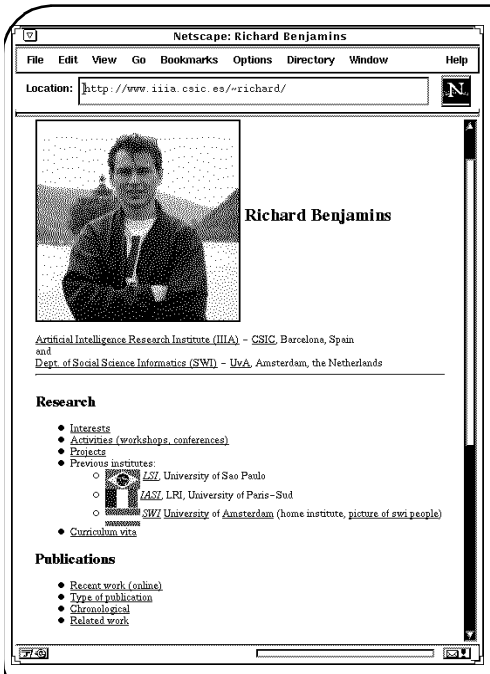
* *

This line states that the object denoted by the handle '*http://www.iiia.csic.es/~richard*' is an instance of the class *Researcher*. Actually, the handle given above is the URL of Richard Benjamin's homepage; thus, from now on he is a researcher denoted by the URL.

Each class is associated with a set of attributes. Each instance of a class can define values for these attributes. To define an attribute value on a Web page the knowledge provider has to list the object, the attribute, and the value. For example, the ontology contains an attribute email for each object of class *Researcher*. If Richard Benjamins wants to provide his email address, he puts the following annotation on his homepage.

* *

This line states that the object denoted by the handle 'http://www.iiia.csic.es/~richard' has the value '*mailto:richard@iiia.csic.es*' for the attribute email.




```
<html>
<head><TITLE> Richard Benjamins </TITLE>
  <a onto="page:Researcher"> </a>
</head>

<H1> <A HREF="pictures/id-rich.gif">
<IMG SRC="pictures/richard.gif"></A>
<a onto="page[photo=href]"
  HREF="http://www.iiia.csic.es/~richard/pictures/richard.gif" >
</a>

<a onto="page[firstName=body]">Richard</a>
<a onto="page[lastName=body]">Benjamins </a>
</h1> <p>

<A HREF="#card">
  Artificial Intelligence Research Institute (IIIA)</A> -
<a onto="page[affiliation=href]"
  href="http://www.csic.es/">CSIC</a>, Barcelona, Spain
<br> and <br>
<A onto="page[affiliation=href]"
  HREF="http://www.swi.psy.uva.nl/">
  Dept. of Social Science Informatics (SWI)</A>
-
<A HREF="http://www.uva.nl/uva/english/">UvA</A>
```

Fig. 8.2. An example of an annotated HTML page.

Several objects and attributes can be defined on a single Web page, and several objects can be related to each other explicitly. Given the name of a relation REL and the object handles Obj1 to Objn this definition looks like:

* ... *

A set of keywords with special meanings is allowed as part of the annotation syntax. The keyword *page* represents the whole Web page where the ontological mark-up is contained. This is useful when looking at the page as representative of an object. For example, a homepage of a researcher might represent that person in the knowledge base, defined by the following kind of annotation:

* *

The following annotation defines the affiliation attribute of the object denoted by the URL of the current page and takes the value from the anchor-tag's href-attribute:

IIIA - Artificial Intelligence Institute.

The href-keyword allows the establishment of relations between objects without a lot of typing, because the hyperlinks can be 'reused' within the ontological markup.

Hyperlinks are not the only element that can be directly integrated as semantic information, the text that is rendered by a browser can also become a part of the formal knowledge, e.g.

*Richard" *

defines the text *Richard* (between *<a ...>* and **) as a value of the attribute *firstName*. The keyword *body* enables the reuse. By these conventions the annotation of Web pages becomes more concise and redundancy can almost be avoided. The tight coupling between metadata and content eases maintenance when resources are frequently changing, since changes to the rendered data are automatically reflected in the semantic markup.

8.3 OntoPad - an Annotation Tool for HTML-A[(1)]

Annotating HTML documents manually with HTML-A is possible, but is very cumbersome and error-prone: a knowledge annotator has to examine the HTML-code of the page to annotate and a representation of the ontology in parallel to identify possibilities for annotation. Then the annotator has to write the annotation directly in the HTML source, using terms from the ontology, which often results in typing errors. Furthermore, browsing a large ontology without tool support is very uncomfortable and time consuming, since this means scrolling in a large text file.

Clearly, tool support for the annotation process simplifies and speeds up the creation of semantic annotations by providing convenient access for browsing the ontology and inserting annotation into the HTML source. A tool providing annotation support for HTML-A needs to fulfill the following requirements:

- It needs to provide *facilities for browsing* of an arbitrary domain-ontology. Ontologies may have several hundred or even thousands of concepts and attributes. An interactive browsing facility enables annotators to deal with large amounts of information.

1. *OntoPad*, an HTML editor providing support for HTML-A, was initially presented in [Benjamins & Fensel & Decker & Gomez-Perez, 1999] as part of the $(KA)^2$-Initiative (see Chapter 10) and was to our knowledge the first HTML-editor supporting expressive metadata.

- To *automate the integration* of the metadata-annotation in the HTML as much as possible. No typing of already electronic available data (names of concepts and attributes) should be necessary.

In Ontobroker the following editing process was chosen: the annotator sees the HTML document as it appears in a Web browser, and selects the ontology they want to use as a basis for the annotation. Then the annotator selects the part of the HTML page to be annotated (usually text or pictures) and opens the annotation dialog, which offers a facility to browse the ontology, select classes of the ontology, and specify the information that should be inserted into the page. In the example page shown in Figure 8.3 the knowledge provider selects the text *Benjamins* and chooses to add an annotation. Then the ontotag-dialog window pops up (see Figure 8.4) and gives the possibility to open the hyperbolic ontology view(1) (see Figure 8.4), which is used to browse the ontology and to select classes. In the example the annotator selects the class *Researcher*. After this selection all attributes of *Researcher* defined in the ontology are available in the annotation dialog window. The knowledge provider chooses the attribute *lastname* and the HTML-A keyword *body*, saying that the selected text *Benjamins* is the lastname of the page owner. After the annotator selects *OK*, the annotation is inserted into the HTML text and the HTML source can be saved to disk..

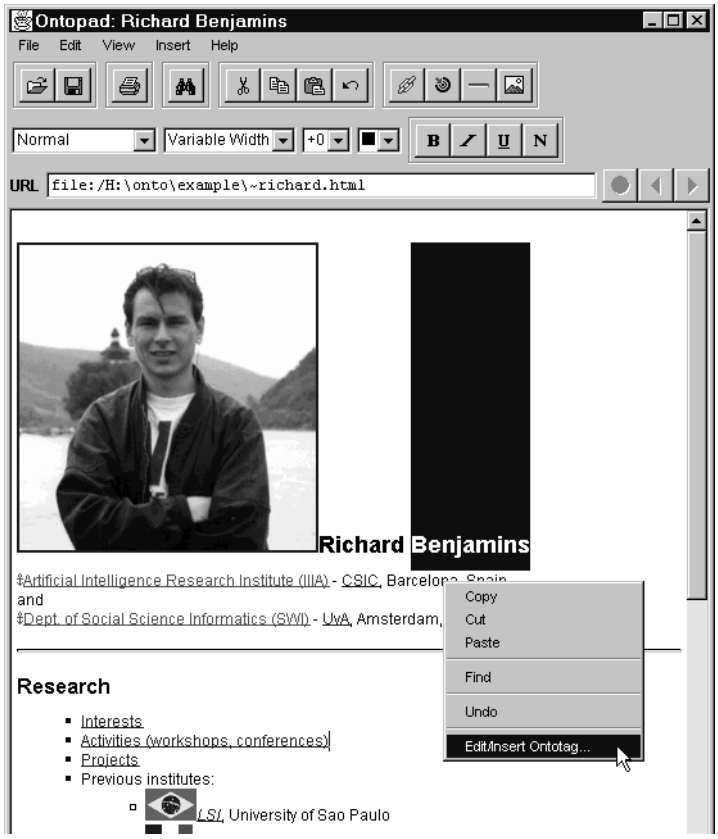


Fig. 8.3. Screenshot of OntoPad

1. The hyperbolic view is discussed in more detail in Chapter 9.

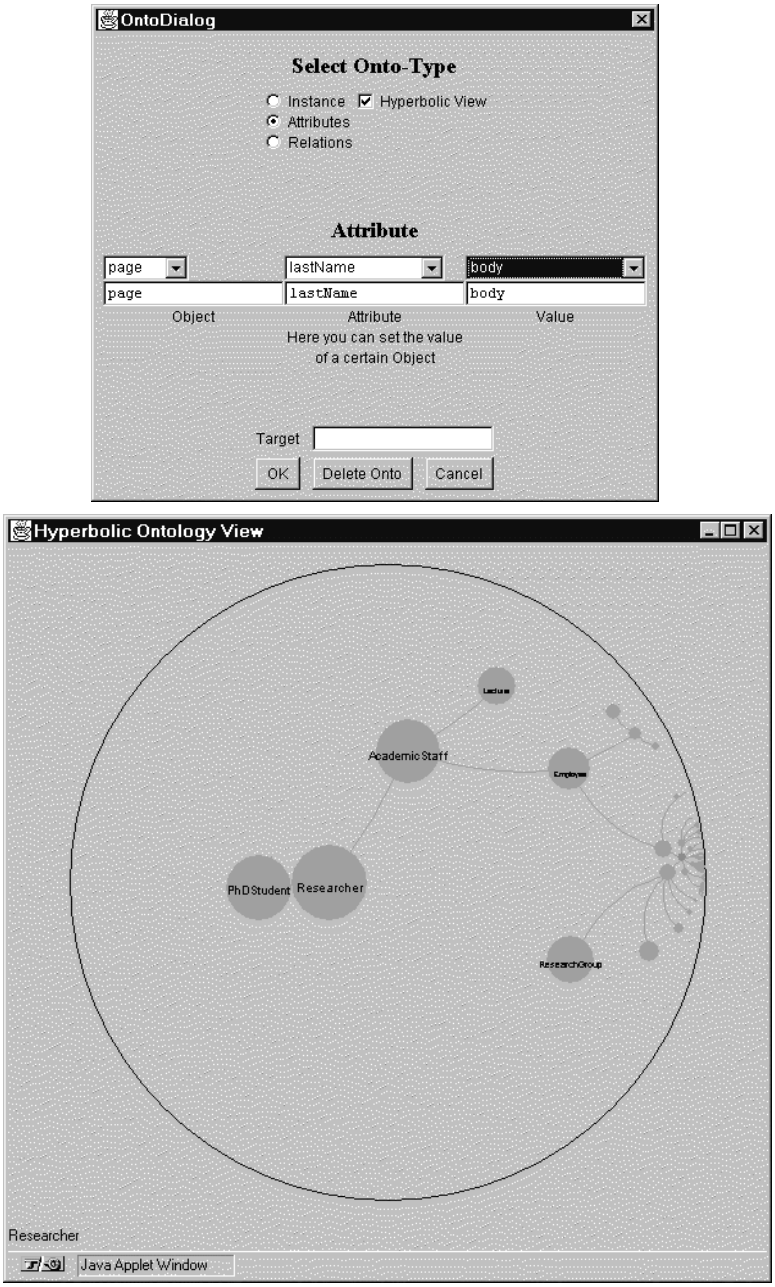


Fig. 8.4. Ontotag Annotation Dialog and Hyperbolic View with Class Researcher in Focus

8.3.1 Related Work on Annotation Editors

The annotation process is similar to the model construction in Knowledge Acquisition methodologies like MIKE (cf. [Angele et al., 1998], [Angele et al., 1996], [Neubert, 1993]) or VITAL (cf. [Motta et al., 1996]): the methodologies start with a set of text documents, which are examined and entities like activities, concepts, and attributes are identified and marked in the text. These methodologies provide tools similar to the annotation editor - especially the Acquisition Tool in MIKE (cf. [Neubert, 1993]) and the Protocol editor of PC-Pack. (cf. [Milton et al., 1999]) are similar. Text sources are marked with some predefined terms, e.g., *activity*, *concept*, *attribute* etc.

These predefined terms act as a limited ‘hardwired’ representation ontology. With this analogy in mind, OntoPad can be regarded as a generalization of these specialized tools in the knowledge acquisition area, since OntoPad allows arbitary ontologies. Other annotation tools have been developed, some in a commercial setting - e.g., [Hori, 2000] presents an editor enhanced with annotation capabilities. Hori supports the annotation with RDF[1] annotations, but also possesses only a fixed ontology. Also related are annotation tools like Annotea [Kahan et al., 2001]. Annotea is a Web-based shared annotation system based on an RDF infrastructure. However, Annoteas annotations also follow a fixed ontology, and thus Annotea is not able to provide domain-specific, general machine processable data. It would be promising to look at combining both approaches: Ontology based metadata, and Web-based shared annotation systems, since shared annotations allow for the definition of formal annotations about information without necessarily owning the annotated site.

A weakness of OntoPad is discussed in [Erdmann et al., 2000]: OntoPad does not remember annotations, such that already defined objects can be reused in the annotation process. [Erdmann et al., 2000] and [Handschuh et al 2001] present a successor to OntoPad which possesses a memory component.

8.4 XAL - The XML Annotation Language

HTML-A has weaknesses: it is not suitable for the annotation of XML documents, it allows only one ontology in an HTML document, and does not support annotation of regular structures like tables and lists. This section presents the XML Annotation Language, which is remedying these issues.

8.4.1 HTML and XML

> *“Although HTML is the most successful electronic-publishing language ever invented, it is superficial: in essence, it describes how a Web browser should arrange text, images and push-buttons on a page. HTML’s concern with appearances makes it relatively easy to learn, but it also has its costs.”* [Bosak & Bray, 1999]

HTML was designed for the purpose of platform independent Web site presentations. It has been extended to handle dynamic content and to support different applications like video and audio clips or Java applets. But HTML is considered to lack the flexibility for solving future requirements of the Internet. XML (Extensible Markup Language) [Bray et al., 1998] is widely regarded as the more flexible successor of HTML.

XML enables one to specify a tag set and to represent the structural relationships of those tags. The HTML set of tags is fixed, which allows users to leave the language specification out of the document and makes it much easier to build applications, but this ease comes at the cost of limiting HTML in several important respects. Compared to HTML, XML enables separation of the structure and the layout of a document.

1. see Chapter 13 for more information about RDF

8.4.2 XML and Semantics

Jim Gray writes in the preface of *Data on the Web* [Abiteboul et al., 2000] about XML:

> *"This is an area and era of great intellectual ferment. It is the collision between three cultures: the everything-is-a-document culture, the everything-is-an-object culture, and the everything-is-a-relation culture. Each group sees different aspects of the elephant.[...]XML is the one thing all three groups agree on (minus a few details). It, or one of its children, will become the intergalactic dataspeak: the standard way to interchange semistructured and structured data among computer systems."*

Starting from Jim Gray's view, two different perspectives can be distinguished:

- *XML is a language to describe data*. Adopting this perspective it is important to investigate how conceptual models like ontologies relate to XML and DTDs. [Erdmann & Studer, 2001] and [Erdmann, 2001] have, among others, investigated this viewpoint and describe how ontologies can be used as conceptual models for XML-based data.
- *XML is a language to structure documents*. XML is a simplified version of SGML [ISO 8879, 1986]. SGML's primary use is to structure text documents, e.g., manuals and other text documents (e.g., for Boeing aircraft). Here the tags don't describe the *content* of the text, but rather the *structure* (e.g., chapters and sections in manuals).

Starting from the latter viewpoint, an annotation language enabling the annotation of XML documents with ontology based metadata is required. HTML-A has some limitations, which make a direct transfer of HTML-A unsuitable. As a consequence, XAL - the XML Annotation Language - has been developed.

8.4.3 Limitations of HTML-A and Requirements for XAL

The following requirements for an XML annotation language were identified:

- *Usage of multiple ontologies*. In HTML-A it is not possible to use more than one ontology for annotations. This option needs to be included in XAL, since a document may be annotated according to more than one domain ontology.
- *Annotation of single tags including access to other tags*. HTML-A enabled the annotation of single specific tags (anchor tags) with meta information. But HTML-A provides no capabilities to access the content of the previous or the next tag in the document or even a tag anywhere else on the Web. This characteristic should be included in an XML annotation language, since the information may be distributed over the document or even over the Web.
- *Exploiting regular document structures*. HTML-A provides no support for the annotation of regular structures, like tables or lists. Every single element needs to be annotated. Regular structures occur frequently in XML documents and their annotation should be supported.
- *Annotations of HTML-code*. The new annotation language for XML should be backward compatible.
- *Variables*. In order to make the annotation process easier and to preserve the readability of the document code it should be possible to declare variables and give them a value. Inside the document the value of the variable is used instead of the variable name itself.

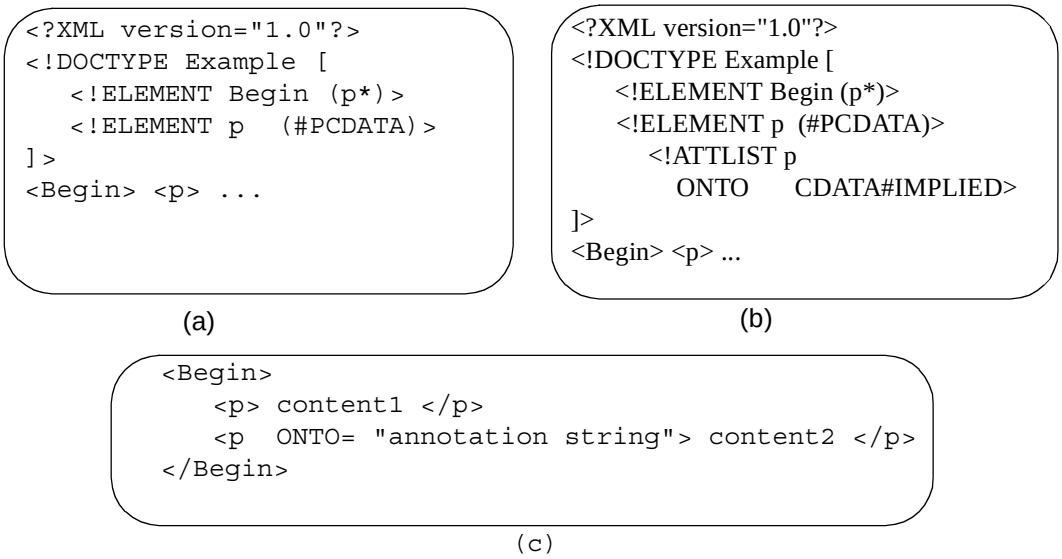


Fig. 8.5. : Example for an internal DTD without (a) and with (b) the Declaration of the Onto-attribute, and an XML document (c) following the DTD in (b)

- *Generalized short-cuts*. The short-cuts defined in HTML-A (*BODY, PAGE* and *TAG*) should be extensible such that other tags are easily accessible.
- *Text-filtering*. HTML-A only allows use of complete text that is the content of a tag or attribute. It should be possible to filter the text content of a tag.
- *More expressive data-types*. HTML-A only provides a generic 'number' datatype. More datatypes should be possible to enable real world applications.
- '*As simple as possible, but not simpler*'. The new language as a whole should fit to the principle that a publisher has to be able to use the language in an efficient way.The complexity should be as low as possible. On the other hand, the language should offer enough functionality to realize all the mentioned requirements. Variables may be declared and used throughout the document.

8.4.4 Representing Ontological Information in XML Documents

XAL follows the HTML-A approach of document annotation - using a specific XML attribute named *Onto* - the content of the Web documents is annotated inside the document tags. Thus, information is not repeated and redundancy is avoided. As mentioned above, the ontological annotation - the value of the *Onto* attribute - has the form of a string. Appendix C provides the complete EBNF syntax diagrams for XAL.

8.4.4.1 Transformation of DTDs

An XML document is considered to be *valid* if it follows the grammar rules specified by its *Document Type Definition* (DTD). To preserve validity, the mentioned ONTO-attribute has to be declared for each anotated tag within the DTD. Otherwise, after the annotation process, the XML document would be not valid anymore (but the document would still be *well-formed*, which means the document is still a syntactically correct XML document). Figure 8.5 depicts a DTD before (a) and after (b) the declaration of the ONTO-attribute is inserted. The same transformation (as shown

```
<BODY
     xmlns:ontology1  = "http://www.semanticweb.org/swrc#"
     xmlns:ontology2 = "http://www.semanticweb.org/wordnet#">
       <A  ontology1:ONTO = "id1234 : Person">  ... </A>
       <B  ontology2:ONTO = "word1235 : Synset">  ... </B>
...
</BODY>
```

Fig. 8.6. : Example for namespace declarations and usage

in Figure 8.5) is universally applicable for all DTDs and all elements, and easily automatable. The elements of the XML document can then be annotated with an ontological string as attribute value of the ONTO-attribute-defined in the DTD (see Figure 8.5(c) for an example).

8.4.4.2 Multiple Ontologies

To enable the usage of multiple ontologies for annotations (as mentioned in Section 8.4.3) a mechanism is needed to distinguish between ONTO-annotations adhering to different ontologies. The approach we propose is to add a prefix representing the ontology in front of the ONTO-attribute-name. With the namespace extension the XAL-parser can assign the ONTO-annotation to the appropriate ontology.

As ontology prefixes XML namespace prefixes as described in [Bray et al., 1999] are used. XML namespaces provide a simple method for qualifying names used in Extensible Markup Language documents by associating them with namespaces identified by URIs.

From an XML point of view, the ONTO-attribute is just a regular XML-attribute. The difference is that the attribute name consists of an XML namespace prefix and the string ONTO, according to the XML namespace syntax. The namespace prefix acts here as the name of a declared ontology. We assume that applications, able to process the annotations, know this ontology and its namespace, and are thus are able to interpret the information.

The ontology name is structured as an *NCName* (as specified in [Bray et al., 1999]): an XML name without a colon (':').

Additionally, an XML namespace declaration specifying the used ontologies as namespace prefixes is required. There are several possibilities for where to place the namespace declaration. Placing the namespace declaration either in the head of the XML document or in the start tag of an element which is located at a high level of the document hierarchy is suggested, since XML namespace declarations are valid for the element where the declaration is located and for all offspring elements of this element. If the namespace declaration is located at the root element of the document, the namespace is defined for the whole document. See Figure 8.6 for an example, depicting namespace declarations and ONTO attribute definitions.

8.4.4.3 Declaration Types

The value of an Onto-attribute (the right side of the equation) consists of one or more declarations using the ontologies, possibly separated by semicolons. XAL provides four different types of annotation declarations:

1. *Tag Annotation*. A tag annotation of a single element content or attribute value is only valid for this special data record.

2. *Structure Annotation.* XAL provides additional possibilities for annotating data records, i.e., tables and lists, in an efficient manner. The annotation type annotates not only a single element, but also several elements which are either nested or offspring elements. Ideally, only one annotation in the root element describes the whole document semantically, e.g., for annotating every element of a table it is only necessary to annotate the root element of the table.
3. *Relation Annotation.* Relation annotations correspond to the relations of HTML-A.
4. *Declaration Annotation*. A declaration annotation provides the possibility to declare special variables and assign values to them.

Tag and Structure Annotations are inspired by the F-logic syntax. The general template is object*:class[attribute$_1$=value$_1$;...;attribute$_1$=value$_n$]* with several possibilities to construct values and object IDs. The following section explains the annotation types in more detail.

8.4.4.4 Tag Annotations

A Tag Annotation is either an:

1. *Instance definition.* An object ID can be defined as an instance of a certain ontology class. The object identifier is required to be globally unique (e.g., be a URI). The object ID might also be a predefined keyword.
2. *Attribute-value definition*. A value of an object's attribute can be defined. The object ID is followed by one or more attribute-value-pairs in brackets, separated by a semicolon. Instance definitions and attribute value definitions can also be combined. Simple attribute values are either a URI (an object ID), a string, or a number.

An object ID is defined by one of the following options, which can also serve as attribute values.

1. *The PAGE*-Keyword. The PAGE keyword is replaced by the URI of the currently processed XML document. The meaning of PAGE is identical to its meaning in HTML-A.
2. *The TAG-Keyword*. The named fragment of the document defined by the current tag becomes an instance of the specified class.
3. *An ID-Name or a Path*. The ID is given by either the ID of the referenced tag or - if this element doesn't have an ID - the path of this tag in the document tree. The difference with the TAG-keyword is that not only the current tag but any tag of the document can be referred to. If elements without an ID-attribute are referred to, a path is generated. A path (using the XPOINTER recommendation(1)) can be explicitly provided by the author or it is generated automatically if an ID is omitted. The path or ID-name may be terminated with an attribute

1. see http://www.w3.org/TR/xptr/

name of the selected tag or the keyword *Body*. The following example illustrates the usage of path expressions. In this example the BODY of the second <A>-element within the ROOT-element is defined to be the value of the attribute *firstname*.

```
<ROOT>
  <A> ... </A>
  <A> Stefan </A>
  <A ont1:ONTO="PAGE:Person [firstname =
           PATH(ROOT[0].A[1]).BODY ]">  ... </A>
</ROOT>
```

Fig. 8.7. Example of Path Expressions inside a Tag-Annotation

4. It is possible to filter values with regular expressions, as in the following example. The regular

```
<A  ont1:ONTO="PAGE : class1 [ age = BODY((\d+)) ] ">
          Name: Thomas Smith;  Age: 34 years
</A>
```

Fig. 8.8. Example of a Regular Expression inside a Tag-Annotation

expression just filters the digits out of the body of the tag.

5. *An XML-attribute-name*. This name points to an attribute of the current tag. Then the object ID or attribute value is the value of the named attribute. This is a generalization of the HREF keyword in HTML-A.
6. *A skolem function.* A skolem function (cf. [Hull & Yoshikawa, 1990]) is a function which produces as a result a unique value for given parameters. The following example uses skolem

```
<company   ID="c76"  ont2:ONTO = "TAG:organization [
          boss = S(TAG, BODY)[ salary = BODY ] ] ]" >
1.000.000</company>
```

Fig. 8.9. Example of a Skolem Function inside a Tag-Annotation

functions and nested attribute-value structures. In this example the value of the *boss-attribute* is itself an object (generated by the skolem function), which has an attribute *salary* which gets the value '`1.000.000`'.

Additionally, the following attribute values are possible:

1. *Strings*. The simplest way to define an attribute value is to use a string.
2. *Data types*. Strings or string-producing expressions (like BODY expressions) can be converted into other data types, like FLOAT, DOUBLE, LONG, SHORT, BYTE, INTEGER and BOOLEAN.

The example depicted in Figure 8.10 illustrates the use of the so-far-introduced primitives:

```
<song   ID = "47"   lang = "en"> The Boxer </song>
...
<song   ID = "48"   lang = "de"> Männer </song>
...
<band ID = "53"   kind = "pop"   no_of_songs = "432"
      ont1:ONTO = "TAG : Interpret [
language = ID(47).lang ;
              name = BODY;
              nr = SHORT(no_of_songs) ;
              sort_of_music = kind ;
              example = ID(47).BODY ] " >
Simon & Garfunkel </band>
...
<band ID = "54"   kind = "dt_rock"  href = "www.groenemeyer.de"
      ont1:ONTO = "href : Interpret [
              sort_of_music = kind ;
              example = ID(48).BODY ;
              persons = SHORT("5") ;
              name = 'Herbert Grönemeyer' ] " >
```

Resulting F-Logic facts:

```
http://www.semanticweb.org/music|53 : Interpret[
      language->"en";
      name->"Simon & Garfunkel";
      nr -> 432;
      sort_of_music -> "pop";
      example ->" The Boxer"].

http://www.groenemeyer.de : Interpret[
      language->"de";
      name->"Herbert Grönemeyer";
      nr -> 432;
      sort_of_music -> "dt_rock";
      example -> "Männer";
      persons -> 5].
```

Fig. 8.10. : Example of Tag-Annotations and resulting F-Logic statements

8.4.4.5 Structure Annotations

In the previous section annotations were introduced that, like HTML-A, annotate only one element content or attribute value. This is a suitable way to go for documents in which only a few elements have to be annotated, or for irregularly structured documents.

For regularly structured XML documents (like enumerations, lists or tables) single tag annotation only may not be suitable. As an example, consider a price list containing 200 products and 600 prices encoded in XML. If the Tag Annotation method is used to annotate this document with semantic information there are 800 elements (200 products and 600 prices) which have to be annotated. For large, regularly structured documents a more efficient method of annotation is desirable, which requires only the embedding of one single annotation for the whole structure. This is the purpose of *Structure Annotations*.

Structure Annotations differ from Tag Annotations in the way object-IDs are constructed. Consider the example shown in Figure 8.11. The XAL parser replaces for each <NAME>-tag all {NAME} expression with the path expression pointing to the <NAME>-tag. In the example Figure 8.11 it results in two annotations. Then *{NAME}.BODY* expressions (where {NAME} is replaced by the path expressions are resolved. Thus two objects will be generated. The object-ID's of these objects

```
<LIST    ID="ID48"   ont1:ONTO = "{NAME} : member
           [first_name = {NAME}.BODY]">
     <NAME> Michael </NAME>
     <NAME> James </NAME>
</LIST>
```

Fig. 8.11. Example of a Structure Annotation

are the tag-ID's or - if there are no ID's - the path in the XML-document to the current <NAME>-element. For example, the object-ID for the second fact is "*http://www.test.com/example.xml|ROOT[0].LIST[0].NAME[1]*". The value of the attribute *first_name* of this generated object is the content of the list element, in this case *James*.

8.4.4.6 Relation Annotations

XAL also provides the ability to define relationships. All primitives that are allowed for Tag- and Structure Annotations are also allowed for the arguments of the relation. Figure 8.12 depicts an example of a Relation Annotation. The name of the relation defined here is *holiday_nr2482* with 4 arguments. The first argument is the keyword PAGE, which is a placeholder for the URI of the document.

```
<holiday  company:ONTO="RELATION(holiday_nr2482(
     PAGE;
     'http://…/mayer.xml';
     '24.10.99';
     '14' ) )" >
```

Fig. 8.12. Example of a Relation Annotation

8.4.4.7 Declaration Annotations

Declaration Annotations provide the ability of define macros. Figure 8.13 depicts an example of a Declaration Annotation. The variable *PAGE* is defined and a value is assigned to it. The keyword

```
<holiday   company:ONTO = "DECLARATION (
     PAGE='frame1';USA='United States of America')">
     ...
     <A  ont1:ONTO="PAGE : class1 [ attrib = USA ]"> ... </A>
```

Fig. 8.13. Example of a Declaration Annotation

PAGE in the last line of Figure 8.13 will not be replaced by the URI of the document but by the defined string *frame1* because the declaration of PAGE takes precedence.

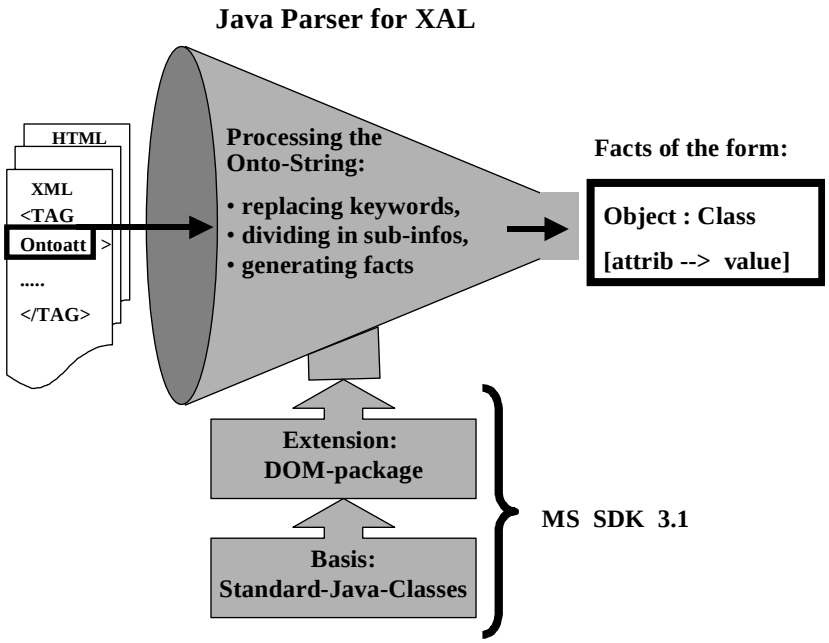


Fig. 8.14. Architecture of the Fact Extractor

8.4.5 An Extractor for XAL

A fact extractor for XAL has been implemented in Java [Schenk, 1999]. The purpose of the extractor is to read an annotated XML or HTML document and to generate a formal representation of the annotation, suitable for the SiLRI inference engine.

The way the extractor is realized is shown graphically in Figure 8.14. The basis for the development of this Java program is Document Object Model (DOM) API[(1)], a recommendation of the W3C for accessing and modifying HTML and XML documents. DOM is a static model for XML and HTML documents, and provides a standardized way to access documents in memory. A streaming API for accessing the documents is not suitable, since the annotation may contain forward and backward references. These references need to be resolved at runtime, so the complete document needs to be accessible (but also implementations of DOM based on secondary storage are available, cf. [Huck et al., 1999]).

The extractor retrieves the string-values of all *Onto-attributes* elements and processes them in the following way: the value of the XML-onto-attribute is decomposed step-by-step into smaller substrings. Then these parts are replaced by the corresponding resource strings fetched from the basic document elements and element contents. That means that keywords will be replaced by parts of the original data. After this process has finished, the facts are stored in the metadata knowledge base.

1. http://www.w3.org/DOM/

8.4.6 Assessment of XAL

XAL is an extension and generalization of HTML-A, while still being backward-compatible. XAL enables the usage of multiple ontologies for the annotation of one document by exploiting XML namespaces. Annotations may use information which is distributed over the document by deploying path expressions based on the W3C XPointer specification. Regular expressions allow for filtering of specific information from text content of tags. The annotation of regular structures is enabled by automated ID generation. XAL provides more expressive data-type than XAL. A parser for XAL was implemented, which extracts F-logic for the SiLRI inference engine from XML documents. In experiments conducted especially the structure annotation facility has been proven to reduce the effort considerably, since a table or list is annotated with just one annotation statement.

XAL complements approaches like RDF (see Chapter 13): XAL integrates ontology-based annotations with XML documents, whereas RDF metadata is separated from the document content. To facilitate the construction of ontology-based annotation of XML documents, a tool similar to OntoPad is necessary, which provides the means for browsing the ontology and inserting annotations.

Chapter 9 The Crawler and Query Interface

The previous chapters presented the *information provider* and *inference aspects* of Ontobroker. This chapter presents the information *consumer aspects* of Ontobroker. The parts of the Ontobroker that deal with consumer aspects are the Information Crawler and the Query Interface. In the following the Information Crawler is discussed first, and then the three different Query Interfaces that Ontobroker possesses are presented.

9.1 The Information Crawler

The crawler is the component of Ontobroker that crawls the Web and fills the database with document metadata from the available sources. [Ashish & Knoblock, 1997] distinguish three classes of Web data sources.

1. *Multiple-instance sources* share the same structure but provide different information, for instance the CIA World Fact Book(1) provides information about more than 200 different countries stored on more than 200 similarly structured pages (one page per country).
2. *Single-instance sources* provide large amounts of data in a structured format.
3. *Loosely structured pages* possess no generalizable structure, for instance personal homepages.

Sources falling into the first two categories (*structured sources*) allow for the implementation of wrappers (cf. [Kushmerick et al., 1997]). For unstructured sources, Ontobroker uses explicit metadata based on HTML-A, XAL or RDF.

9.1.1 Structured Sources

In the case of structured information sources, wrappers [Kushmerick et al., 1997] can be created that automatically extract factual knowledge from Web sources. Examples are specialized shopping agents that use the regularity of online store data to extract product information (cf. [Etzioni, 1997]). Wrappers are also deployed in mediation systems like TSIMMIS for integrating information from heterogeneous information sources (cf. [Hammer et al., 1997]). Such wrappers can be used to directly extract the factual knowledge that is used by the inference engine of Ontobroker. In this scenario a wrapper replaces the annotation process and the process of translating annotations into facts. Ontobroker deployed wrappers for the CIA World Fact Book to extract factual knowledge from this multiple-instance source. An ontology was developed that could be used to describe the provided knowledge and to formulate queries about it. An extract of the ontology used is shown in Figure 9.1.

[Dagasan & Witt, 1998] describe a wrapper that extracts knowledge from the fact book and translates it into facts formulated according to the ontology. When building the wrapper two problems were encountered:

1. see CIA, The World Factbook, http://www.cia.gov/cia/publications/factbook/

```
land[name =>> string;
     geography =>> geography;
     people =>> people;
     government=>> government;
     economy =>> economy;
     communication =>> communication;
     transportation =>> transportation].

people[population =>> string;
     age_structure => age_structure;
     birth_rate =>> string;
     death_rate =>> string;
     net_migration =>> string;
     sex_ratio =>> sex_ratio;
     ...].

geography[map =>> string;
     flag =>> string
     location =>> string;
     coordinates =>> string;
     area =>> area;
     boundaries =>> boundaries;
     coastline =>> string;
     maritime_claims =>> maritime_claims;
     climate =>> string;
     terrain =>> string;
     elevation_extremes =>> elevation_extremes;
     natural resources =>> string;
     land_use =>> land_use;
     irrigated_land =>> string;
     natural_hazards =>> string].
```

Fig. 9.1. Wrapper Ontology for the CIA World Factbook

- The authors of the fact book used HTML as a layout language rather than as a markup language. For instance, headings that indicate sections are not made explicit by using the <Hi>...</Hi>-tag of HTML. Instead tags that simply define the rendering style of text (like bold, italic etc.) were used.
- The pages are generated manually, i.e., starting from the same format frame small differences (missing subjects, different typing styles etc.) exist between different pages.

As a consequence, a wrapper must be able to handle variations in the structure of the pages (cf. the pattern matching approach of TSIMMIS [Hammer et al., 1997] or the error-tolerant parser of Jedi [Klein & Fankhauser, 1997]. The wrapper developed for Ontobroker matches keywords appearing in headings and maps the following paragraphs as values for attributes defined in the ontology. The implemented wrapper extracted more than 3 MBytes of factual knowledge from the fact book which can then be used to answer queries with Ontobroker.

The Ontobroker extension described in [Erdmann & Studer, 2001], [Erdmann, 2001] that enables Ontobroker to deal with ontology-based XML files can also be understood as a wrapper: it wraps XML data and translates it back into F-logic facts.

9.1.2 Unstructured Sources

In the unstructured document case, Ontobroker requires that documents are annotated with HTML-A or XAL. The task of the information crawler is to find pages and extract the annotated metadata information. Ontobroker aims only at *semantic island*s that adhere to a certain ontology. Therefore, the crawler does not - as conventional search engines do - crawl the complete Web. Instead, the crawler maintains an index of information providers. Information providers are able to sign up for this index. An information provider provides the URI to an index page, which contains all URIs the information crawler is supposed to visit and to extract metadata from. During each crawl the crawler gets all the index pages and downloads all the pages listed on those index pages. Then the crawler extracts the information and integrates them into the Ontobroker knowledge base. When integrating the new information it overwrites the old information from previous crawls. Crawling is repeated periodically. It is also possible to initiate the crawling manually. It is assumed that the crawler just deals with metadata according to one specific ontology.

9.2 Formal-Language-based Interface

The formal-language-based interface for Ontobroker is aimed at the expert user. The input form (Figure 9.2) requires a user to type in a query in F-logic syntax. The query in Figure 9.2 asks for the object-ID of researchers with last name *Benjamins* and their email addresses. After submitting the query to the Web server the Ontobroker system returns a set of answer substitutions. Since object-IDs are represented as URIs, these URIs can be directly used as links to identifiable content.

9.2.1 Evaluation of the Text-based Query Interface

The text-based query approach has serious drawbacks:

- Users have to know the syntax and semantics of the query language, since they have to exactly type in the syntactical expressions. However, users are not willing to learn a query language for occasional usage. Furthermore, people make mistakes when typing text. Since Ontobroker does not aim to serve only expert users, this is devastating for the acceptance of Ontobroker.
- The user also has to know the ontology when formulating a query. This is an even bigger problem than the lack of knowledge of the query language: without knowledge of the ontology one cannot formulate a useful query, because all knowledge is organized by the ontology, which may contain several hundred concepts and attributes.

However, the text-oriented interface is less restrictive than graphical interfaces - it provides the full power of the query language. The interface enables for combination of queries about the ontologies with queries about instances. Examples include queries like "*Which instances of subclasses of person cooperate with researchers?*", which is represented in F-logic as follows:

FORALL per <-

EXISTS res,subc subc::Person AND NOT subc = Person AND

per:subc[cooperatesWith->>res:Researcher]].

Such queries are not possible with the other two query interfaces that will be presented in the following sections.

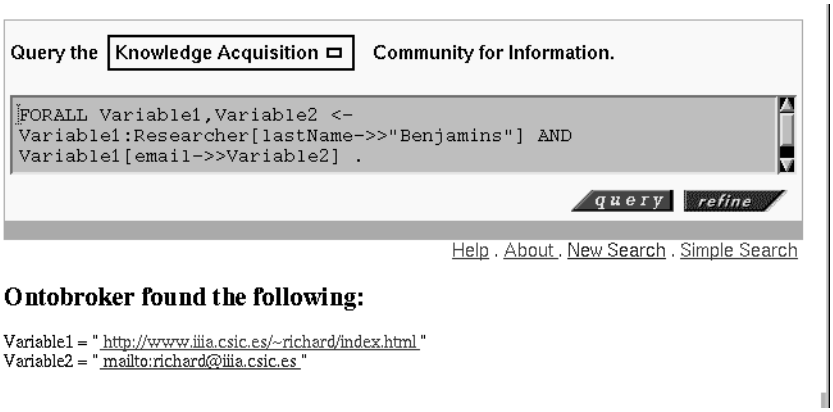


Fig. 9.2. Text-based Query Interface for Ontobroker

9.3 Form-based Interface

This section presents an interface which remedies the drawbacks of the formal-language based interface. The interface guides the user by providing a graphical method to define queries and to browse the ontology.

9.3.1 Approach

To remedy the drawbacks mentioned in the previous section, a graphical user interface was constructed. The following requirements for a graphical user interface were identified:

- The user should not be required to learn the syntax of the query language. Thus, a graphical interface should lead the user and should help to avoid mistakes.
- The user should not be required to know the classes and attributes of the ontology. An interface should present the ontology and allow for the easy integration of concepts and attributes from the ontology in a query.

To remedy the first drawback the regular structure of the query language is exploited. The structure of an elementary expression is:

Object:Class[Attribute->>Value]

The structure provides guidance when designing a query interface. Each part of the above depicted elementary expression can be related to an entry field. Possible values of the entry field may then be selected from a menu (e.g., variable names). The capability to select entries frees users from typing and understanding logical expressions as much as possible. The simple expressions can then be combined by logical connectives (*AND*, *OR*, *AND NOT*) as shown in Figure 9.3 which asks for researchers with the last name *Benjamins* and their email addresses. Each time a new logical connector has been selected, a new row is added to the tabular interface. The *AND* and *AND NOT* connector realizes joins over the stored data.

This does not resolve the second drawback: users also need support for selecting classes and attributes from the ontology. To allow browsing and the selection of classes, the ontology has to be presented in an appropriate manner. Ontologies may become quite large, and can often be regarded as large structured graphs or trees. Usually an ontology can be represented as a large hierarchy of concepts. Concerning the handling of the hierarchy, a user has two requirements: first is the need to

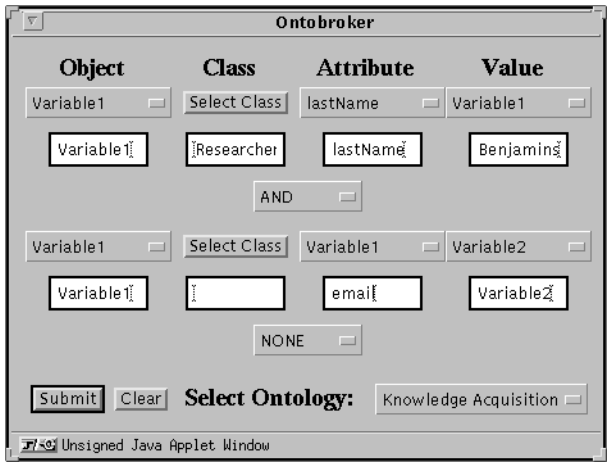


Fig. 9.3. Tabular Query Interface for Ontobroker

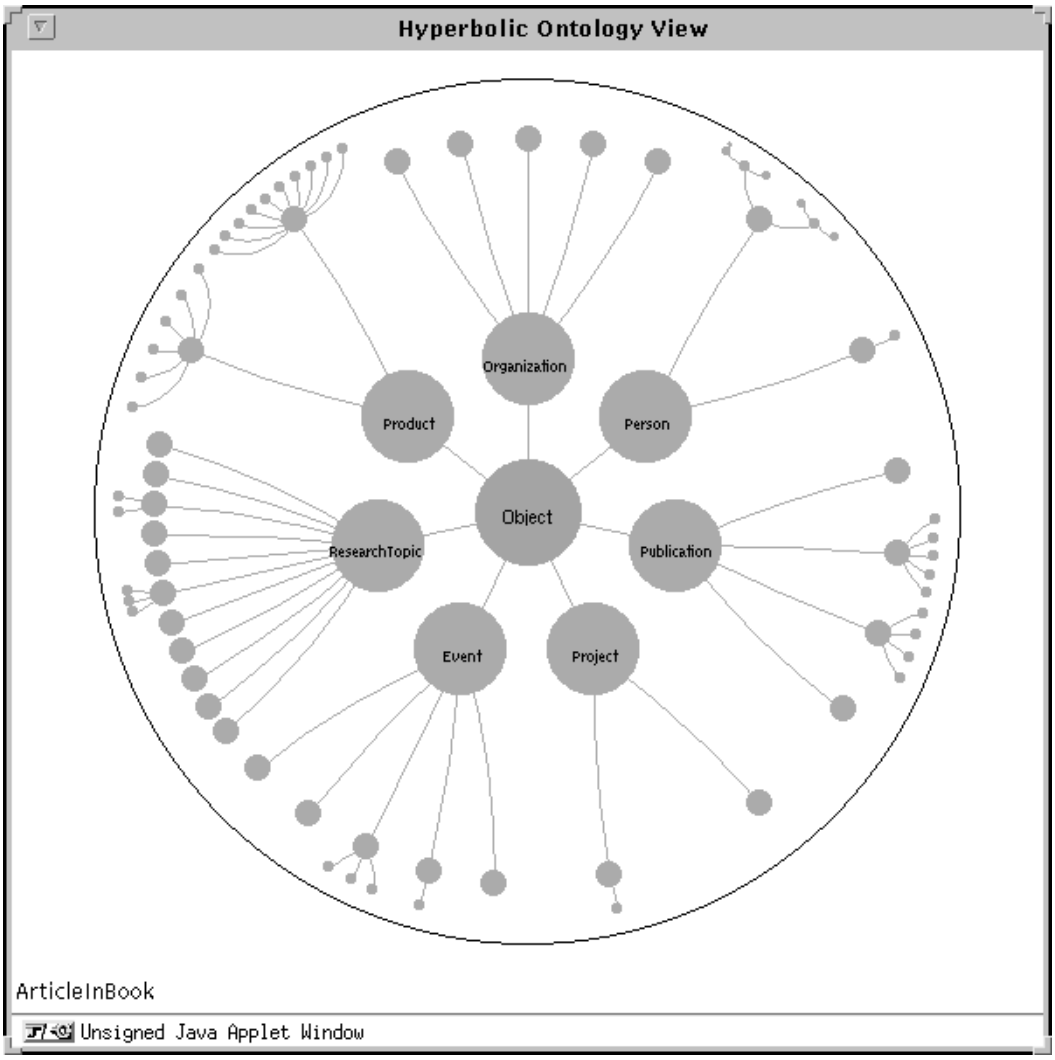


Fig. 9.4. Initial View of the Hyperbolic Browser with the Root in the Center, Showing the (KA2) Ontology[(1)]

scan the vicinity of a certain class, looking for classes better suited to formulation of a certain query. Second, a user needs an overview of the whole hierarchy to allow easy and quick navigation from one class in the hierarchy to another class. Browsing technologies that allow for these kinds of browsing are known as *focus and context* browsers. Ontobroker deploys a focus and context browser based on Hyperbolic Geometry,[(1)] described by [Lamping & Rao, 1996]. The hyperbolic browser initially displays a tree with the root at the center (see Figure 9.4). By drag-and-drop operations other nodes can be brought into focus. The amount of space available to a node falls off as a continuous function of the distance in the tree from the node in focus. The context always includes several generations of parent, sibling and offspring nodes, which makes it easier for the user to browse the hierarchy without getting lost. The hyperbolic browser supports effective interaction with much larger hierarchies than conventional hierarchy viewers. [Lamping & Rao, 1996] claim that in a 600 pixel by 600 pixel window, a standard 2D hierarchy browser can typically display 100

1. The hyperbolic browser in Ontobroker view is based on a Java-profiler written by Vladimir Bulatov (cf. http://www.physics.orst.edu/~bulatov/HyperProf/index.html).

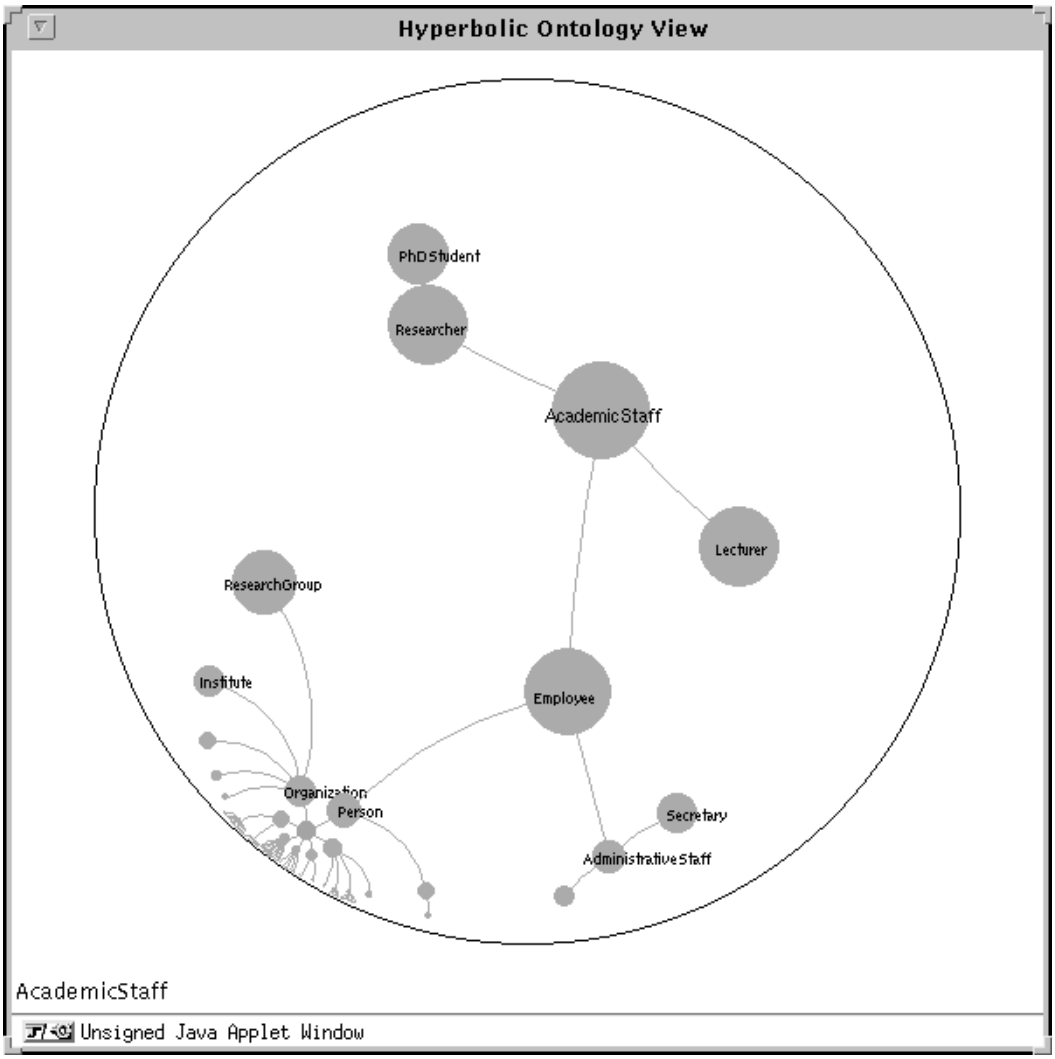


Fig. 9.5. Hyperbolic Browser after a Drag Operation

nodes. The hyperbolic browser can display 1000 nodes of which about the 50 nearest the focus can show up to 20-40 characters of text. Thus, the hyperbolic browser can display up to 10 times as many nodes while providing more effective navigation.

The visualization technique allows for quick navigation to classes far away from the center as well as for close examination of classes and their vicinity. Figure 9.5 shows the hyperbolic browser after a (continuous) drag operation.

A problem for browsing ontologies with the hyperbolic browser is that the browser is not able to display arbitrary graphs very well since concept hierarchies may be graph structured instead of tree structured. However, it is always possible to turn a graph into a tree. In the case of multiple inheritance copying the graph for each parent node has been chosen. Thus the same subgraph may be browsed from different parent nodes.

9.3.2 Using the Interface

This section illustrates how the form-based interface is used to construct the query: "*What is the email address of the researcher with last name 'Benjamins'?"* It is assumed that the hyperbolic browser shows the (KA^2) Ontology (see Appendix D for a complete listing of the (KA^2) ontology).

The users select a variable from the choice menu directly below the label *Object* (see Figure 9.3). The variable is a place holder for the object ID. The next step is to select an appropriate class from the hyperbolic ontology view. Because the user ideally selects the most specific class describing the real world person *Benjamins,* the user navigates through the ontology part describing *persons*. This is done by simply clicking on one node representing a class and dragging the node to another location inside the surrounding circle. The rest of the hierarchy is drawn according to the new position of the dragged node. The most specific class describing Richard Benjamins is *Researcher.* Therefore the user clicks on the node containing the string *Researcher.* In the query window the string *Researcher* appears in the class section and all the attributes defined for the concept *Researcher* are available for selection in the attribute choice menu.

To determine the object ID of the researcher with the last name *Benjamins* the user has to select the attribute *lastName* out of the list of all available attributes, which are determined through the selected class. Finally, he just has to fill in the name *Benjamins* in the field value, because this is the value of the attribute *lastName* of the object being investigated. Furthermore, since the user wants to ask for the email-address of this researcher the need arises to ask for the attribute value. Because the current interface supports only one attribute-value pair of objects per row, the user needs to enlarge the query window. This is done by selecting an operator from the choice menu in the center of the applet window (with the label *NONE*). The user selects *AND* since they are looking for the email address of the same researcher. After selecting the appropriate connector the window is enlarged and provides an additional row. In the new row the query part asking for the email address has to be provided. The query part is constructed by first selecting the same variable that was selected for the *Object* column of the first row, since the same object is being searched for. Furthermore, to have the attributes available in a way that the user can select an appropriate attribute, they also need to select the class *Researcher* in the Hyperbolic Ontology Browser. Because the user wants to know the value of the attribute *email* they select this attribute in the *Attributes Choice* menu. Finally the user needs to select a new variable for the value slot of the second row, which is used for the email address. Then the query can be submitted to the Ontobroker query engine.

9.3.3 Suitability of the Hyperbolic Browser for Browsing Ontologies

[Pirolli et al., 2000], [Pirolli et al., 2001] conducted with eye tracking studies an extensive evaluation of the browsing performance of the Hyperbolic Browser compared to a conventional tree visualization of information used by a.o., Microsoft's File Explorer (see Figure 9.6), on a variety of tasks and tree information. They reported that on average (that means a wide distribution of tasks and information types) the browsing performance of the hyperbolic tree was not significantly different than the browsing performance with Windows Explorer. However, different results were obtained when the notion of *information scent,* developed in [Pirolli & Card, 1999], was taken into account. [Pirolli et al., 2000] point out that:

> *"Information scent, developed in information foraging theory can be used to predict the circumstances under which the attentional spotlight will be affected by the density of information in a visualization. Information scent is provided by the*

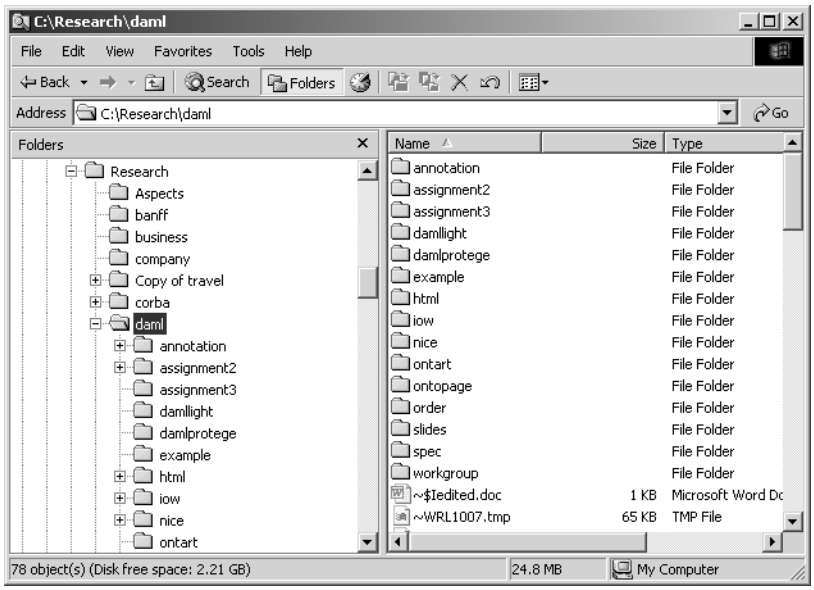


Fig. 9.6. Windows Explorer Tree View

proximal cues perceived by the user that indicate the value, cost of access, and location of distal information content. In the context of foraging for information on the World Wide Web, for example, information scent is often provided by the snippets of text and graphics that surround links to other pages. The proximal cues provided by these snippets give indications of the value, cost, and location of the distal content on the linked page. Computational modeling of human information foraging suggests that users' browsing choices are based on the evaluation of information scent."

For information sources with a high information scent, [Pirolli et al., 2000] reported that these sources are more easily browsed with a hyperbolic browser than sources with a low information scent. The Hyperbolic browser requires only 0.92 sec/level in the hierarchy compared to 1.75 sec/level or 53% as long - it allows the user to access a target faster if the user knows where it is or at least the path that it is on (that is, if there is a strong information scent)—about twice as fast as for the Explorer-style browser. If the user must engage in a visual search, it is possible to search more nodes/sec. Practice or expertise has a strong effect on performance when scent is low.

By transferring the theory of information scent to ontologies we are able to make the following oberservations: an ontology aims to represent the terms of a certain domain in a logical, intuitive way. For instance, when constructing a query about *Richard Benjamins* using the ontology depicted in Figure 9.4 it is intuitively clear that the path to find the most specific description of *Richard Benjamins* leads through the node labeled *Person* and not through the node labeled *Event*. Therefore, the ontology provides a high information scent: a clear separation of concepts and, given a user with some domain knowledge, a clear path to the most specific concept describing a certain instance, in this case *Richard Benjamins*.

Given the argument that a high-quality ontology should represent the terms of a certain domain in a logical, intuitive way, the notion of information scent provides a quality criteria for a good ontology. We draw the following specific conclusions for ontologies, based on the evaluation of [Pirolli et al., 2001]:

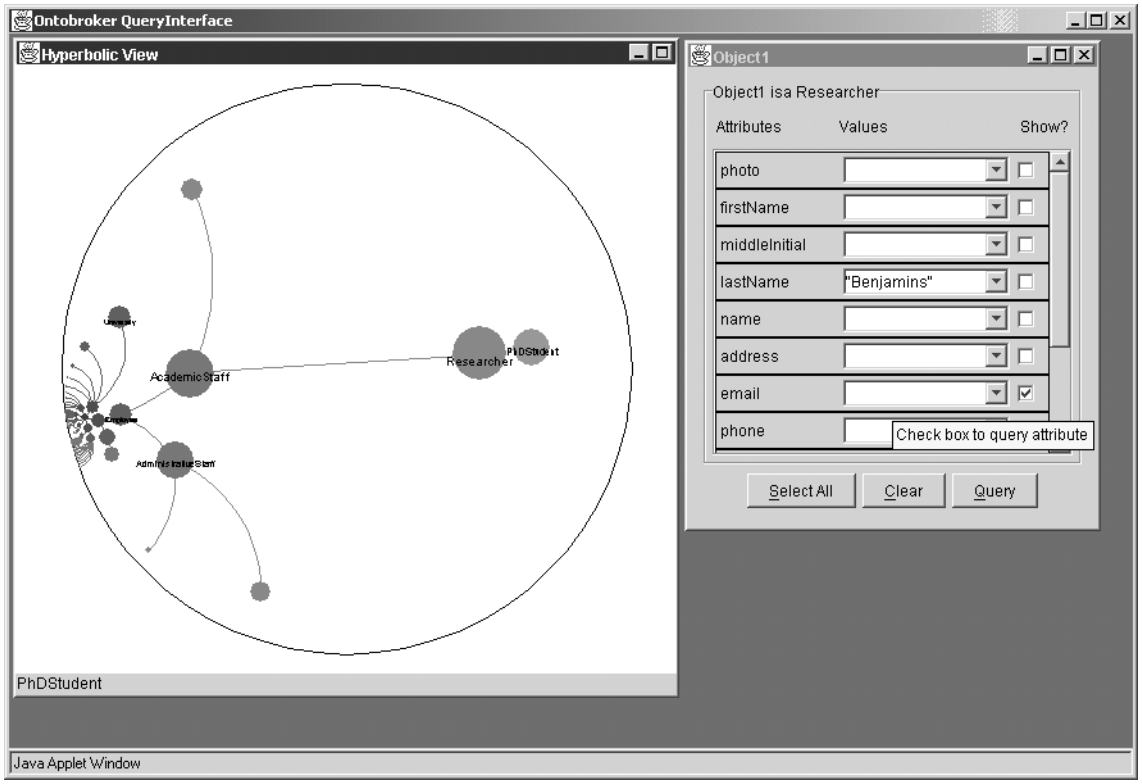


Fig. 9.7. Object Interface with Hyperbolic Ontology Browser

- The hyperbolic browser is a better visualization tool for 'high quality' domain ontologies than conventional visualization mechanisms like tree viewer, provided the user has an intuition about the domain. 'High quality' means that the domain ontology reflects the intuitive understanding of a domain expert.
- Vice versa, the browsing performance of ontologies with the hyperbolic browser can be used as a measure for judging the quality of an ontology. Since quantitative measuring of the quality of an ontology is as yet an unsolved task, this poses an exiting possibility for future research.

9.4 Object-based Query Interface

The tabular interface introduced in Section 9.3 has advantages compared to the text based interface. However, it still does not exploit the conceptual, object-oriented model of the ontology. Instead, the tabular interface is modeled after the syntax of basic F-logic expressions, and sophisticated queries which involve several conditions are hard to assemble. To simplify the query construction even further a third query interface was implemented, which takes advantage of the object-oriented conceptual model of the ontology and the instance data. Figure 9.7 provides a screenshot of the object-oriented query interface of Ontobroker. The hyperbolic browser is used to browse the ontology. A double click on a concept (e.g., *Researcher*) opens an *instance template*. Arbitrarily, many instance templates may be opened. Each instance template is assigned an object-ID, which serves as a variable. Users may now either fill in known information in the text box next to the attribute names (e.g., *Benjamins* in the text box for the attribute *lastname*) or select object-ID of other created instance templates, if the value of an attribute is a complex object. Attribute values that

are queried need to be marked (like the *email* attribute in Figure 9.7). This interface, although it is much closer to the conceptual model, offers less expressiveness than the text and form interface. The interface only allows the construction of conjunctive queries with existential quantification.

9.5 Related Work

In the database community a plethora of visual query languages have been defined. The form-based and the tabular interface borrow heavily from the Query-By-Example (QBE) approach [Zloof, 1977], since entry fields are used which are generated by the ontology. Close to current approach are VOODOO [Fegaras, 1999] and PESTO [Carey et al., 1996]. VOODOO is a simple visual language to express ODMG OQL queries. The language is expressive enough to allow for most types of query nesting, aggregation, universal and existential quantifications, group-by, and sorting. PESTO [Carey et al., 1996] provide an integrated user-friendly interface for both query formulation and data browsing. The PESTO query language does not support query nesting. However, integrating data browsing with query formulation is highly desirable because data can provide a very useful feedback to the query formulation process itself.

[Cruz, 1992] presents DOODLE, a complete visual language for F-Logic, which allows for integration of user defined visualizations on a class-by-class basis. DOODLE allows for the expression of very complex F-Logic programs, whereas the current focus was on simple graph-based data. Thus, the visualizations presented here are much simpler than DOODLEs. DOODLE uses variables to perform joins and existential quantifications, similar to the current approach.

Joins and existential quantifications are handled in a similar manner to DOODLE. The main difference from all the mentioned visual database query languages is the integration of an ontology/ schema browsing mechanism into the query interface. Furthermore, the current research did not aim to provide a complete query language for databases, since simplicity was an explicit design criteria.

Chapter 10 (KA^2) - A Knowledge Management Application of Ontobroker[1]

This chapter provides an explanation of how to use Ontobroker for Knowledge Management and presents as a case study: the Knowledge Annotation Initiative of the Knowledge Acquisition Community, $(KA)^2$ [Benjamins et al., 1998a]. The relevant issues involved are presented and discussed in the context of this Knowledge Acquisition research community. One of the reasons for choosing a research community for a case study has been that many researchers already have homepages and often put information related to their research areas on the Web. Moreover, researchers are often enthusiastic participants in practical experiments. In addition, Knowledge Engineering is a field that, during the past 17 years, has been concerned with capturing, analyzing, organizing, structuring, representing, manipulating and maintaining knowledge in order to obtain intelligent solutions for hard problems [Studer et al., 2000]. Thus, the KA community was a good candidate for experiments with Ontobroker in the area of Knowledge Management.

10.1 Using Ontobroker for Knowledge Management

The Ontobroker approach comprises three main subtasks: (1) ontological engineering to build an ontology of the subject matter, (2) characterizing the knowledge in terms of the ontology, and (3) providing intelligent access to the knowledge. Ontobroker captures distributed, rather than centralized information. The information is directly accessed at its original location (in HTML pages) rather than being separately housed in a database. The approach allows the user to infer knowledge that is not explicitly known, but that can be deduced based on a domain knowledge as captured by the ontology. For example, in the context of human resource management if, in a particular company, only senior managers can lead projects and Mr. Paton is project leader, then it can be deduced that Mr. Paton is a senior manager, even though this is not stated explicitly anywhere.

Figure 10.1 gives a general overview of the Ontobroker approach to Knowledge Management. An ontology of the subject matter has to be built, which is used to characterize the subject matter (i.e., to fill the ontology with instances). An intelligent Web crawler receives a query in terms of the ontology, consults the subject matter (the instances), interprets them using the ontology and generates an answer. The instances (the actual knowledge to be managed) are distributed over different HTML pages (of an intranet or the Internet).

There are different uses of ontologies for the purpose of Knowledge Management. Within the Ontobroker approach to Knowledge Management ontologies are used for building, maintaining and querying a distributed community knowledge map. In order for our approach to work in a particular organization, we assume that it has an Intranet/Extranet or access to the Internet and that each member of the organization has a browser. Many companies already have an intranet, which is an

1. The chapter is partially derived from [Benjamins & Fensel & Decker & Gomez-Perez, 1999].

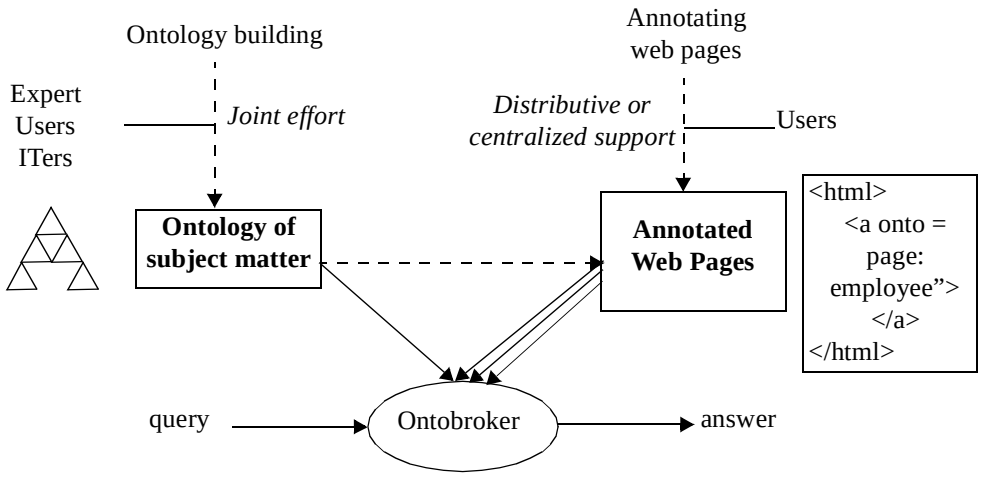


Fig. 10.1. The Knowledge Management Approach

easy-to-use infrastructure that gives companies access to a large variety of Internet techniques. Therefore, for users already familiar with browsers, this approach has a short learning curve. In addition, the approach requires that the knowledge of interest is available in HTML pages on the net, or in a format from which pages such as databases or text documents can be generated by wrappers [Kushmerick et al., 1997].

10.1.1 Ontobroker and the Knowledge Management Cube

To classify this approach, Knowledge Management Cube (KMC) may be recalled, as derived in Section 3.3 (see Figure 3.6). In the following some of the sub-cubes of the KMC are discussed.

- *Knowledge goals*. The goal of using an Ontobroker-like technology for Knowledge Management is to make distributed information accessible and queryable with the effect of achieving a better knowledge for members and outsiders of what is going on within the community.
- *Knowledge identification* has two facets: the *engineering of the ontology* and the *annotation of the HTML pages*. From a human perspective, experts need to be identified who are able and willing to define the ontology and people need to be identified who are capable of providing information. A further step requires the identification of documents, which contain the knowledge that is suitable for formalization, or even databases.
- *Knowledge acquisition* possesses the same two facets: the *engineering of the ontology* and the *annotation of the data*. From a human perspective, ontology engineering means that subject matter experts have to derive an ontology about the domain, and annotation means that the annotation provider needs to have an incentive for providing the information. From a *technology* perspective within Ontobroker only the annotation step is supported with the HTML annotation editor. However, for Ontology construction tools like the Ontolingua server [Farquhar et al., 1997], Protégé [Eriksson et al., 1999], and OntoEdit [Sure & Studer, 2001] may be used.
- *Knowledge dissemination*: From the technology perspective the Ontobroker system is used to disseminate the knowledge. Currently Ontobroker is a *pull* technology: Knowlege has to be requested. Turning Ontobroker into a *push* technology by linking Ontobroker's domain ontology to a workflow ontology has been investigated in [Staab & Schnurr, 2000].

- *Knowledge utilization* takes place during the research activities of researcher, which query the Ontobroker engine and use the delivered results for their daily work.
- *Knowledge assessment* helps to maintain the ontology and to judge the instance data.

The organizational aspects are not discussed here, because an organization in a strict sense does not exist in this scenario. Also, management aspects were not incorporated in the classification above, but did appear in the course of the project.

In Section 4.1 two kinds of viewpoints on IT contributions for Knowledge Management were introduced, namely:

- the *process-centered* view, and
- the *product-centered view.*

The current approach covers both aspects (process and product-centered). The building of an ontology is a social process, in which different stakeholders have to agree on a shared terminology. Such a process has to be defined, and the right abstractions have to be found that enable the building of supporting tools for such kinds of processes.

Furthermore, in Section 4.2.1 three kinds of ontologies were introduced, which are useful for organizational memory information systems (see Chapter 4). They are recalled here briefly:

- An *information ontology* describes the information meta-model, e.g., the structure and format of the information sources. This is the lowest level of ontology.
- A *domain ontology* is used to describe the content of the information source. This is exactly the kind of ontology being aimed at.
- An *enterprise ontology* is used for modeling business processes. Its purpose is to model the knowledge needs in a business process in order to describe a process context, which enables active knowledge delivery.

The usage of the ontology itself for knowledge dissemination results in an organizational memory information system. With respect to the above-mentioned three types of ontologies, Ontobroker only deals with the domain ontology. Explicit information ontologies are not needed, since Ontobroker has only two fixed sources: annotated HTML-pages and collections of similarly structured HTML-pages. Similarly, structured HTML-pages all use the same HTML layout to present their information, so that it pays to construct a wrapper that makes the information of the Web pages available to an organizational memory information system. A typical example for such a collection of Web pages is the CIA World Factbook[(1)]. Furthermore, because we are not dealing with an explicit organization, the enterprise ontology is not needed. However, if the system described later is integrated into a specific workflow, the use of this kind of ontology would enable the active delivery of knowledge relevant to the context in any specific business process.

10.1.2 Ontological Engineering

In order to come up with a consensual ontology of a particular domain, it is necessary that the people who have to use the ontology have a positive attitude towards it. Dictating the use of a particular ontology to people who have not contributed to it is not likely to succeed. Preferably, an ontology is

1. See http://www.cia.gov/cia/publications/factbook/

constructed in a collaborative effort by domain experts, representatives of end users and IT specialists. Such a joint effort requires (1) the use of a methodology that guides the ontology development process and (2) tools to inspect, browse, codify, modify and download the ontology.

In a human resource management context, a joint effort typically results in classes like *employee*, *manager, project leader, skill,* and *area of expertise*. Applied to a concrete company, an ontology can fulfill the role of an *enterprise knowledge map*. Notice that terms from the human-resource management ontology are reusable for any kind of company.

10.1.3 Characterizing the Knowledge

In the current approach, the knowledge to be managed is distributively organized in HTML pages (e.g. in a company's intranet or on the WWW). The relevant knowledge can thus be maintained autonomously by different persons responsible for the respective HTML pages. The subject matter knowledge within the HTML pages is annotated using the ontology as a scheme for expressing metadata. For example, in the human resource management domain, the homepage of Mr. Paton would state that he is a project leader. The addition of metadata using HTML-A makes this explicit. The ONTO-attribute of HTML-A does not affect the visualization of HTML documents in standard Web browsers such as Netscape or Explorer. But the ONTO-attribute makes visible the subject matter knowledge for the Ontobroker's Web crawler. This small extension of HTML has been chosen to keep annotation as simple as possible. Also, it enables the direct usage (actually, reuse) of textual knowledge already in the body of the anchor. This prevents the knowledge annotator from representing the same piece of information twice. This simple solution suffices for the current approach because the HTML pages only contain factual knowledge.

10.2 The $(KA)^2$ Initiative

In order to investigate the feasibility of the Ontobroker approach, a large-scale Knowledge Management initiative was conducted, where the subject matter was the scientific knowledge acquisition community: the Knowledge Annotation Initiative of the Knowledge Acquisition Community.

From an organizational perspective the Knowlege Acquisition communities are a virtual organization consisting of researchers, universities, projects, publications, etc. The information resides on the World Wide Web in the homepages of the KA researchers where they publish information about their affiliation, projects, publications, research interests, etc.

From a concrete Knowledge Management point of view, the $(KA)^2$ initiative is by no means an esoteric, academic toy example. Imagine a large multinational company with thousands of employees worldwide. For such a large organization, effective human resource management (HRM) is of vital importance. However, finding '*who knows what*' in large organizations has always been a time-intensive process.(1) A Knowledge Management system that allows the HRM team to find adequate people based on their skills, experience and area of expertise is of high value. For large companies that have an organization-wide intranet, the current approach is a real possibility for enhancing the HRM task. It allows improvement of the precision and presentation of the results of searches on an intranet or the WWW.

1. "*If Only HP knew what HP knows...*", see [Davenport, 1997]

```
Class hierarchy (10 classes defined):
    Person
       Employee
          Academic-Staff
             Lecturer
             Researcher
          Administrative-Staff
             Secretary
             Technical-Staff
       Student
          Phd-Student

23 relations defined:

  Address, Affiliation, Cooperates-With, Editor-Of,
  Email, First-Name, Has-Publication, Head-Of-Group,
  Head-Of-Project, Last-Name, Member-Of-Organization,
  Member-Of-Program-Committee, Member-Of-Research-Group,
  Middle-Initial, Organizer-Of-Chair-Of, Person-Name,
  Photo, Research-Interest, Secretary-Of, Studies-At,
  Supervises, Supervisor, Works-At-Project
```

Fig. 10.2. $(KA)^2$ Person Ontology

Notice, however, that the fact that $(KA)^2$ is naturally related to the HRM task does not imply that it is limited to this Knowledge Management task. In principle, the subject matter of the Ontobroker approach can concern itself with any kind of company-vital knowledge that needs to be managed more effectively.

10.2.1 Ontological Engineering in $(KA)^2$

$(KA)^2$ requires an ontology of the KA community (an *enterprise knowledge map*). Since an ontology should capture consensual knowledge, in $(KA)^2$ several researchers cooperated together - at different locations - to construct the ontology. This ensured that the ontology would be accepted by a majority of KA researchers. The design criteria used to build the $(KA)^2$ ontology were: modular, to allow more flexibility and a variety of uses, specialization of general concepts into more specific concepts, classification of concepts by similar features to guarantee inheritance of such features, and standardized name conventions. The current ontology for the KA community consists of 7 related ontologies: an *organization* ontology, a *project* ontology, a *person* ontology, a *research-topic* ontology, a *publication* ontology, an *event* ontology, and a *research-product* ontology. To make this ontology accessible to the entire community, a decision was made to translate this (F-logic) ontology into Ontolingua [Farquhar et al., 1997] and to make it accessible through the Ontology Server. Translation between the F-logic and Ontolingua formats of the ontology is performed automatically by translators, which are part of ODE (Ontological Design Environment [Fernández & Gomez-Perez, 1999b]). These translators ensure that the F-logic and Ontolingua versions of the ontology are always in sync. For illustration purposes, examples are included here of two sub-ontologies of the KA ontology: the person ontology and the publication ontology.(1) The Person ontology (see Figure 10.2) defines the types of persons working in academic environments, along with their characteristics. This ontology defines 10 classes and 23 relations. The overview does not

1. The full $(KA)^2$ ontology represented in SiLRI F-logic notation can be found in Appendix D.

```
Class hierarchy (13 classes defined):

    Online-Publication
    Publication
       Article
          Article-In-Book
          Conference-Paper
          Journal-Article
          Technical-Report
          Workshop-Paper
       Book
       Journal
          IEEE-Expert
          IJHCS
          Special-Issue

28 relations defined:

  Abstract, Book-Editor, Conference-Proceedings-Title,
  Contains-Article-In-Book, Contains-Article-In-Journal, Describes-Project,
  First-Page, Has-Author, Has-Publisher, In-Book, In-Conference,
  In-Workshop, Journal-Editor, Journal-Number, In-Journal
  Journal-Publisher, Journal-Year, Last-Page,
  On-Line-Version, Online-Version-Of, In-Organization
  Publication-Title, Publication-Year,
  Technical-Report-Number, Technical-Report-Series,
  Type, Volume, Workshop-Proceedings-Title
```

Fig. 10.3. $(KA)^2$ Publication Ontology

show which classes the relations connect to. Indentation denotes the *subclass-of* relation. The publication-ontology (see Figure 10.3) defines - in 13 classes and 28 relations - the usual bibliographic entities and attributes.

10.2.2 Annotating Pages in $(KA)^2$

Annotating HTML pages in $(KA)^2$ means that each participating researcher in the KA community has to annotate the relevant knowledge in his or her homepage environment using HTML-A. Experience has shown that it takes roughly one hour to annotate five pages. At the Ontobroker site, an annotation checker is available to detect syntax errors. An important factor that motivates people to annotate their pages is self-publicity. By annotating pages, researchers make themselves more visible to others, which enhances the likelihood that others will use and refer to their work, which - in the academic world - is a good thing. The HTML-A annotation of about 30 researchers results in about 1800 facts.

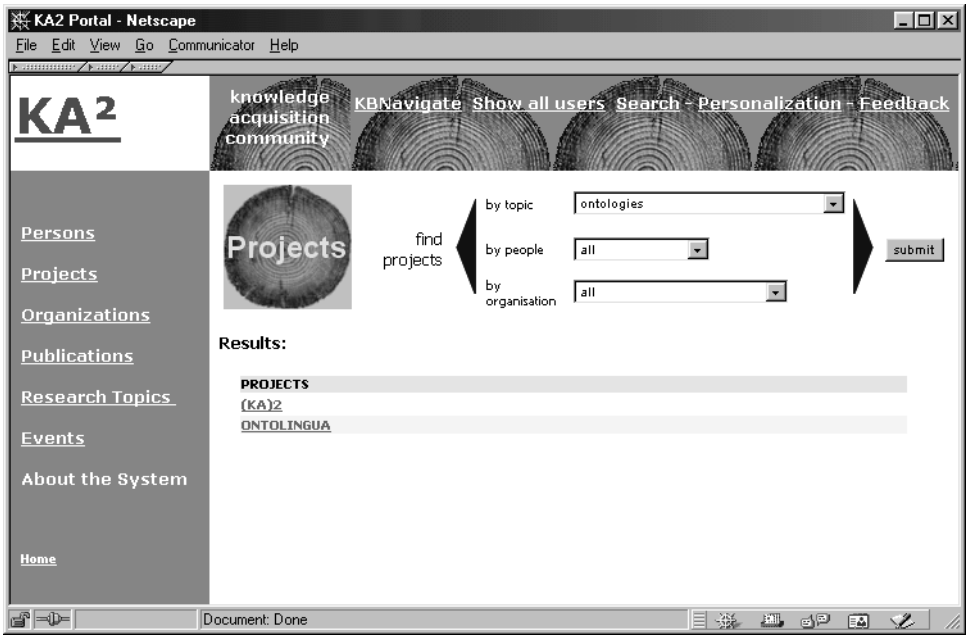


Fig. 10.4. $(KA)^2$ Portal

10.2.3 Querying the KA Community

In $(KA)^2$ researchers were required to register their pages. That is, they had to tell Ontobroker's crawler which URLs it needed to visit. Once that was done, knowledge retrieval was possible using the interfaces described in Chapter 9.

Possible queries include asking for all researchers in the KA community. The answer would not only include researchers who have their homepage annotated, but also additional researchers who cooperate with these researchers. The ontology defines cooperation between researchers, which enables the following deduction: *if X cooperates with Y, then X and Y must be researchers*. Ontobroker uses this type of information, not for consistency checking (which would not be a very good idea in an open Web environment), but for deriving new facts (for instance, Y is also a researcher). This example illustrates that it is possible to access knowledge that is not explicitly represented, which is an important advantage of the current approach compared to keyword-based search. All researchers who have worked together on some project could be asked for, or for abstracts of all papers on a particular topic.

[Staab et al., 2000] describe an extension of the Ontobroker approach towards a *Semantic Community Web Porta.l*[(1)] Instead of just providing query mechanisms the Web Portal provides a browsable interface to the database, thereby exploiting the structure of the ontology (see Figure 10.4). The query language is completely hidden from the Web site user. Although the content of the knowledge base and most of the underlying technology stays exactly the same, the improved access mechanisms enable a much better exploitation of the available information. [Erdmann, 2001] presents additional access mechanisms, e.g., a tool called *Yahoo-a-lizer*, which presents the $(KA)^2$ Ontology in a Yahoo-like way, or *RDFMaker*, which generates an RDF metadata representation of the available instance data, thereby exploiting the rule definitions available in the ontology.

1. see http://ka2portal.aifb.uni-karlsruhe.de/

10.3 Feasibility of Ontobroker as a Knowledge Management System

To judge the feasibility of the Knowledge Management approach applied in $(KA)^2$, the risks involved must be considered. Risks come from various resources, and will be discussed resource-wise, i.e., technological risks, social and organizational risks.

10.3.1 Technological Risks

From a technology point of view, there are several factors that endanger the success of the Knowledge Management approach.

Tool support

First of all, such an initiative is likely to fail without dedicated tools to support the tasks involved. In particular, tools are needed for (1) constructing and maintaining the ontology, (2) annotating information sources, and (3) querying them (see Figure 10.5). Prototype tools for annotation and querying have been discussed. Although those tools are fairly stable, they lack commercial quality. Despite this, several commercialization attempts are on their way.

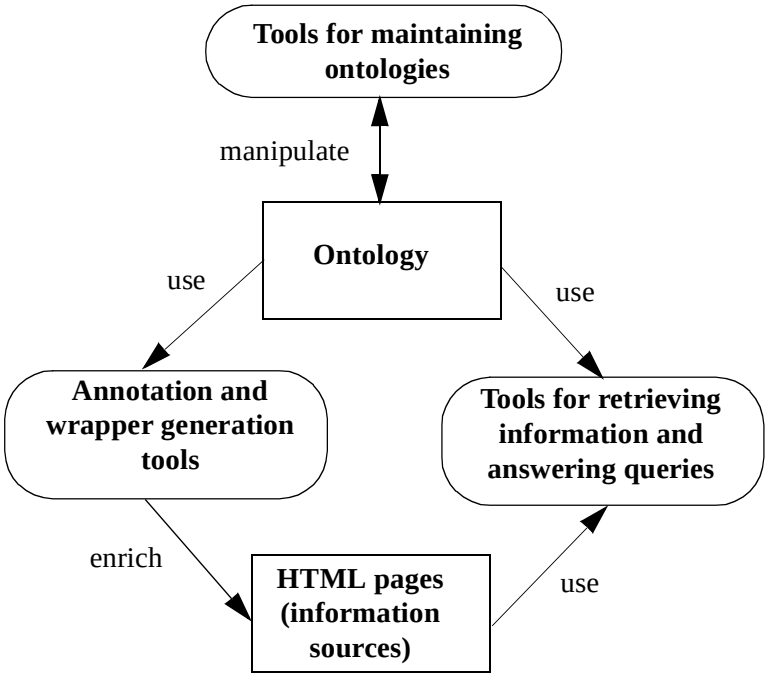


Fig. 10.5. Tools to support Ontology-based Knowledge Management

Maintenance

An approach is needed to maintain knowledge, both at the instance level where researchers annotate their personal pages, as well as at the ontology level. Changes to the ontology might have dramatic consequences for updating the annotations in HTML pages, especially in pages that are annotated with an ontology term that has become obsolete. Moreover, often there will not be a simple one-to-one correspondence between the old and the new ontology terms. In order to solve the issues involved with a changing ontology, one has to consider at least two aspects of the problem:

- The first, more technical, problem is how to deal with knowledge expressed in old ontology terms (in HTML pages), while Ontobroker uses an updated, newer ontology.
- The second is a social problem: how to organize the process to encourage people to update their Web pages using the new terminology? A first observation is that such a process should avoid radical changes and enforcements, which generally discourage people from following it.

For these issues, a process model is needed. The following process model is suggested:

1. If a term from the ontology is replaced by one or more other terms, it is marked with a 'depreciated' flag. That means that it does not show up anymore in the query interface and annotation tool. This way, only up-to-date terms can be used for querying and annotation.
2. However, many HTML-pages that use the old ontology terms may exist. In order to have the knowledge defined in those pages still available, mapping rules have to be defined. These mapping rules map the old terminology (which is used in the antecedent to the rules) to the new terminology (which is used consequent to the rule). An exact mapping may not always exist, but a partial translation is often useful and the loss of information can be determined [Mena et al., 1998].
3. After a while, knowledge providers who still use the old ontology terms can be warned, because the source of each fact is known.
4. As a last step, the mapping rules and the old ontology have to be deleted. Old knowledge of providers who have not updated their pages will not be available anymore. This is a loss, but apparently such providers do not maintain their pages and the knowledge would probably be outdated anyway.

Feasible time constraints, for instance how long the old ontology terms should be supported, is currently unknown. Their determination and the evaluation of this process model needs is beyond the scope of this thesis.

Scaling up

What happens when the knowledge is spread over tens of thousands of HTML pages? Apart from the updating problem (see above), the intelligent reasoning part might also become a problem. This is familiar in KBS research, when algorithms developed and tested on toy domains have to scaled up to real word applications. There is, however, a continuum in ontology building that can be exploited to enhance performance. This continuum relates to the number and type of rules defined in the ontology. If performance becomes a problem, the number and/or generality of the rules in the ontology can be reduced. There is a trade-off of *ease and elegance of modeling* against efficiency. In the extreme case all rules are dropped (which would reduce modeling comfort considerably!), and the approach boils down to a database lookup problem.

10.3.2 Social and Organizational Risks

Minimum amount of participants

Without participating researchers, the $(KA)^2$ initiative would have failed. However, the nature of the initiative is such that participation is rewarding; it is a self-promoting activity. That is, researchers are more highly visible if they participate because other researchers and outsiders can more easily find their work. In an organizational context appropriate incentives have to be defined so that providers are rewarded for their efforts.

Competitive mentality

In many companies, the mentality is competitive rather than collaborative. In other words: '*If my colleague wins, then I lose'*. And: '*If I make my knowledge available to others, then others will profit from that, and there will be a risk that they outperform me.*' This mentality is a real threat to the success of Knowledge Management initiatives. Increasingly more companies have become aware that a collaborative mentality leads to better results than competitive thinking [Covey, 1989]. Organizations can stimulate collaborative thinking by changing the incentive system (such as making it financially rewarding to share knowledge).

Incentive system

Given the high work load of today's employees, it may easily be felt that contributing to a Knowledge Management effort is a waste of time, or at least does not have priority. This kills any Knowledge Management initiative. Organizations should therefore reward Knowledge Management contributions equally as results that lead to direct profits. In addition, an effort should be made to reuse existing documents so that knowledge workers do not retain the impression that they have to duplicate knowledge.

10.4 Related Work

The *Math-Net project*(1) [Dalitz et al., 1997] sets up the technical and organizational infrastructure for efficient, inexpensive and user-driven information services for mathematics. Math-Net uses a comparable approach to $(KA)^2$: with the aid of active, structured retrieval mechanisms and passive (profile services) components, electronic information about mathematical resources in Germany is made available to the scientist at his or her workplace. The emphasis is put on information about publications, software and data collections, teaching and research activities, but also on organizational and bibliographical information. Decentralized organization structures, search systems as well as meta-information (metadata) in accordance with the Dublin Core, are used. However, Math-Net does not rely on shared, explicit ontologies. A minimal way to extend homepages is provided via a Web-based homepage annotator. The homepage annotator generates a fixed table structure, which is then interpreted by the Math-Net system, and decomposed into addresses, phone numbers, etc.

PhysNet(2) is a community portal for physics. PhysNet relies on manual work, visitors have the possibility to submit URLs. The metadata facilities of Math-Net are reused.

The *Social Science Information Gateway (SOSIG)*(3) is a freely available Internet service which aims to provide a trusted source of selected, high quality Internet information for students, academics, researchers and practitioners in the social sciences, business and law. It is part of the UK Resource Discovery Network. The technology was developed in the European DESIRE(4) [Belcher & Place, 2000] project. SOSIG uses a Web crawler to automatically create a database of Web resources. The technology takes the URLs from the manually created SOSIG catalogue as seeds in a process that uses robots to gather further resources from the Internet. This data is available for keyword searching but cannot be browsed unless a classification is entered manually by domain experts.

1. see http://www.math-net.de
2. http://ins.uni-oldenburg.de/PhysNet/
3. http://www.sosig.ac.uk/
4. http://www.desire.org

10.5 $(KA)^2$: Assessment

$(KA)^2$ was an initiative without funding. Therefore, it depended heavily on the enthusiasm of its participants and on indirect benefits like publications and publicity. At several meetings during conferences and workshops, $(KA)^2$ has attracted a broad, but floating and often arbitrarily composed group of people.

Furthermore, $(KA)^2$ has produced an ontology that was used to describe about 1800 facts on the Web. The ontology consists of two main parts: a general ontology useful for describing organizations, persons, publications, etc., and a specialized ontology useful for describing research topics of the knowledge acquisition community. The former part is straightforward and could be developed quickly, mainly by reusing existing data models and ontologies. The latter, more interesting, part needed a lot of discussion and was subject to changes with respect to topics of knowledge acquisition and, in addition, some related scientific areas. The overall organization of the topics and the structure of describing each topic has been established in several plenary workshop meetings, and is shown. This ontology has to be instantiated for different research topics covered by the knowledge acquisition community. The topics that were identified in a number of meetings are: *Reuse, Problem-Solving Methods, Ontologies, Validation And Verification, Specification Languages, Knowledge Acquisition Methodologies, Agent-Oriented Approaches, Knowledge Acquisition From Natural Language, Knowledge Management, Knowledge Acquisition through Machine Learning, Knowledge Acquisition through Conceptual Graphs, Foundations of Knowledge Acquisition, Evaluation of Knowledge Acquisition Techniques and Methodologies,* and *Knowledge Elicitation.* Each of these topics has been given to a small group of experts who completed the definition. This distributed approach worked well for some of the topics and did not produce any results for others. It required a significant additional effort of the core committee to fill in the gaps and to achieve some level of homogeneity in describing the different topics.

Already noted is the fact that Knowledge Management also involves people, and therefore any Knowledge Management effort is doomed to fail if human factors are not taken seriously. Knowledge Management only works if people cooperate and are willing to share their knowledge. One way to stimulate sharing of knowledge is to change the incentive system accordingly. Another important social aspect is that, in order to be successful with a large-scale initiative, lightweight tools are a must.

The current approach to Knowledge Management was presented through a large initiative for the knowledge-acquisition community concerned with ontology building on the Internet, knowledge annotation of Web pages and query answering. The results highlight both the technical and social issues when dealing with Knowledge Management. Solution directions for some technical problems (e.g. tool support) were also discussed. Social issues concern, above all, people's willingness to participate in a Knowledge Management initiative which, in a scientific community, seems more feasible than in a commercial organization.

Finally $(KA)^2$ opened the door for much more sophisticated emergent Community Web approaches like OntoWeb[1] and SemanticWeb.org[2] among others.

1. http://www.ontoweb.org
2. http://www.SemanticWeb.org

10.6 Conclusion

$(KA)^2$ was successful, despite the small community approach and limited resources available. The Ontobroker tools were used successfully within the $(KA)^2$ initiative. $(KA)^2$ has also shown the bottlenecks of the Ontobroker approach, which are the joint ontology development and maintenance as well as coping with an evolving ontology. Both topics are beyond the scope of this thesis, but are important areas for future research.

Chapter 11 Related Implementations

While work related to the single technologies used within Ontobroker has already been discussed, this chapter compares frameworks aiming at similar goals with the Ontobroker system.

11.1 SHOE

The *SHOE* (Simple HTML Ontology Extension) [Luke et al., 1997], [Heflin, 2001] approach is very close in spirit to the Ontobroker approach. Like Ontobroker, SHOE provides an extension of HTML for the annotation of HTML pages with ontology based metadata. In contrast to HTML-A and XAL, SHOE relies on annotations that are *separated* from the content, e.g., located at the beginning or at the end of a document, or in a separate file. The SHOE language provides primitives to enable annotators to annotate their Web pages with extensions of existing ontology as well as instance data. The SHOE primitives allow for the definition of:

- *ontologies*. Ontologies may be derived from other ontologies and may contain any number of classes, relations, constants and inference rules.
- *classes* (called *Categories* in SHOE). A predefined subclass relation allows for the definition of a class hierarchy.
- *relations* and the type of arguments.
- *renamings*, which help to relate classes in different ontologies to each other.
- *inference rules*, which are simple horn clauses (without negation). The atoms in the head may be either class definitions or relations. In the body of the horn clause, additionally, comparison operators are allowed.
- *constant definitions*. Often constant definitions are useful within ontologies, e.g., red as an instance of color.
- *data type definitions*. Besides the basic data types NUMBER, STRING, DATE and TRUTH, arbitrary data types may be defined. However, these data types just exist as names and no additional semantics is provided.
- *instance assertions*. Instance assertions don't belong to an ontology, but rather commit to one or more ontologies. Instances may possess relations (slots), type definitions and sub-instances.

The SHOE ontology language and Ontobroker's F-Logic variant possess a very similar expressivity. Major differences are that the ontology (schema definition) is not available on the meta-level, i.e., it is not possible to formulate a query that asks for all attributes of a given class, using the SHOE rule language. For achieving this capability, yet another language and query system is necessary within SHOE.

SHOE and Ontobroker rely on horn-logic as the rule language. Ontobroker has additional syntactic facilities, which simplify the modeling of larger ontologies and rules sets (i.e., full first-order rules). Since these extensions rely on a form of non-monotonic negation, these additional features cannot easily be incorporated into SHOE. SHOE explicitly renounces negation (cf. [Heflin, 2001], p. 69). This is an important distinction, since for the formulation of more sophisticated queries, posed with respect to a finite amount of data (e.g., all facts defined on a specific Web site) require the use of negation with a closed-world-semantics [Reiter, 1982]. Even reasoning on the Web requires some

kind of negation, although it has not yet been incorporated into rule languages for the Web. Suggestions include [Himmeröder et al., 1997] and [Abiteboul & Vianu, 2000], which propose *inflationary semantics* as a semantics for Web negation. SHOE allows anyone to extend an ontology, whereas Ontobroker relies on a central ontology which can only be extended by a central authority. Although the SHOE approach seems to better fit the spirit of the Web (‘*everybody can say everything about anything*’) it is as yet unclear how an ontology evolution process could benefit from enabling distributed ontology extensions. From an annotation perspective SHOE annotations are separated from the content of the annotated document, whereas the Ontobroker annotation language was explicitly designed to integrate document and metadata with the goal of avoiding redundancy.

SHOE also possesses a tool environment: a Java-based *Ontology API* is used to manipulate SHOE ontologies, the *Knowledge Annotator* helps to annotate HTML pages. The Knowledge Annotator corresponds roughly to Ontobroker’s OntoPad. However, OntoPad was developed as an HTML-editor extension, following a *mark & annotate* approach, whereas the Knowledge Annotator separates annotation and HTML source. *Running SHOE* is a SHOE wrapper generator, exploiting regular structures like tables in HTML pages. *EXPOSE* is SHOE’s Web-crawler, which crawls the Web searching for SHOE annotations. The main difference to Ontobroker is that Ontobroker relies on an index system, in which knowledge providers submit their pages to the index in order to get crawled. This decision was taken since the total number of annotators is small compared to the total size of the Web. Thus attempts to crawl the Web in search for Ontobroker metadata did not seem very promising. For query answering and inference rule evaluation XSB and PARKA [Evett et al., 1993] are used. PARKA is an extremely efficient Knowledge Representation system, but inferences are restricted to the *is-a* relationship. *SHOE SEARCH* roughly corresponds to the Ontobroker query interfaces. The ontology is presented in a text window and indentation indicates the subclass relationship. Then a user is able to select available classes and relationships to construct a conjunctive query. An interesting feature is the ability to combine queries to the knowledge base and queries to popular search engines.

11.2 WebKB

This version of WebKB [Martin & Eklund, 1999] permits Web users to store, organize and retrieve knowledge or document elements (DEs) from Web-accessible files. Ontologies as well as instance data may be asserted via Conceptual Graphs (CGs) [Sowa, 1984]. Thus, access to ontologies and instance data is possible within the same formalisms, comparable to Ontobroker, but unlike SHOE.

Several tools are available for editing of CGs, e.g., a Conceptual Graph Textual Editor, the *Conceptual Graph Graphic Editor* (WebKB-GE) and a *Hierarchy Browser*. However, a close integration of metadata annotation and HTML editing, as in Ontobroker’s OntoPad, has not been achieved.

A Knowledge-based Information Retrieval/Handling Tool enables the answering of queries based on a virtual knowledge base. The tool supports the construction of new HTML documents from query answers. However, querying is done via a set of Unix-like text processing commands for exploiting Web accessible documents, e.g., *cat, grep, diff, head, tail*, etc. These tools help to write agents, which exploit the available knowledge, but don’t provide an adequate user interface for a novice user. Furthermore, it is unclear how a user would navigate in large ontologies.

The inference capabilities of WebKB are restricted to the subsumption relationship between classes. Rules which, as in Ontobroker and SHOE, cannot be formulated.

The next generation of WebKB, WebKB-2 [Martin & Eklund, 2001] will enable Web users to jointly store, organize and retrieve knowledge in a large single knowledge base on a WebKB server machine.

11.3 Other Approaches

Approaches aiming to query information on the Web are summarized in [Florescu et al., 1998]. First generation Web Query Languages like *WebSQL* [Mendelzon et al., 1997], *W3QL* [Konopnicki & Shmueli, 1995] und *WebLog* [Lakshmanan et al., 1996] combined information retrieval techniques with formal queries on hyperlinked HTML documents, using different techniques like SQL-style queries, proprietary functions or datalog-like rules.

Second generation languages (called *Web Data Manipulation Languages* [Florescu et al., 1998]) provided query primitives to query the semi-structured data model. Examples include *Lorel* [Abiteboul et al., 1997a], *WebOQL* [Arocena & Mendelzon, 1998] and *StruQL* [Fernández et al., 1998].

All these approaches do not use an ontology to query and integrate information, which is central in the current approach.

Another approach sharing some commonalities with Ontobroker is *FLORID* [May, 2000]. FLORID combines exploration, wrapping mediation and restructuring information of the Web into one coherent system. Deploying a pattern matching approach, FLORID rules are able to extract data from HTML pages and provide them for reasoning and querying within an F-Logic reasoning engine. The conceptual models, while necessary and present, are built adhoc and are not intended for sharing.

However, the strength of FLORID in providing an integrated platform is also a weakness: it is difficult to apply advanced wrapper-learning techniques within FLORID. Although FLORID is also based on F-Logic, it does not provide features such as expressible first-order rule bodies, which are helpful for the task of modeling ontologies.

PART III METHODS AND TOOLS FOR THE SEMANTIC WEB

"But what ... is it good for?"
Engineer at the Advanced Computing Systems Division of IBM, 1968, commenting on the microchip.

Chapter 12 An Information Foodchain for the Semantic Web

12.1 The Semantic Web: An Introduction

The World Wide Web is based on a set of established standards, which guarantee interoperability at various levels: for instance, the TCP/IP protocol has ensured that nobody has to worry about transporting bits over the wire anymore. Similarly, HTTP and HTML have provided a standard way of retrieving and presenting hyperlinked text documents. Applications have been able to use and build upon the common infrastructure and this has led to the WWW.

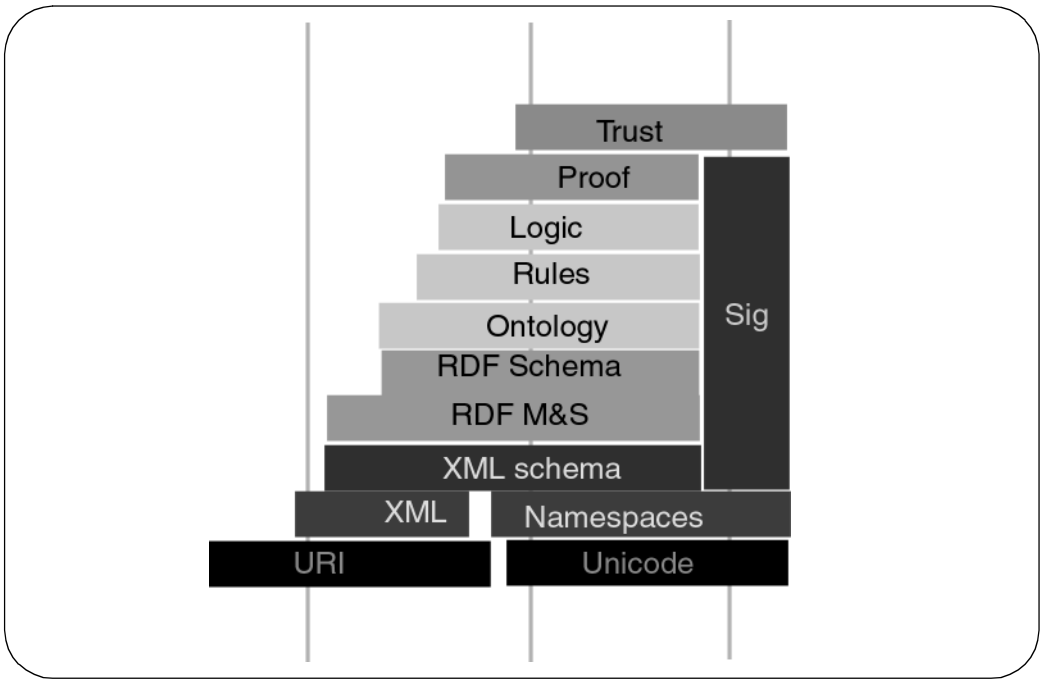


Fig. 12.1. The Technology Stack for the Semantic Web
(from http://www.w3.org/2001/Talks/0228-tbl/slide5-0.html)

The current Web can be characterized as the second Web generation: the first generation Web started with handwritten HTML pages; the second generation made the step to machine generated and often active HTML pages. These generations of the Web were meant for direct human processing (reading, browsing, form-filling, etc). The next, third generation Web, aims at machine-processable information. This coincides with the vision that Tim Berners-Lee calls the *Semantic Web* in his recent book "Weaving the Web" [Berners-Lee, 1999]. The Semantic Web enables automated intelligent services such as information brokers, search agents, information filters etc.

The Semantic Web, consisting of machine processable information, will be enabled by further levels of interoperability. Technology and standards need to be defined not only for the syntactic representation of documents (like HTML), but also for the their semantic content. Semantic interoperability is facilitated by recent W3C standardization efforts, notably XML/XML Schema and RDF/RDF Schema. The technology stack envisioned by the W3C is depicted in Figure 12.1.

Higher level technologies, like ontologies or rules, will be built upon technologies like XML, URIs and namespaces. However, the vision (and Figure 12.1) is generic: the task is to fill in the vision, taking Ontobroker as a starting point, with concrete content.

12.2 An Information Foodchain for the Semantic Web[1]

Ontobroker was an attempt to provide machine-readable information by exploiting shared semantics on the Web. However, Ontobroker had limitations: while Ontobroker dealt with information delivery, the technologies needed to create and deploy machine-interpretable data on the Web go beyond what Ontobroker is able to do. One desired technology is information processing agents that provide an extension to human capabilities (cf. [Genesereth & Ketchpel, 1994]). To facilitate automated agents on the Web, agent-interpretable formal data, metadata and services are required. Creating and deploying ontology-based metadata about a particular domain is a high-effort task, and it is not immediately clear what kind of techniques, data and tools are required to support this task. In this section an information food chain [Etzioni, 1997] is presented for advanced applications on the WWW that clarifies what kind of technologies and tools are needed. Each part of the food chain provides information that enables the existence of the next part.

For data exchange on the Web it is necessary to have a specification of the terminology of a particular domain. Raw data available on the Web without any knowledge or explanation about its semantics is useless. Ontologies [Fridman Noy & Hafner, 1997] (as introduced in Section 2.2.2) were used in part II of this thesis as a means to facilitate knowledge management among human users. They also may play a role in software agents. However, ontologies have only found their way into agent research recently:

> *"The ontology issue has always been considered secondary to other issues such as cooperation, negotiation, formalisation and logics for beliefs, desires and intentions, etc. You will find few papers and research projects truly addressing this issue. Perhaps, this is a case of sheer escapism. Whither the ontology problem? Not a chance! This problem is at the core of the agent inter-operability issue - is it realistic to expect knowledge and co-operation level interoperability without a significant degree of ontological sophistication of the agents concerned? We are now convinced that FIPA ACL and all other such attempts are, at worst, bound to fail or, at best, only enjoy limited and short-lived success due to our skirting round the ontology issue."*[Nwana & Ndumu, 1999]

If ontologies are also an artifact necessary for facilitation of machine-to-machine communication, it is desirable to develop machine-understandable, easily accessible representation formats for ontologies. XML can be regarded as the syntactic representation layer for other languages.[2] Domain-specific DTDs (as used at Biztalk[3]), can be regarded as a very limited kind of ontology. However, DTDs describe only the grammar (the structure) of a document and not the content. More expressive ontology languages have been defined, e.g., RDF Schema [Brickley & Guha, 2000], XOL (XML-based Ontology Exchange Language) [Karp et al., 1999], OML/CKML (Ontology Markup Language/Conceptual Markup Language) [Kent, 1999], and DAML+OIL [DAML Joint

1. This section is an extended and revised version of [Decker et al., 2000a].

2. XML is accepted because the minor restrictions that XML imposes are outweighted by the advantages of reusing XML processing software.

3. http://www.biztalk.com

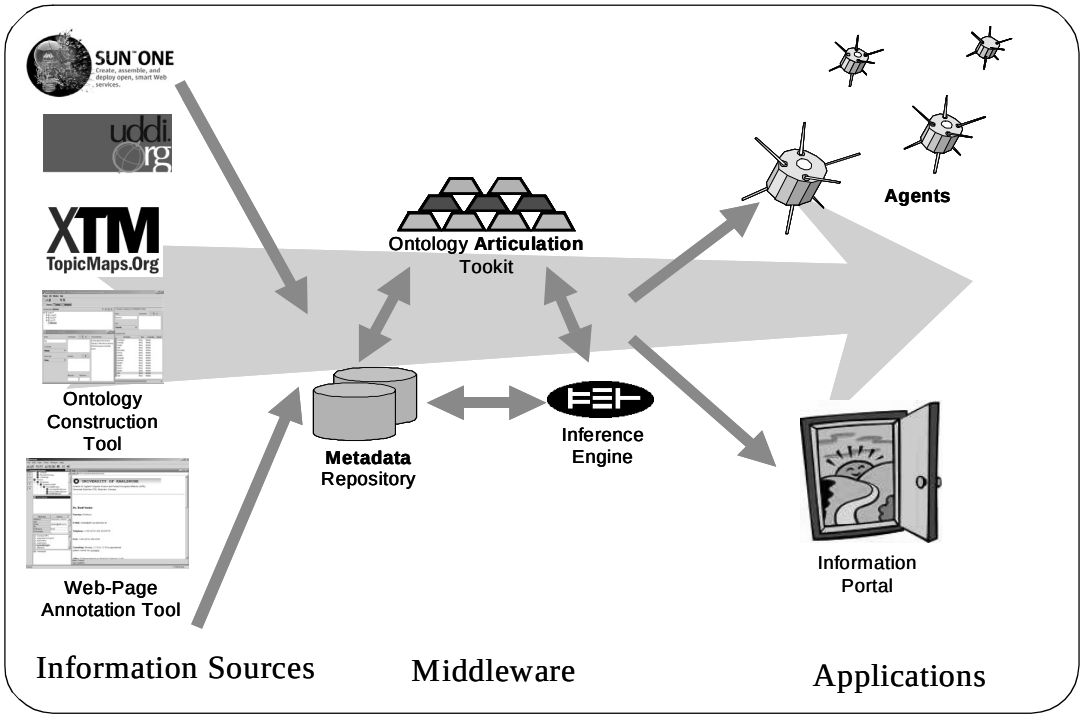


Fig. 12.2. An Information Foodchain for the Semantic Web

Committee, 2001]. An XML-based representation language and formal ontologies are a foundation for automated agents on the web. However, to create and deploy machine-interpretable data, we need manageable infrastructures (e.g., support and deployment tools). Thus, further elements in the information food chain are required. A pictorial description of the needed information food chain is shown in Figure 12.2.

Since an ontology is the foundation for a set of data items, the food chain starts with the construction of an ontology - preferably using a support tool, an Ontology Construction Tool. Though most information sources have a underlying structured repository of data, often in databases, most of the available data on the web now consists of unstructured HTML pages [Lawrence & Giles, 1999]. Thus, a major task is to make this information and structure available.

The next part of the information food chain describes a tool to support this task. A Web page Annotation Tool provides the means to browse an ontology and to select appropriate terms of the ontology and map them to sections of a Web page. The Web page annotation process creates a set of annotated Web pages, which are available to an automated agent to achieve its tasks. Of course, the annotation process itself has a human component: although the effort to generate the annotation of a Web page is of an order lower in magnitude than the creation of the Web page itself, there has to be some incentive to spend the extra effort. The incentive for the creation of the annotation (which is metadata for the Web page) is visibility on the web for a Community Web Portal, which presents a community of interest distributed on the Web to the outside world in a concise manner. The data collected from the annotated Web pages simplifies the task of maintaining a Community Web Portal drastically because changes are incorporated automatically, without any manual work. An automated agent itself needs several sub-components: an important task of the agent is the integration of data from several distributed information sources. Because of the need to describe the relationships between the data in a declarative way (otherwise a new agent has to be programmed

for every new task), an agent needs an inference engine for the evaluation of rules and queries. The inference engine is coupled with a metadata repository - the memory of an agent as to where retrieved information is cached. Furthermore, if an automated agent browses the web it will usually encounter data formulated in unknown ontologies. Therefore, it needs a facility to relate unknown ontologies to ontologies it already knows. This facility is an Ontology Articulation Toolkit for information mediation.

12.2.1 Ontology Editors

Ontologies are engineering artifacts, and constructing an ontology involves human interaction. Therefore, an Ontology Construction tool is necessary to provide the means to construct ontologies in a cost-effective manner. Because ontologies evolve and change over time (as our knowledge, needs, and capabilities change), reducing acquisition and maintenance cost of ontologies is an important task. (see Section 2.2.2 for a discussion of available ontology editors). A new requirement for ontology editors for the Semantic Web is the creation of a joint standardized format - otherwise automated agents will most likely not be able to understand ontologies created by a particular editor.

12.2.2 Webpage Annotation Tool

Creating machine-interpretable data on the Web is a costly process. Hence, an important goal of an information food chain is to reduce the cost of ontology creation as much as possible. The annotation process requires tools that support users in the creation of metadata. The creation of explicit ontology-based data and metadata for HTML-pages requires much effort if done naively. Most HTML pages found on the Web are created using a visual HTML editor, so that annotators often already know how to use these tools. However, HTML-Editors focus on the presentation of information, whereas we need support for the creation of ontology-based metadata to describe the content of information. Thus, the information food chain needs a practical, easy-to-use, ontology-based, semantic annotation tool for HTML pages.

A basic tool, OntoPad (see Section 8.3), was presented in Section 8.3. (cf. [Handschuh et al 2001] for an improved version of OntoPad). However, for significant annotation tasks a basic annotation tool is not sufficient. It still takes a long time to annotate large pages, although a significant improvement was reported when compared to the manual task. Semi-automated annotation seems to be a promising approach to enhance tools like OntoPad. Such a practical tool also exploits information extraction techniques for semi-automatic metadata creation. The precision of linguistic processing technology is far from perfect and reasonably exact automatic annotation is not possible. However, much linguistic processing technology currently exists that is highly appropriate to help users with their annotation tasks [Alembic, 1999], [Day et al., 1997], [Neumann et al., 1997]. Often it is sufficient to give the user a choice to annotate a phrase with one of two or three ontological expressions. Resources like WordNet [Fellbaum, 1998] and results obtained from the Scalable Knowledge Composition (SKC) project [Mitra et al., 2001] provide high-level background knowledge to guide the annotation support.

12.2.3 Ontology-Articulation Toolkit

In order to solve a task that involves consulting multiple information sources with unknown ontologies, an automated agent needs to bridge the semantic gap between the known and the new or unknown ontologies. The Scalable Knowledge Composition Project (SKC) [Mitra et al., 2001] has developed tools that support an expert with the process of defining rules that link different

ontologies, i.e., ontologies from different sources that vary in their terminology and semantic relationships. These rules define a new articulation ontology, to serve the application, and translate terms that are involved in the intersection of those in the source domain. Typical source domains with an intersection articulation are trucking and air-cargo freight logistics. Information produced within the source domain in response to a task can then be reported up to the application using the inverse process. Now the results are translated to a consistent terminology, so that the application can produce an integrated result.

Within the SKC approach not all of the terms in the source ontologies have to be aligned, i.e., be made globally consistent. Aligning just two ontologies completely requires a major effort for a practical application, as well as ongoing maintenance. For instance, it is unlikely that United Airlines and British Airways will want to agree on a fully common ontology, but a logistics expert may be able to easily link the terms that have matching semantics, or define rules that resolve partial match problems. Partial matches occur when terms differ in scope e.g., one airline includes unaccompanied luggage in its definition of cargo and another does not. Partial match is dealt with by having rules in the articulation that indicate which portions of two partially matching useful concepts belonging to different ontologies are semantically similar and can thus be combined. Articulation also provides scalability. Since there are hundreds of domains just in the logistics area, and different applications need to use them in various combinations, the global-consistency approach would require that all domains that can possibly interact at some time must be made consistent. No single application can take that responsibility, and it is unlikely even that any government can mandate national semantic consistency.

However, it is very likely that automated agents on the Web will come across new domains (and domain ontologies) that the agent has no prior knowlege about. It is desirable for an agent to be capable of exploiting the available information sources and services. Exploiting unknown data sources is a difficult problem since the agent does not possess any prior information about the semantics of the terms and the relationships in the new information source. To address this problem methods for the generation of dynamic articulations are necessary, on an as-needed basis. This will allow for the creation of working articulation ontologies or linkages among the source ontologies. If the end-user has the expertise to perform the articulation, then no specialist expert who understands the linkages among the domains is needed. For instance, any frequent traveler today can deal with most domain differences among airlines.

12.2.4 Agents Inference System

Combining information from different sources enables new applications and exploits available information as much as possible. Combining information requires inference techniques to enable the ontology language to relate different pieces of information to each other. An agent inference system provides the means for an effective and efficient reasoning for agents. We expect that this technology will enable agents to reason with distributed available knowledge. Metadata on the Web is usually distributed and added value can be generated by combining metadata from several metadata offerings: for example, in the transport domain one metadata offerer might state that a flight is available from Washingon D.C. to New York. Another statement from another server states the availability of a flight from New York to Frankfurt. Combining these two pieces of information, an agent inference system can deduce that a connection is available from Washington DC to Frankfurt, although this it not explicitly stated anywhere. Even such simple rules as used in Figure 12.3 are neither evaluable by commercial databases (because of their inability to perform recursion) nor by simple PROLOG systems (because of potential infinite looping). Thus, neither type of

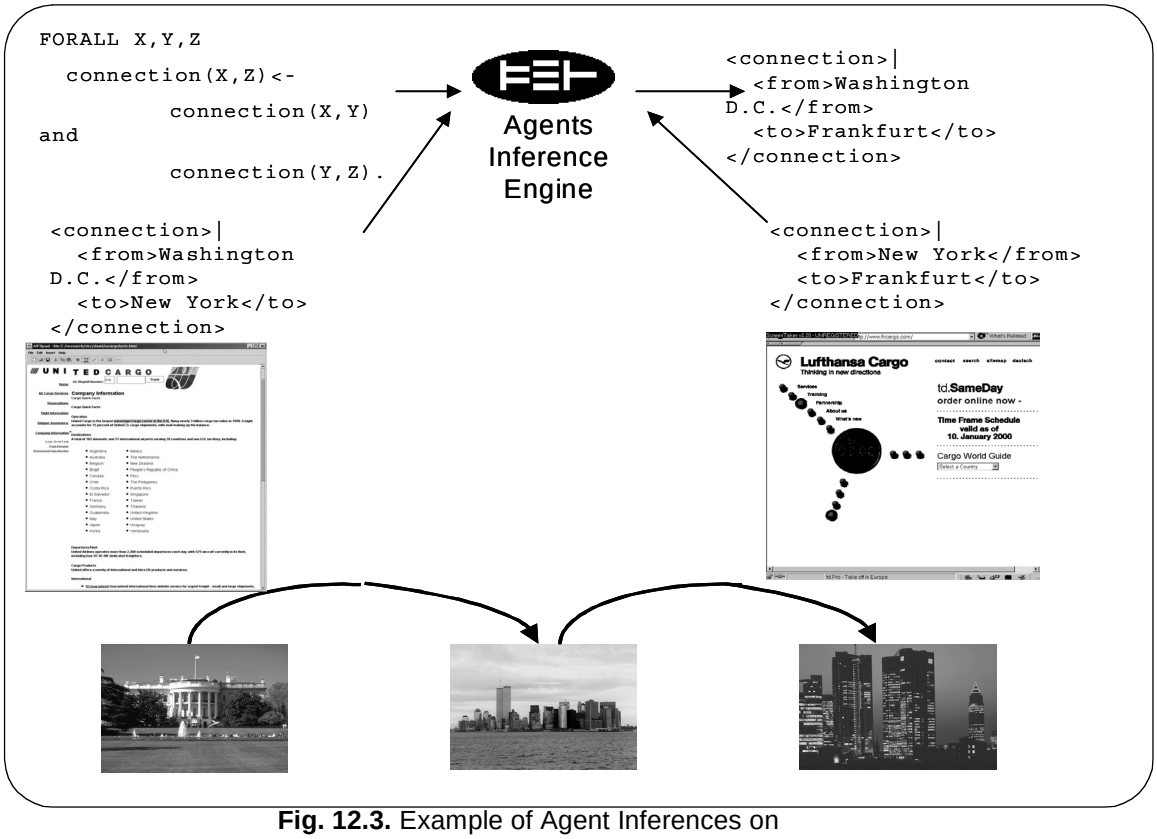


Fig. 12.3. Example of Agent Inferences on the Web

system is usable for this task. Inference engines also reduce metadata creation and maintenance cost. Having to state every single assertion of metadata explicitly leads to a large metadata creation and maintenance overheads. An inference engine helps reduce the amount of explictly stated metadata by infering further implicit metadata [Mitra et al., 2001]. Implicit metadata can be derived from already known metadata by using general background knowledge, as stated in ontologies. The properties of these reasoning capabilities have to be carefully chosen. A reasoning mechanism which is too powerful has intractable computational properties, whereas a too-limited approach does not enable the full potential of inference. Deductive database techniques have proven to be a good compromise between these tradeoffs. The proposed usage of the inference engine is slightly different from the usage of SiLRI within Ontobroker (see Chapter 7). In Ontobroker the inference engine was used to evaluate axioms of the ontology. Within the Semantic Web scenario the focus is on integration of rules for distribution information.

12.2.5 Declarative Web Site and Community Portal Design

A Semantic Community Portal Web site is an application demonstrating the use of ontology-based markup. From the very beginning of the Web, communities of interest have created Web sites that covered topics they deemed to be of interest to their group. These users create what is now commonly known as community Web portals [Staab et al., 2000]. Community Web portals are hierarchicaly structured, similar to Yahoo!, and require a collaborative process with limited resources (manpower, money) for maintaining and editing the portal. Since the presented information is normally created in a distributed manner, updating and maintaining such a site is a major effort. Technology for supporting communities of interest has not adequately kept up with the complexity of the tasks of managing community Web portals. Ontologies provide an underlying

structure [Fröhlich et al., 1998] for a Web site. This structure and Web pages annotated with ontology-based metadata is used to generate the Automated Community Portal automatically. This significantly reduces the effort to maintain a Community Portal.

12.2.6 Advanced Agent Applications

Intelligent autonomous agents are a possibile vision for providing information processing support on the Internet. However, current technology does not enable agents to arbitrarily understand Web pages. Instead *Wrappers* (cf. [Sahuguet & Azavant, 1999]) have to be built that parse particular Web pages and assign a predefined meaning to a page. However, wrappers are costly to build and to maintain - a single change in the output format of an HTML-page makes the wrapper unusable. But even if the generation of wrappers were feasible on a large scale basis wrappers have to assign meaning to parts of a data source, which makes a shared vocabulary between a wrapper and the wrapper-using application necessary. With the support provided by the lower levels of the proposed information foodchain (metadata and ontologies) these problems disappear and agent browsing the Web and processing the information become feasable. An example of multiple agent application occurs in travel scheduling, where combining the services provided by several distributed service providers occurs in an ad hoc way.

12.3 Related Work

In the Ontobroker [Decker et al., 1999] and SHOE project [Heflin, 2001] means were investigated to annotate Web pages with ontology-based metadata, thus realizing part of the food chain. Agent based architectures usually focus on inter-agent communication instead of ontology creation and deployment.

12.4 Conclusion

An information food chain was defined that delivers an infrastructure to empower intelligent agents on the Web and deploys applications that will facilitate automation of information processing on the Web. Fundamental to that approach is the use of a formal markup language for annotation of Web resources. This information infrastructure is expected to be the basis for the *Semantic Web* idea - that the World Wide Web (WWW) will become more than a collection of linked HTML-pages for human perusal, but will be the source of formal knowledge that can be exploited by automated agents. Without automation, and precision of operations, business and governmental uses of the information will remain limited.

Chapter 13 RDF and Semi-Structured Data(1)

13.1 Introduction

Nowadays, services on the Web are single islands. Combining several services usually requires, at the very least, time consuming copy-paste actions. Mechanisms for service and information integration are required, allowing the fast enabling of inter-service communication. Integration of several services requires standards for data interchange and the means to deal with different semantics of the data.

Similar topics arise with dynamic Business-to-Business (B2B) communication. Globalization and increased competition have raised the demand for streamlining business processes and for flexible (B2B) communication. In a fast-changing economy the facility to substitute vendors and suppliers with little effort is being demanded. While the Internet has enabled fast electronic (B2B) communication in principle, the problem of understanding the transmitted data still remains - different vocabularies and data formats complicate the data transfer task. Traditional data exchange means such as EDIFACT are hard to set-up and to change, semantic differences are hard to integrate.

To summarize, a data model and data exchange standards are required that enable the fast integration of different data-sources and allow users to bridge semantic differences.

Semi-Structured Data has been proposed as a means for data integration and mediation. The Web community has proposed the *Resource Description Framework* (RDF) [Lassila & Swick, 1999] as a data model suitable for information integration tasks.

In AI frame-based knowledge representation system has been suggested for the flexible representation of heterogenous knowledge. The following section examines these three representation schemes in more detail.

13.2 Knowledge and Information Representation for the Web

The representation of Knowledge on the Web (in the sense of machine-understandable data) poses new requirements for Knowledge Representation Languages: the Web introduces characteristics, like the inherent distribution, diversity and heterogeneity that need to be adressed. In the following properties of a data model for the Web are identified that may serve as a foundation for knowlege representation tasks.

1. *Data on the Web has to be uniquely identifiable.* Giving the same ID to different data items leads to confusion and conflicts. Fortunately, there is a well-known mechanism for uniquely identifying resources on the Web. Uniform Resource Identifiers (URIs) are used to locate resources. However, within RDF URIs just act as unique identifiers and not necessarily as resource locators. An actual URI is not required to be able to retrieve a resource. For example,

1. The section is an extended and revised version of [Decker et al., 2000b].

an URI, like *http://www.decker.cx/person/stefan*, could be used as an ID representing a person, but users are not able to download a specific document from that location. While URIs do not guarantee that ID will not happen, the possibility is greatly reduced.

2. *Data on the Web is heterogenous and often unstructured*. Because of the diversity of communities using the Web and different requirements from each of those communities, there will be no universally usable domain-specific knowledge and representation language. Instead, a knowledge representation language has to provide a basis for more specialized languages, such that each communitiy can build what it needs. On the other hand, the benefit of the Web and the Internet is the ease of communication. Although different communities have different requirements for the representation of information, they want to communicate with each other. Thus the data model has to support information integration and articulation.
3. *Data representation must be simple and extensible*. The data model serves as a foundation for other languages for all communities that exist on the Web. Therefore, the language needs to be simple and acceptable for a variety of user communities, and extensible in the sense that the user communities are able to adapt the language to their needs without compromising the entire approach.
4. *Data on the Web is distributed*. Data on the Web might be distributed on several servers so it should be possible to link to other sources and other data resources, and to create a web of machine-processable data. This closely resembles the idea of hyperlinks in HTML pages.
5. *Data on the Web is biased*. There is no such thing as universal truth. Anybody can say anything on the Web. The freedom to express any idea is already criminally exploited, for example with forged stock market news.[(1)] So an important requirement for representing data is the ability to say something about foreign data (especially about data residing on other servers), whether it is believable, supported, wrong, biased, etc. The requirement for a representation language is that the language needs facilities for expressing metadata about metadata.
6. *Support for Syntactic Interoperability.* By syntactic interoperability means the ease with which data can be read and a representation obtained that can be exploited by applications. For example, software components like parsers or query APIs should be as reusable as possible among different applications. Syntactic interoperability is high when the parsers and APIs needed to manipulate the data are readily available.
7. *Support for Semantic Interoperability*: Semantic interoperability means the difficulty of understanding foreign data. Semantic Interoperability is different from syntactic interoperability: syntactic interoperability talks about parsing the data, while semantic interoperability means to define mappings between unknown terms and known terms in the data. This requirement is one of the most important issues, since the cost of establishing semantic interoperation is usually higher than the cost of establishing syntactic interoperability, due to the need for content analysis.
8. *Universal expressive power.* It should be possible to encode everything in the data format. Since it is not possible to anticipate all potential uses, a data format must have enough expressive power to express any form of data. Clearly this requirement limits the level of support that is possible: the more domain specific a language is, the more it is possible to support the language

1. cf. http://news.cnet.com/news/0-1005-200-2126256.html

with tools and technologies. The challenge for the Semantic Web is to find a data format that does not restrict anybody (thus is by no means domain specific), and also allows the creation of an effective and efficient infrastructure that users as well as software developers can build upon.

The heterogeneity of information on the Web cannot be captured by a fixed schema, since the schema would impose limitations on the expessibility, which is not acceptable on the Web, where the knowledge representation tasks cannot be foreseen. Instead, the information model has to be self-describing, capable of representing arbitrary information, but structured enough to support an effective query processing. In the following we discuss RDF and semi-structured data in more detail.

13.3 RDF

13.3.1 The RDF Data Model

RDF consists of two parts - the RDF data model specification and a suggestion for an XML-based serialization syntax. The data model definition is the core of the specification - the syntax is necessary to transport RDF data in a network, but unimportant otherwise. The advantage of handling the RDF data model in this way is the independence from any particular syntax. A non-RDF data source can be made available to an RDF-aware system just by providing an RDF adapter that extracts semi-structured data from the data source.

This section provides an introduction to the data model, and discusses the serialization syntax briefly.

Semi-structured data consists of entities, represented by their unique identifiers, and binary relationships between those entities. In RDF, two entities and a binary relationship between these entities is called a statement. Represented graphically, the source of the relationship is called the subject of the statement, the labeled arc itself is the predicate (also called property) of that statement, and the destination of the relationship is called the object of that statement. The data model of RDF distinguishes between resources, which have an URI-identifier, and literals, which are just strings. The subject and the predicate of a statement are always resources, while the object can be a resource or a literal. In RDF diagrams, resources are always drawn as ovals, and literals are drawn as boxes. An example of a statement is given in Figure 13.1: resource *http://www.daml.org/projects/#11* is a subject and has a property *http://www.SemanticWeb.org/schema-daml-01/#hasHomepage* and the value of the property (the object) is http://www-db.stanford.edu/OntoAgents.

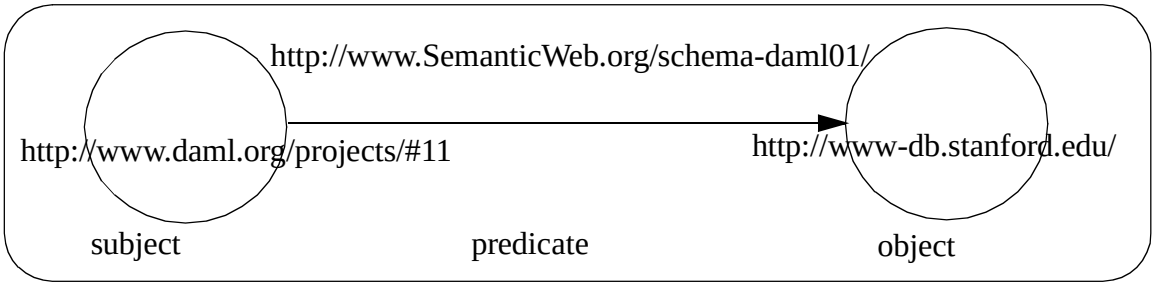


Fig. 13.1. The simplest possible RDF graph: two nodes and an arc

The statement can be read as: The resource *http://www.daml.org/projects/#11* has a homepage, which is the resource *http://www.SemanticWeb.org/schema-daml-01/#hasHomepage*.

This sentence can be broken down into the following parts:

Subject (Resource)	http://www.daml.org/projects/#11
Predicate (Resource)	http://www.SemanticWeb.org/schema-daml01/#hasHomepage
Object (Resource)	http://www-db.stanford.edu/OntoAgents

At first glance it might look strange that predicates are also resources and thus have a URI as a label. However, to avoid confusion it is necessary to give the predicate a unique identifier. Simply *hasHomepage* would not be sufficient, because different vocabulary providers might define different versions of the predicate *hasHomepage* with possibly different meanings.

A set of statements forms a graph. Figure 13.2 shows an extension of Figure 13.1: property *http://purl.org/DC/#Creator* with value *Stefan Decker* (a literal) has been added to the graph..

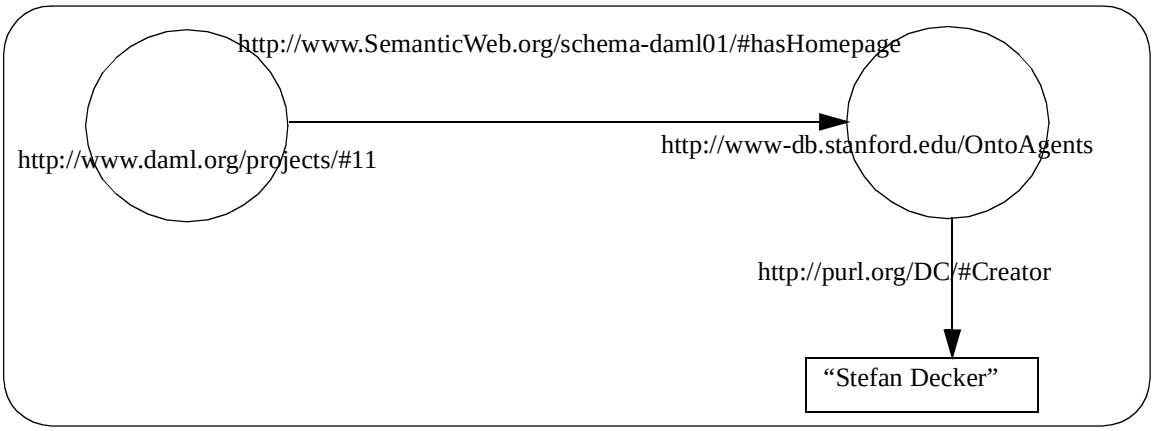


Fig. 13.2. An extension of Figure 13.1

This is the ‘core’ of RDF. To allow a more convenient data representation, additional vocabulary and conventions were introduced.

In the following the URIs are abbreviated by using the XML-namespace syntax. So instead of writing *http://www.SemanticWeb.org/schema-daml01/#hasHomepage* the namespace -form *sw:hasHomepage* is used with the assumption that the substitution of the namespace-prefix ‘sw’ with *http://www.SemanticWeb.org/schema-daml01/#* is defined somewhere. In the following the the namespace prefixes *rdf* for explaining vocabulary defined in the *RDF Model & Syntax Specification* document [Lassila & Swick, 1999] is used. The *rdf* prefix is expanded to *http://www.w3.org/1999/02/22-rdf-syntax-ns#*, which is the namespace defined for the RDF specific vocabulary.

The RDF namespace contains a property (predicate) definition *rdf:type*. *rdf:type* is used to denote a type relationship between two objects, which is similar to a *instanceOf* relationship in frame-based knowledge representation languages. *rdf:type* is useful in conjunction with schema languages for RDF, which also impose a strictly defined semantics for *rdf:type*. Figure 13.3 shows a typing statement. Vocabulary to describe the container is now introduced.

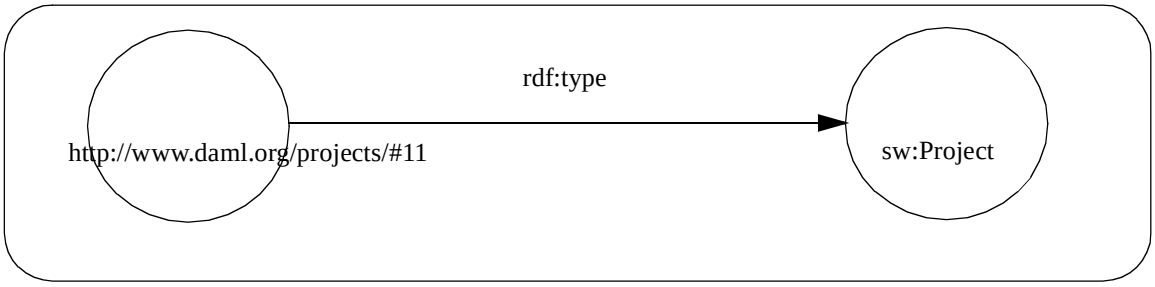


Fig. 13.3. Typing in RDF

Often there is a need to use a collection of resources. For example, it might be desirable to make a statement about all members of a certain project. A single statement about each element of the collection can be made. However, if the wish is to make a statement about all members of the project (which might change) and not about each member individually, then a container needs to be used to represent all members of the project.

RDF containers can be used to represent collections of resources or literals. RDF defines three types of containers:

- *Bags*. A bag is an unordered list of resources or literals. For example, a bag can be used to represent a class of students. The order of the students in this list is not important. Duplicate values are allowed in bags.
- *Sequences*. A sequence is an ordered list of resources or literals. a sequence can be used to represent a batch of jobs submitted to a processor. The order of these jobs is important and is preserved by using a sequence. Duplicate values are also permitted in sequences.
- *Alternatives*. Alternatives are a list of resources or literals out of which the property can take only a single value.

Fot the given task of representing possible travel scenarios for a journey from San Francisco to San Diego, one can either drive or fly the route. The different options can be expressed as a list of alternatives and the application can choose the suitable alternative.

Figure 13.4 presents an example of an *rdf:Bag*. The Bag contains all the project members of a project. The intermediate node representing the bag is the value of the *sw:hasMembers*-property. An *rdf:type* arc types it as an *rdf:Bag*. From the intermediate node ordinal arcs (*rdf_1, rdf_2*, etc.) point to the project members - in this case, literals representing their names.

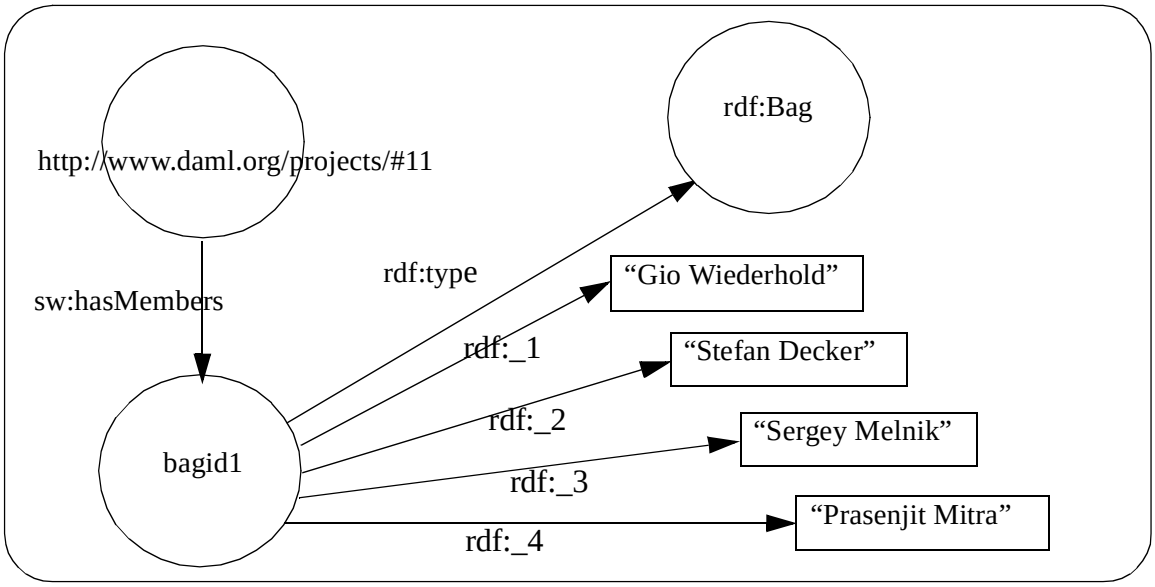


Fig. 13.4. Eample of an RDF Bag

As already discussed, since there is no universal truth on the Web, it is necessary to allow statements about statements. In order to make such statements about statements, the statement about which a statement is being made must be mapped to a single resource. RDF specifies how statements can be modeled as resources. This process is formally called reification and the statement that has been modeled as a resource is called a reified statement. E.g., the statement: *http://www.daml.org/projects/#11 is an NSF-funded project* is not a true statement, since DAML is funded by DARPA. So one might want to represent the statement "*The members of the project oppose the statement that http://www.daml.org/projects/#11 is a NSF-funded project.*" In order to oppose the original statement the statement is reified by introducing a new resource (node in the graph), which represents the original statement. The new resource representing a statement is typed with the resource *rdf:Statement*. It remains to model the parts of the statement: the subject, object and predicate. The RDF specification defines special properties: *rdf:subject*, *rdf:predicate*, and *rdf:object*, whose respective values are the subject, predicate and object of the original statement.

In the above mentioned example (depicted in Figure 13.5), the *rdf:subject* property of the reified statement is the URI of the OntoAgents project, the value of the property *rdf:object* is the object of the original statement - the identifier of the NSF, and value of the *rdf:predicate* property is the (resource representing the) predicate in the original statement - *sw:fundedBy*. The only thing missing

now is the statement about the reified statement: that all the members of the OntoAgents project oppose the original statement. To model this, an additional arc is introduced, labeled by *sw:opposedBy*, which ends at the *rdf:Bag* resource defined in Figure 13.4.

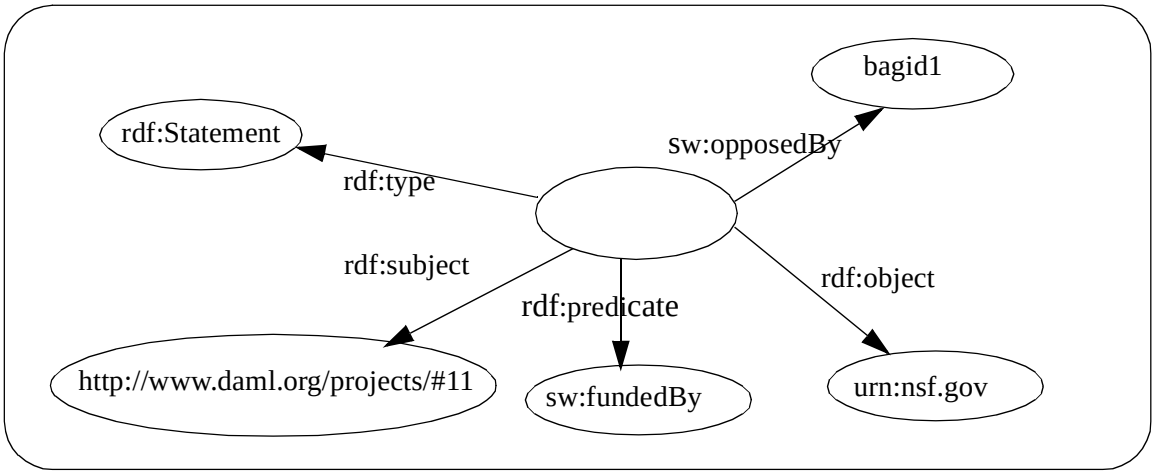


Fig. 13.5. Eample for a reified statement

13.3.2 RDF Schema

The basic data model for RDF has now been introduced. For practical purposes it is necessary to define schema information, since a common vocabulary needs to be agreed. This is the purpose of the RDF-Schema (RDFS) specification [Brickley & Guha, 2000]. RDF-Schema is an RDF-application - it is defined in RDF itself. The defined vocabulary is very similar to the usual modeling primitives available in Frame-based languages. In this section the vocabulary used in the former mentioned examples is defined using RDF Schema. The namespace-prefix *rdfs* is used as an abbreviation for *http://www.w3.org/2000/01/rdf-schema#*, the RDFS-namespace identifier.

Figure 13.6 depicts an RDF-Schema, defining the class *sw:Project* and two properties *sw:hasHomepage* and *sw:hasMembers*. The class node is defined by typing the node with the resource *rdfs:Class*, which represents a meta-class in RDFS. *sw:Project* is also defined as a subclass of *rdfs:Resource*, which is the most general class in the class hierarchy defined by RDF-Schema. The *rdfs:subclassOf* property is defined as transitive.

Properties (predicates) are defined by typing them with the resource *rdfs:Property*, which is the class of all properties. Furthermore, the domain and range of a property can be restricted by using the properties *rdfs:range* and *rdfs:domain* to define value restrictions on properties. For example, the property *sw:hasHomepage* has the domain *sw:Project* and a range *rdfs:Resource* (which is compliant with the use of *sw:hasHomepage* in Figure 13.1). Using these definitions, RDF data can be tested with compliance regarding a particular RDF Schema specification.

RDF-Schema defines more modeling primitives:

- The property *rdfs:label* allows to define a human readable form of a name.
- The property *rdfs:comment* enables comments.
- The property *rdfs:subPropertyOf* indicates that a property is usually subsumed by another property. For example the *fatherOf*-property is subsumed by the *parentOf* property, since every father is also a parent.

- The properties *rdfs:seeAlso* and *rdfs:isDefinedBy* are used to indicate related resources.
- The classes *rdfs:ConstraintResource* and *rdfs:ConstraintProperty* are used to allow the definition of more advanced constraint mechanisms.

As a convention an application can expect to find an RDF-Schema document declaring the vocabulary associated with a particular namespace at the namespace URI.

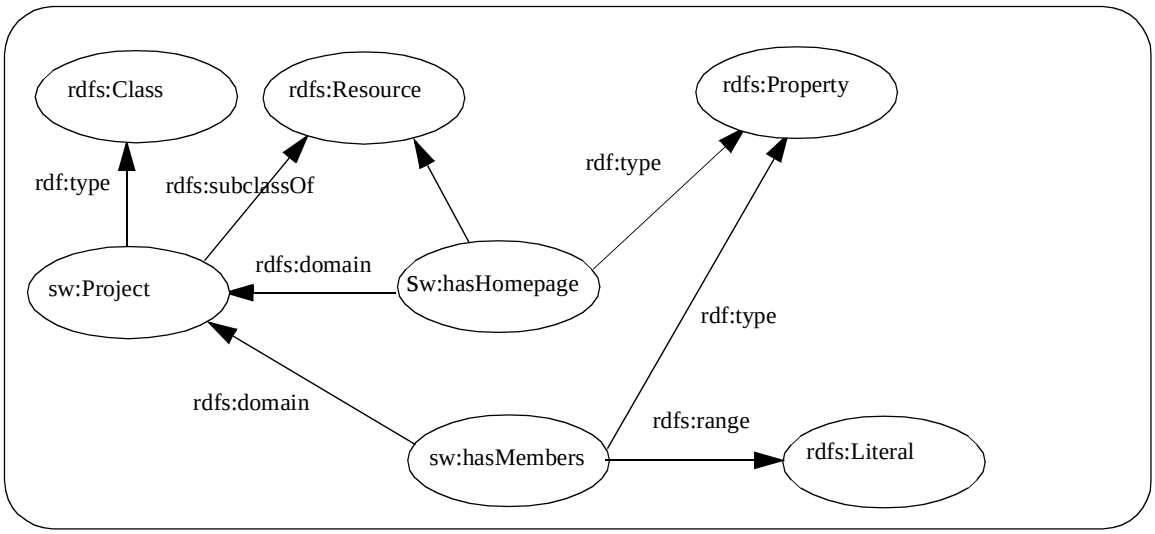


Fig. 13.6. Example of an RDF Schema

13.3.3 RDF Syntax

In order to facilitate interchange of the data that is represented in RDF a concrete serialization syntax is needed. XML [Bray et al., 1998] is an obvious choice, however, it is worth noting that the data model is not tied to any particular syntax and can be expressed in any syntactic representation (or vice versa, extracted from other forms of data, e.g., Topic Maps (ISO/IEC 13250)). Furthermore, because of the XML serialization the RDF syntax definition is rather complicated. RDF-APIs (see below) are supposed to shield developers from the details of a particular serialization syntax, and handle RDF-data as graphs. So not all the details of the XML-serialization syntax are being presented.

The RDF specification suggests two standard ways to serialize RDF data in XML: abbreviated syntax and standard syntax. Both serialization possibilities use the XML namespace mechanisms [Bray et al., 1999] to abbreviate URIs as sketched in Section 3.2. An example is given only for the abbreviated syntax. The abbreviated syntax is very close to how one would intuitively model data in XML and also very close to the data representation part of SOAP (cf. [Box et al., 2000]).

Figure 13.7 shows the XML-serialization of the RDF graphs given in Figure 13.2 and Figure 13.3. In this figure the namespace declarations have been left out. To make the RDF-snipped in Figure 7a correct RDF document, it is necessary to add the following namespace declaration:

```
<?xml version='1.0'?>
<rdf:RDF
     xmlns="http://www.SemanticWeb.org/schema-daml01#"
     xmlns:rdf=  "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
<Project rdf:about="http://www.daml.org/projects/#11">
   <hasHomepage>
         <rdfs:Resource rdf:ID="http://www-db.stanford.edu/OntoAgents">
           <dc:Creator>Stefan Decker</dc:Creator>
         </rdfs:Resource>
   </hasHomepage>
</Project>
```

Fig. 13.7. XML Serialization

```
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:dc="http://purl.org/DC#">
```

together with the following end-tag: `</rdf:RDF>`

XML documents can carry RDF code, which is mapped into an RDF data-model instance. The start and end of RDF Code is indicated by tags <rdf:RDF> and </rdf:RDF>. These element tags are optional, if the processing application already knows that it has to expect RDF data.

13.4 Semi-Structured Data

13.4.1 Introduction

As argued by the database community, semi-structured data is particularly suitable for solving the problems described in the previous section (cf. [Abiteboul, 1997] for an elaborated introduction). Roughly speaking, semi-structured data is data that is neither raw data, nor very strictly typed as in conventional database systems. Semi-structured data arises often when integrating several (possibly structured) sources. For an example, consider the integration of two retailer databases: one retailer might represent customer addresses as strings and the other as structured tuples. The retailers would probably represent dates, prices, and invoices differently. Information present in one source is possibly missing in the other. More generally, we expect a wide heterogeneity in the organization of data from different sources and not all can be resolved by an integration mechanism. Therefore we need a data model to represent the heterogeneity.

13.4.2 Graph-based Data

OEM (*Object Exchange Model*) is a popular semi-structured data model (cf. [Goldman et al., 1996], [Suciu, 1998]) and was developed for integration tasks in the Stanford TSIMMIS project [Garcia-Molina et al., 1995]. We follow the definition of OEM as used in [McHugh et al., 1997] and [Abiteboul et al., 2000] and formalized in [Erdmann, 2001].

Definition 13.1 Given a set `oid` of object identifiers. A *OEM database* $\boldsymbol{D}$ is a tuple $\boldsymbol{(G, \lambda_E, R, val)}$ with:

- a finite, directed graph $\boldsymbol{G = (V, E, \varphi)}$. The set of vertices $\boldsymbol{V} = \boldsymbol{V}_a \cup \boldsymbol{V}_c$ is disjointedly partitioned in a set $\boldsymbol{V}_a$ of atomic vertices and $\boldsymbol{V}_c$ of complex vertices with $\boldsymbol{V}_a, \boldsymbol{V}_c \subseteq \texttt{oid}$. $\boldsymbol{E}$ is a set of edges, and φ: $\boldsymbol{E} \rightarrow \boldsymbol{V}_c \times \boldsymbol{V}$ is a mapping from the set of edges of the graph to pairs of vertices. The first vertex is called the *start node*, and second vertex is called the *end node* of the edge. Please note that atomic vertices can not be the start node of an edge.
- a annotation mapping λ_E: $\boldsymbol{E} \rightarrow \texttt{string}$.

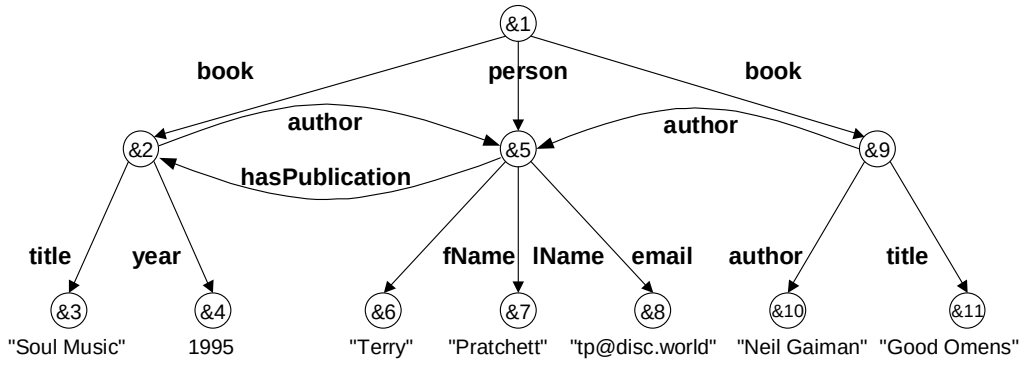


Fig. 13.8. OEM Example

- a finite set ***R*** of names and a function ***name:*** $\boldsymbol{R \rightarrow V}$, which associates names with vertices from the graph. The function identifies roots of the graph.
- a function ***val***: $\boldsymbol{V}_a \rightarrow$ `int` $\cup$ `string` $\cup$ `date` $\cup$ .... which maps atomic vertices $\boldsymbol{V}_a$ to values. The sets `int`, `string`, `date` etc. are pair-wise disjoint.

Every vertex in ***D*** must be reachable from at least one vertex ***name(N)***, with $\boldsymbol{N \in R}$. A vertex **v** is reachable from a vertex **w** iff there exists a finite sequence of edges $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \ldots \boldsymbol{e}_n$ ($\boldsymbol{e}_i \in \boldsymbol{E}$ für $1 \leq \boldsymbol{i} \leq \boldsymbol{n}$) such that the start node of $\boldsymbol{e}_1$ = **w** and the end node of $\boldsymbol{e}_n$= **v** and the start node of $\boldsymbol{e}_i$ is the end node $\boldsymbol{e}_{i-1}$ for $2 \leq \boldsymbol{i} \leq \boldsymbol{n}$.

Figure 13.8 presents an example of an OEM database.

The RDF data model is almost identical to the data model of OEM. However, RDF takes into account various requirements for data representation imposed by the Web (e.g., URIs and linkage between data residing on different servers, namespaces etc.) that were not considered in prior approaches. RDF is not only suitable to represent metadata, but also any other kind of machine-processable data on the Web.

13.5 Frames and Semantic Nets

Another possible viewpoint on RDF is the perspective of a Frame-based Knowledge Representation Language (cf. [Minsky, 1975], [Fikes & Kehler, 1985]). A frame is similar to an object known from object oriented programming, and may have slots with associated values. Frame-systems often have *procedural attachments*, which are similar to triggers in the database context. Similar to frame systems are semantic networks. A semantic network is a data structure typically consisting of labeled nodes and labeled, directed arcs [Shapiro & Rapaport, 1987], with typically predefined semantic relationships like '*is_a*' or '*part_of*'.

13.6 On the Relationship of RDF, Semi-Structured Data, and Semantic Networks

When comparing RDF and OEM the following observations can be made:

- RDF and OEM are both based on labeled directed graphs - they rely on the same data structure.
- OEM does not specify how the OIDs are generated, whereas RDF requires them to be URIs.

- RDF nodes and edges are labeled are URIs - unlike OEM.
- RDF does not require the declaration of root objects as OEM does.
- RDF just supports ‘String’ as its atomic value type, whereas OEM supports more types such as integer etc.

However, all those differences are minor. Since both data formats are very similar, it means that the technologies that were developed for semi-structured data (like OEM) are directly applicable to RDF (e.g., storage infrastructure like LORE [Abiteboul et al., 1997a], interoperation techniques [Abiteboul et al., 1997b], [Papakonstantinou et al., 1996]).

Compared to frame-languages and semantic networks these languages usually focus on the predefined semantic relationship like *is_a* or *part_of*. Not taking these predefined relationships into account, RDF is very similar to semantic networks. However, the primitives defined in semantic networks and frame based systems are necessary for the definition of domain models or ontologies. From this similarity it follows that the necessary knowledge representation primitives can be incorporated into RDF. Within RDF those primitives constitute special vocabularies, which do not belong to the core data model. The next chapter clarifies how knowledge representation languages may be defined on top of RDF.

Chapter 14 The Roles of XML and RDF in the Semantic Web(1)

14.1 Knowledge Representation for the Web

The representation of Knowledge on the Web (in the sense of machine-understandable data) poses new requirements for Knowledge Representation Languages since the Web has new characteristics, like the inherent distribution, diversity and heterogeneity that needs to be addressed before Knowledge Representation can take place. In Section 13.2 requirements for Knowledge Representation on the Web were identified. Semi-structured data is fulfilling some of the above-mentioned requirements.

Also, XML is often regarded as a possible candidate data model. The following discussion provide arguements why XML is not suitable as a basis for Knowledge Representation, and how to define ontology representation languages on top of RDF.

14.2 XML

XML is already widely known in the Internet community, and is the basis for a rapidly growing number of software development activities. It is designed for markup in documents of arbitrary structure, as opposed to HTML, which was designed for hypertext documents with fixed structures.

A well-formed XML document is a balanced tree of nested sets of open and closed tags, each of which can include several attribute-value pairs. There is no fixed tag vocabulary or set of allowable combinations, so these can be defined for each application. In XML 1.0 this is done using a document type definition to enforce constraints on which tags to use and how they should be nested within a document. A DTD defines a grammar to specify allowable combinations and nestlings of tag names, attribute names, and so on. Developments are well underway at W3C to replace DTDs with XML-schema definitions [Thompson et al., 2000] but although XML schemas offer several advantages over DTDs, their role is essentially the same: They define a grammar for XML documents.

XML is used to serve a range of purposes:

- *Serialization syntax for other markup languages*. For example, the Synchronized Multimedia Integration Language (SMIL) [Hoschka, 2000] is syntactically just a particular XML DTD; it defines the structure of a SMIL document. The DTD is useful, because there exists a common understanding of the meaning of the DTD elements and the structure of the DTD.

- *Semantic mark-up of Web pages*. An XML serialization can be used in a Web page with an XSL style sheet to render the different elements appropriately.

- *Uniform data-exchange format*. An XML serialization can also be transferred as a data object

1. This section is an extended and revised version of [Decker et al., 2000c].

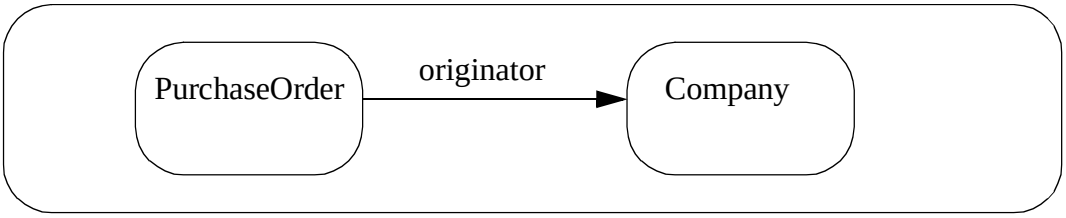


Fig. 14.1. Binary Relationship

between two applications.

It is important to note that a DTD (or XML-Schema) only specifies syntactic conventions; any intended semantics are outside the realm of the XML specification. Figure 14.1. show a binary relationship 'principal' between two concepts: *Purchase Order* and *company.* Tab. 14.1. depicts

Encoding DTD	Example XML Instance Data

`<!ELEMENT PurchaseOrder (originator)>`
`<!ATTLIST PurchaseOrder`
`  id ID #REQUIRED>`
`<!ELEMENT originator (Company)>`
`<!ATTLIST Company id ID #IMPLIED>`
	`<PurchaseOrder id="X">`
	`  <originator>`
	`  <Company id="Y"/>`
	`  </originator>`
	`<PurchaseOrder>`

`<!ELEMENT originator (PurchaseOrder,`
`  Company)>`
`<!ELEMENT PurchaseOrder (#CDATA)>`
`<!ELEMENT Company (#CDATA)>`
	`<originator>`
	`  <PurchaseOrder>X</PurchaseOrder>`
	`  <Company>Y</Company>`
	`</originator>`

`<!ELEMENT PurchaseOrder (id,`
`  principal)>`
`<!ELEMENT id (#CDATA)>`
`<!ELEMENT originator (Company)>`
`<!ELEMENT Company (id)>`
	`<PurchaseOrder>`
	`  <id>X</id>`
	`  <originator>`
	`  <Company>`
	`  <id>Y</id>`
	`  </Company>`
	`  </originator>`
	`</PurchaseOrder>`

`<!ELEMENT rel EMPTY>`
`<!ATTLIST rel`
`  src CDATA #REQUIRED`
`  type CDATA #REQUIRED`
`  dest CDATA #REQUIRED>`
	`<rel`
	`  src="X"`
	`  type="originator"`
	`  dest="Y"/>`

`<!ELEMENT PurchaseOrderInfo`
`  (Company)>`
`<!ATTLIST PurchaseOrderInfo`
`  orderID ID #REQUIRED>`
`<!ELEMENT Company (#CDATA)>`
	`<PurchaseOrderInfo orderID="X">`
	`  <Company>Y</Company>`
	`</PurchaseOrderInfo>`

Tab. 14.1. Different Representation Possibilities of a Binary Relationship in XML

several encoding possibilities of the binary relationship in XML. Just from the depicted DTD fragments it is impossible to determine what the relationship is and what the concepts are. So other parties receiving the data have to perform an interpretation of the XML document. This example is a very simple case, a single binary relationship. Significantly more encoding options exist in cases of multiple, ordered, etc. relationships. XML is used in scenarios as depicted in Figure 14.2: in a simple peer-to-peer communication scenario two applications try to communicate to each other. Both applications agree on the use and intended meaning of the document structure given by DTD A, but if frequent changes occur or new communication partners are added, the XML-based

exchange is not suitable. Before data exchange can be established between the applications, a model of the domain of interest has to be built, since it is necessary to clarify what kind of data is sent from the first application to the second.

The domain model is usually described in terms of objects and relations (e.g., in UML or Entity Relationship Modeling). Even if an explicit model is not constructed, entities and relationships between entities are the usual starting points for data description. From the domain model a DTD or an XML Schema is constructed. Since a DTD just describes a grammar there exist multiple possibilities to encode a given domain model into a DTD (see the previous example Figure 13.7 and Figure 14.1, [Erdmann & Studer, 2001] for a more elaborate discussion). So the direct connection from the DTD to the domain model is lost and cannot be easily reconstructed.

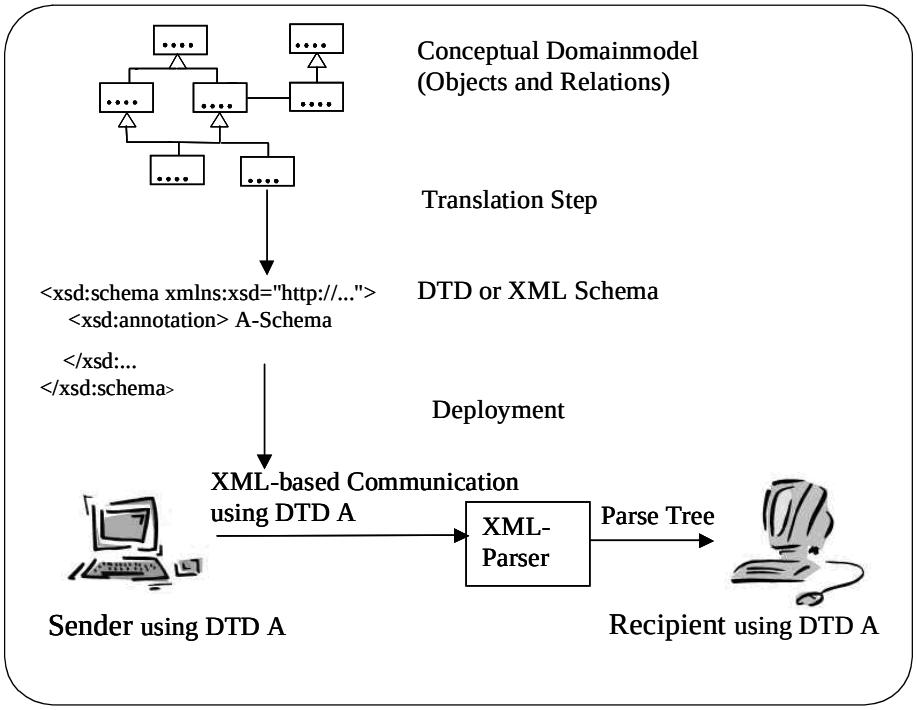


Fig. 14.2. Peer2Peer Data Exchange

The advantage of using XML in this case is limited to the reusability of the parsing software components. Certainly, such reuse is relevant and useful. But the scenario depicted in Figure 14.2 is a one-on-one communication with parties who have reached agreement in advance. It does not account for the dynamics of the Web, where one often has to cope with multiple and frequently changing communication partners who may not be known in advance. The second scenario (sketched in Figure 14.3) occurs frequently: new business partners have to be added to an existing relationship, new information sources become available etc. Since change may be a very frequent operation, it is important to reduce the costs of adding new communication partners as much as possible. However, using XML and DTDs (or XML Schemas) for this is a high effort task (see Figure 14.4): for the adaptation to two communication partners it is vital to define mappings between the different DTDs (XML-Schemas). A direct mapping based on the different DTDs is usually not suitable, since the task is not to map one grammar to the second one, but to map objects and relations from one domain of interest to another one. The mapping of the grammars may be deduced from the domain mappings. So companies have to define the mappings between the

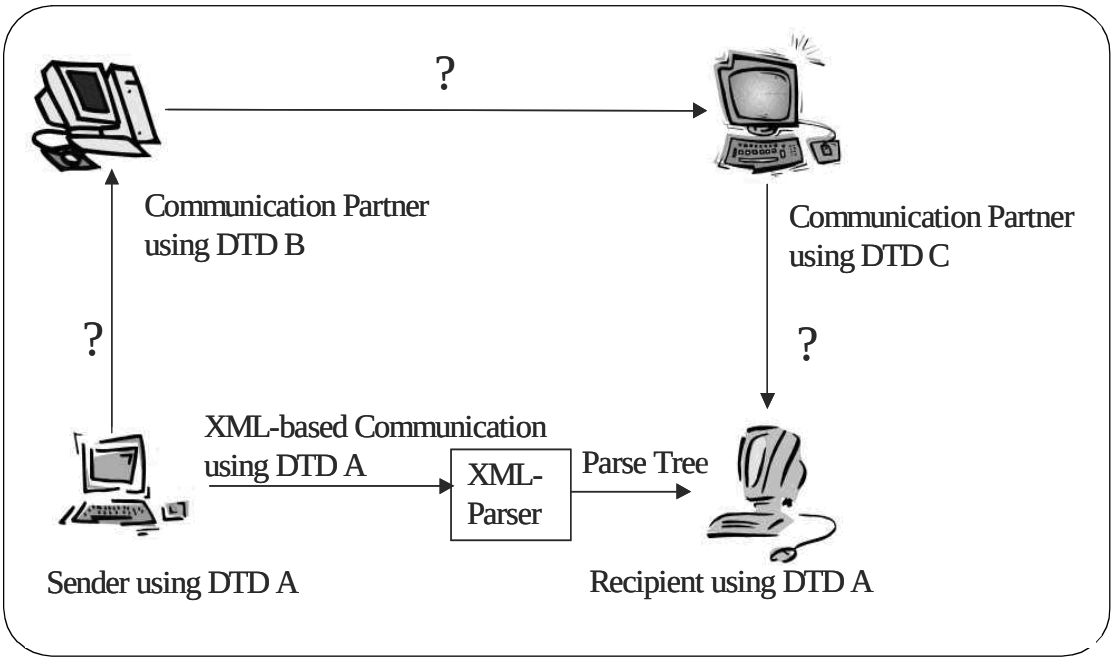


Fig. 14.3. New Communication Partner enters an existing Communication

different domain models, not between the different DTDs. Defining the mapping between DTDs is a second step, after the correct domain mappings are identified. The following tasks have to be performed.

1. *Reengineering of the original Domain Model from the DTD or XML Schema* (see Figure 14.4, step 1). Since we want to map semantic entities (and not part of the grammars) it is necessary to reconstruct the original domain model that was used to construct the DTD (for both DTDs).
2. *Establishing mappings between the entities in the domain model*. Since now the semantic model is under consideration (talking about concepts from the domains of interested entities) one is able to define mappings between the different concepts and relationships. Here techniques developed in the Knowledge Engineering and Database area are often helpful (see [Jannink et al., 1999], [Noy & Musen, 1999], [McGuinness et al., 2000]).
3. *Defining translation procedures for XML documents*. Since XML documents are exchanged, the mappings established in task 2 have to be translated into mapping procedures for XML documents. This is again a high-effort task, since it depends on the particular encoding chosen to construct the initial DTDs.

Although step (2) is already non-trivial, additional effort has to be spent in the particular encodings: these are the translations of the original domain model into an XML-DTD (consider the reengineering of the domain model in case of additional communication partners, and the translation of the established mappings back to translations of XML documents (see Figure 14.4)). Much of this additional effort can be saved by using a more suitable formalism than pure XML, since then the translations (‘media breaks’) are not necessary anymore.

In a nutshell, this means that XML is very suitable for data interchange between applications that both know what the data is about, but not when the addition of new communication partners occurs frequently.

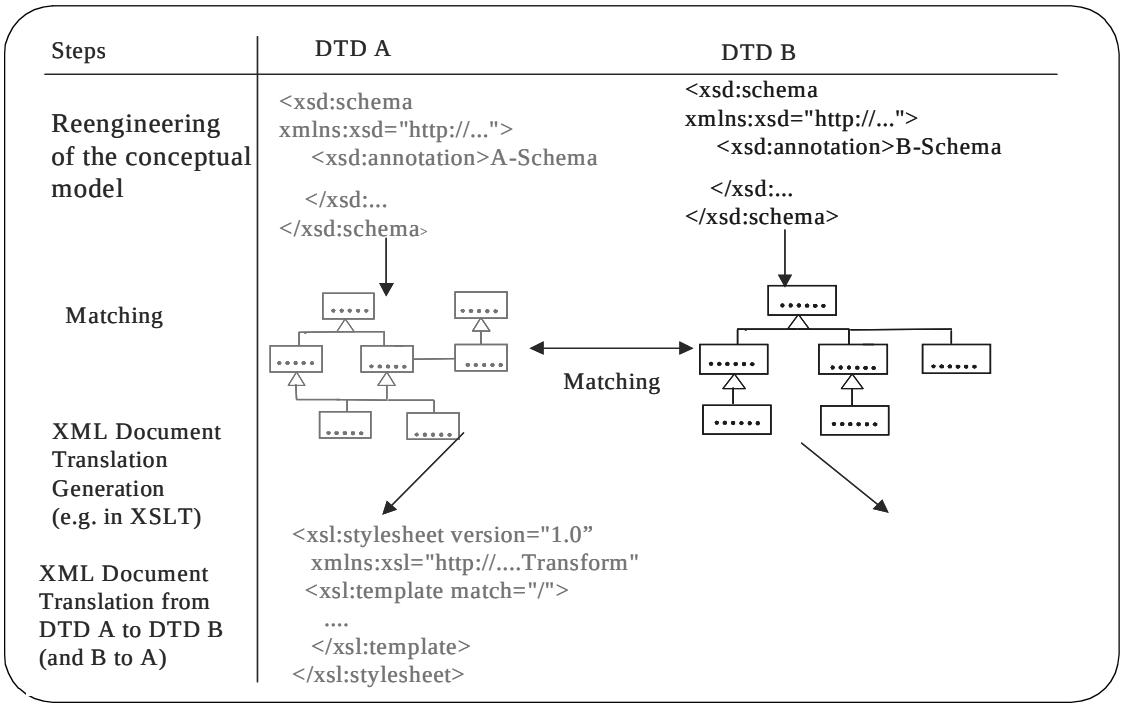


Fig. 14.4. Necessary Steps to align two different DTDs

14.3 Data Exchange and RDF

RDF defines a nested object-attribute-value structure. The universal expressive power holds for RDF: the next section demonstrates extension techniques for RDF.

When it comes to semantic interoperability, RDF has significant advantages over XML in that semantic units are given naturally through its object-attribute structure: all objects are independent entities. A domain model, defining objects and relationships of a domain of interest, can be represented naturally in RDF so translation steps, as required when using XML, are not necessary. To find mappings between two RDF descriptions, techniques from Knowledge Representation are directly applicable. Of course, this does not solve the general interoperability problem of finding semantic-preserving mappings between objects. However, the usage of RDF for data interchange raises the level of potential reuse much beyond parser reuse, which is all that one would obtain from using plain XML.

Furthermore, since RDF describes a layer independent of XML, the RDF model (and software using the RDF model) can still be used, even if the current XML syntax is changed or disappears.

Another way of phrasing the advantages of RDF is as follows. Ideally, a universally shared KR language to support the Semantic Web is desirable. For a variety of pragmatic and technological reasons, this ideal is not achievable in practice and one has to live with a multitude of metadata representations. RDF contains as much KR technology as can be shared between widely varying metadata languages. Furthermore, as will be shown in the next two sections, the RDF Schema language is powerful enough to define richer languages on top of the relatively limited primitives of RDF.

It has been argued that RDF is more suitable for representing formal data on the Web. However, the data model of RDF is quite simple, so one might think that it is not universally expressive (as XML is, because XML defines grammars). However, in the following section ways to express complex data in RDF is described, along with methods to exploit the common object structure for interoperability purposes.

14.4 Building Languages on Top of RDF

This section shows how to define data models on top of RDF. The techniques are universally applicable. Before doing so, Brachman's distinction of three layers in a knowledge representation system [Brachman, 1979] is recalled:

- *Implementation level*: consists of data structures of a particular implementation of a Knowledge Representation System.
- *Logic level*: defines, in an abstract way, the inferences that are performed by the Knowledge Representation System.
- *Epistemological level*: defines adequate representation primitives. These are usually the primitives used by a knowledge engineer.

Usually the epistemological level is defined by a grammar that defines the language of interest. However, the representation primitives can also be regarded as an (representation) ontology, and so as objects of a particular domain. Using this view, domain-modeling techniques are again suitable for defining a knowledge representation language. As a consequence, the epistemological level of a representation language is itself given by an ontology that defines the terms of the representation language. Defining an ontology in RDF means defining an RDF Schema, which defines all the terms and relationships of the particular language. Thus ontology and instances, or meta-language and language, are represented by the same formalism and can be processed with the same tools.

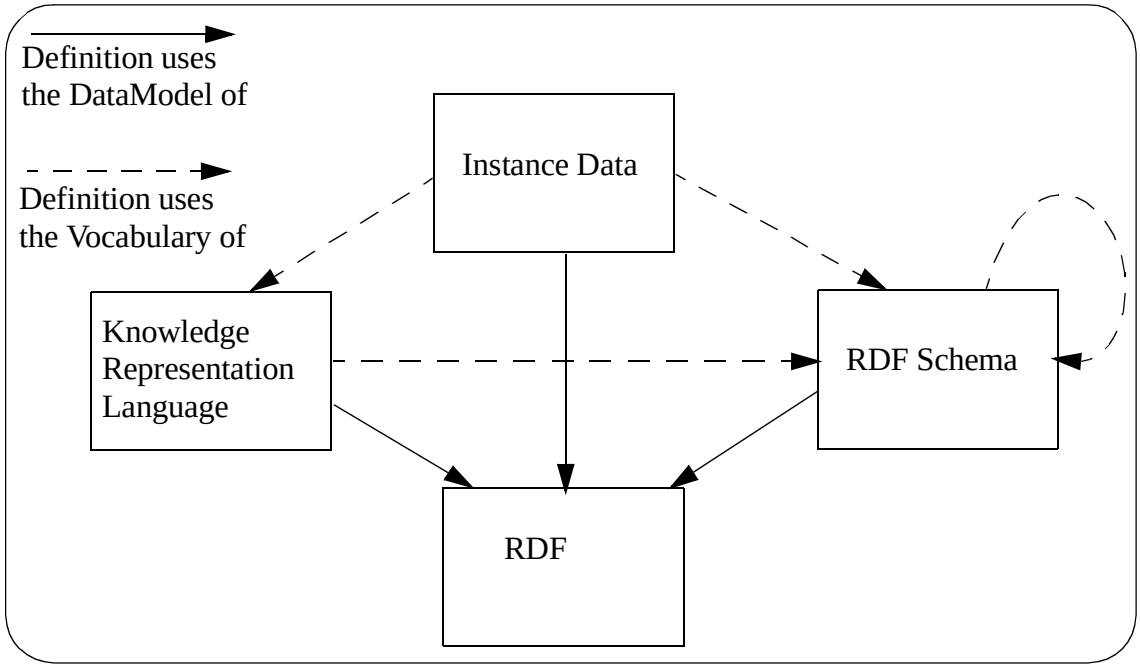


Fig. 14.5. Relationship between RDF, RDF Schema, Languages and Instances Data

Figure 14.5 visualizes the relationship between representation languages, RDF and RDF schema. The current proposal can be summarized as follows: a mechanism for encroaching the very simple data-model of RDF upon modeling primitives from other representation languages (called L). This mechanism works as follows:

- Step 1: use RDF Schema to describe the modeling primitives from language L (effectively writing the meta-ontology of L into an RDF Schema).
- Step 2: use the resulting RDF Schema document (containing the meta-ontology of L) to describe specific instances in L.
- Step 3: use the RDF Schema documents from steps 1 and 2 to describe instances of the specific language L modeled in step 2.

14.5 Summary

The chapter argues that XML is not ideal as a foundation for the Semantic Web, rather that RDF is more suitable. Ontologies are one of the building blocks of the Semantic Web, and mechanisms are required to build ontologies in RDF. The chapter presents an extension mechanism for RDF, which allows for the creation of ontology languages on top of the RDF infrastructure.

Chapter 15 Ontology Languages on Top of RDF

In this chapter several ontology languages are presented, defined on top of RDF and based on the principles disussed in the previous chapter. The starting point is with OIL, the result of a European research project. DAML+OIL, which further developed OIL, is then presented. The chapter closes with a critical assessment of both languages that may be regarded as successors to Ontobroker's ontology representaton languages as they are suitable for exchange and processing with the same machinery as the instance data.

15.1 OIL

In this section the representation and inference language OIL (cf. [Horrocks et al., 2000], [Fensel et al., 2000], [Decker et al., 2000d]) is presented along with a demonstration of how it can be embedded into the Semantic Web infrastructure by placing it on top of the RDF/RDF-Schema layer. OIL aims to combine the most attractive features of frame based languages (i.e., their more intuitive syntax and their acceptability within the ontology community) with the expressive power and formal rigor of a very expressive description logic. The OIL language is based on the *SHIQ* description logic (cf. [Horrocks et al., 1999]).

OIL unifies three important aspects provided by different communities (see Figure 14.3): (1) formal semantics and efficient but limited reasoning support as provided by Description Logics, (2) epistemologically rich modeling primitives as provided by the Frame community, and (3) the relationship to semi-structured data and the standard RDF data model for information interchange and integration as provided by the Web community. The following describes the Description Logics and Frame Roots in more detail.

Description Logics (DL)

DLs describe knowledge in terms of concepts and role restrictions that are used to automatically derive classification taxonomies. The main effort of the research in knowledge representation is in providing theories and systems for expressing, accessing and reasoning with formalized, machine accessible knowledge. Description Logics (cf. [Brachman & Schmolze, 1985], [Baader et al., 1991]), also known as terminological logics, form an important and powerful class of logic-based knowledge representation languages.(1) They result from early work on the formalization of semantic networks. DLs attempt to find a fragment of first-order logic with high expressive power which still has a decidable and efficient inference procedure (cf. [Nebel, 1996]). Implemented systems include BACK, CLASSIC, CRACK, FLEX, K-REP, KL-ONE, KRIS, LOOM, and YAK.(2) A distinguishing feature of DLs is that classes (usually called concepts) can be defined intentionally in terms of descriptions that specify the properties that objects must satisfy to belong to the concept. These descriptions are expressed using a language that allows the construction of composite descriptions, including restrictions on the binary relationship's (usually called roles) connecting objects. Various studies have examined extensions of the expressive power for such languages and

1. Pointers to papers, project, and research events in this area can be found at http://dl.kr.org/.

2. http://www.research.att.com/sw/tools/classic/imp-systems.html

the trade-off in computational complexity for deriving *is-a* relationships between concepts in such a logic (and also, although less commonly, the complexity of deriving instance-of relationships between individuals and concepts). Despite the discouraging theoretical complexity results, there are now efficient implementations for DL languages (cf. [Borgida & Patel-Schneider, 1994], [MacGregor, 1994], [Horrocks and Patel-Schneider, 1999]), see for example DLP[(1)] and the FaCT system.[(2)] OIL inherits from Description Logic its *formal semantics* and the *efficient reasoning support* developed for these languages. In OIL *subsumption* is decidable and FaCT is an efficient reasoner. In general, subsumption is only one of several reasoning tasks for working with an ontology. Other reasoning tasks are, for example, instance classification, query subsumption and query answering over classes and instances, navigation through ontologies, etc. However, many of these can be reformulated in terms of subsumption checking. Others may lead to different super- and subsets of the current OIL language version. OIL can be seen as a starting point for exploring the plethora of possible choices in designing Ontology languages and characterizing them in terms of their pros and cons.

Frame-based systems

The central modeling primitives of predicate logic are predicates. Frame-based and object-oriented approaches take a different point of view. Their central modeling primitives are classes (i.e., frames) with certain properties called attributes. These attributes do not have a global scope but are only applicable to the classes they are defined for (they are typed) and the ‘same’ attribute (i.e., the same attribute name) may be associated with different value restrictions when defined for different classes. A frame provides a certain context for modeling one aspect of a domain. Many additional refinements of these modeling constructs have been developed and have led to the success of this modeling paradigm. Many frame-based systems and languages have been developed, and under the name object-orientation the paradigm has also conquered the software engineering community.

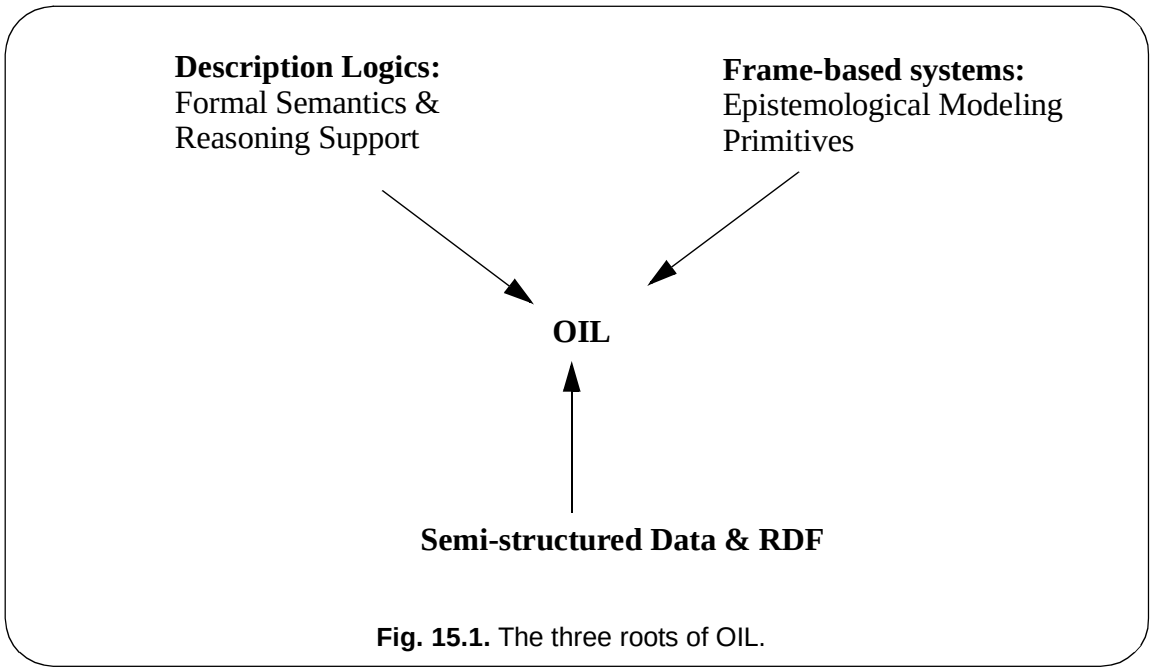


Fig. 15.1. The three roots of OIL.

1. http://www.bell-labs.com/user/pfps/
2. http://www.cs.man.ac.uk/~horrocks/software.html

Therefore, OIL incorporates the *essential modeling primitives* of frame-based systems into its language. OIL is based on the notion of a concept and the definition of its superclasses and attributes. Relations can also be defined not as attributes of a class, but as independent entities having a certain domain and range. Like classes, relations can be arranged in a hierarchy. In DLs, roles are not defined for concepts. Actually, concepts are defined as subclasses of role restriction. One could rephrase this in a frame context as follows: a class is a subclass of its attribute definitions (i.e., all instances of the class must fulfil the restrictions defined for the attributes). However, asking which roles could be applied to a class does not make much sense for a DL, as nearly all slots can be applied to a class. With frame-based modeling the implicit assumptionis made that only those attributes can be applied to a class that are defined for this class.

Figure 15.2 presents an ontology represented in OIL:.The ontology consists of a list of class

```
class-def primitive animal% animals are a class
class-def primitive plant% plants are a class
      subclass-of NOT animal% that is disjoint from animals
class-def primitive tree
      subclass-of plant% trees are a type of plants
class-def primitive branch
       slot-constraint is-part-of% branches are parts of trees
           has-value tree
class-def primitive leaf
        slot-constraint is-part-of% leaves are parts of branches
           has-value branch
class-def defined carnivore% carnivores are animals
        subclass-of animal
        slot-constraint eats% that eat only other animals
           value-type animal
class-def defined herbivore% herbivores are animals
        subclass-of animal
        slot-constraint eats% that eat only plants or parts of plants
           value-type plant OR (slot-constraint is-part-of has-value plant)
class-def primitive giraffe% giraffes are animals
        subclass-of animal
        slot-constraint eats% and they eat leaves
           value-type leaf
class-def primitive lion
        subclass-of animal% lions are also animals
```

Fig. 15.2. Animal Ontology represented in OIL

definitions (class-def) and *slot definitions* (slot-def) (slot-defs are omitted in the example). A class definition (class-def) associates a class name with a class description. A class-def consists of the following components (any or all of which may be omitted):

- *The type of definition*. This can be either primitive or defined; if omitted, the type defaults to primitive. When a class is primitive, its definition (i.e., the combination of the following subclass-of and slot-constraint components) is taken to be a necessary but not sufficient condition for membership of the class.

- *Subclass-of*. A list of one or more class-expressions, the structure of which will be described below. The class being defined in this class-def must be a sub-class of each of the class expressions in the list.

- *Slot-constraint*. A list of zero or more slot-constraints, the structure of which will be described below. The class being defined in this class-def must be a sub-class of each of the slot-constraints in the list (note that a slot-constraint defines a class).

A class-expression can be either a class name, a slot-constraint, or a boolean combination of class expressions using the operators AND, OR or NOT. Note that class expressions are recursively defined, so that arbitrarily complex expressions can be formed.

A slot-constraint is a list of one or more constraints (restrictions) applied to a slot. A slot is a binary relation (i.e., its instances are pairs of individuals), but a slot-constraint is actually a class definition-its instances are those individuals that satisfy the constraint(s). For example, if the pair (Leo, Willie) is an instance of the slot eats, Leo is an instance of the class lion and Willie is an instance of the class wildebeest, then Leo is also an instance of the *has-value* constraint wildebeest applied to the slot eats. A slot-constraint consists of the following main components:

- *Name*: a slot name (a string). The slot is a binary relation that may or may not be defined in the ontology. If it is not defined it is assumed to be a binary relation with no globally applicable constraints, i.e., any pair of individuals could be an instance of the slot.

- *Has-value*: a list of one or more class-expressions. Every instance of the class defined by the slot constraint must be related via the slot relation to an instance of each class-expression in the list. For example, the has-value constraint: slot-constraint eats has-value zebra, wildebeest defines the class each instance of which eats some instance of the class zebra and some instance of the class wildebeest. Note that this does not mean that instances of the slot-constraint eat only zebra and wildebeest: they may also be partial to a little gazelle when they can get it.

- *Value-type* - a list of one or more class-expressions. If an instance of the class defined by the slot-constraint is related via the slot relation to some individual x, then x must be an instance of each class-expression in the list.

In the following an RDF-Schema definition for OIL is provided. RDF relies on namespaces and Namespace prefixes: RDF-vocabulary is prefixed with a Namespace prefix, which is resolved to a complete URI by an RDF processor. The following namespace prefixes are used: ‘oil’: for OIL; *rdf* and *rdfs* for RDF; and RDF-Schema *dc* and *dcq* for Dublin Core Vocabulary and Qualifiers, respectively. The usual RDF-Dublin Core encoding, which can be obtained at [Dublin Core] is used for Dublin Core. OIL relies heavily on RDF-Schema itself, since OIL is defined as an extension of RDF-Schema and reuses concepts of RDF-Schema as much as possible. This strategy was taken to facilitate the reuse of existing RDF-Schema-based applications and tools. However, certain extensions of RDF-schema were required. For example, OIL allows implicit definitions of classes in the form of boolean operators (AND, OR, NOT) as value of the subclass-of relation, whereas in RDFS the value of the *subClassOf* statement is always an explicit class. *oil:ClassExpression* was introduced as a placeholder class for the three boolean operators, which are also modeled as classes, to allow their use as value for the subClassOf statement.

15.2 DAML+OIL

DAML+OIL is an ontology language developed by an ad-hoc US-American/European committee in the course of the DARPA Agent Markup Language program (DAML)[1]. DAML+OIL supersedes OIL and provides some enhancements:

- DAML+OIL introduces XML Schema data types like integer, string or real. Properties are distinguished in two disjoint classes: Object properties, whose value is an instance of a class, and data-type properties, whose value is instances of the XML Schema data types.
- A possibility to explicitly import another DAML+OIL ontology containing definitions that apply to the current DAML+OIL ontology.
- An explicit syntactical way to define enumerations.

15.3 An Assessment of OIL and DAML+OIL

The amalgamation of semi-structured data and knowledge representation techniques offers the advantages that efficient storage and query facilities can be provided for large and multiple ontologies expressed using Description Logics, since ontologies are expressed as a special vocabulary of semi-structured data. However, using Description Logic requires the overhead of deploying an interference engine, and in some applications it might not be appropriate to pay for the overhead.

[Brachman et al., 1991] gives some hints when a Description Logic system might be useful:

- Description Logic systems support the automatic classification of objects into a hierarchy. Thus, a Description Logic may be beneficial where a large set of objects can be naturally represented in terms of *features* or *roles*.
- If the domain is constantly evolving Description Logics might be a suitable representation mechanism: DL allow the user to maintain an incomplete view of the world without deducing incorrect information about the world.
- Sometimes the closed-world assumption (as usually applied to databases) may be incorrect. Description Logics do not support a closed world assumption.

It seems feasable that most ontologies on the Web are not ontologically sophisticated - the class hierarchy is given explicitly and is usally not very deep, since most ontologies cover rather simple data formats like calendars[2], business cards[3], business processes etc. So, the arguments given above of sophisticated objects that need to be classified do not apply. At least in the early stages of the Semantic Web it does not seem reasonable to demand that every developer should support a fully-fledged description logic. Insisting on Description Logic technology would likely hamper broad scale acceptance of ontology technology since description logic engines are not a commodity technology. What is proposed here instead is the choice of a reasonable subset of a description logic (e.g., similar to UML), which already has a broad acceptance and can be supported cheaply. In the

1. see http://www.daml.org

2. E.g., see the calendar ontology at http://ilrt.org/discovery/2001/06/schemas/ical-full/hybrid.rdf. The ontology consists of 43 classes, which are three levels deep.

3. http://www.w3.org/TR/vcard-rdf

long-term, Description Logics might offer considerable advantages e.g., for distributed ontology development, since the classifier is able to classify distributively-defined classes into a coherent hierarchy - something that would require a lot of work if performed manually.

Chapter 16 Conclusion and Future Work

> *"Indeed, the danger this time is that we get six hundred people creating reasoning engines in their garages across the land. But if they try to patent what they're doing, each one of them thinking they've found the grand solution first, or if they build palisades of proprietary formats and use peculiar, undocumented ways of doing things, they will just get in the way."*
> Tim Berners-Lee in *Weaving the Web*, 1999, p. 196

16.1 Summary

This thesis began with the observation that knowledge and Knowledge Management have become important for our economies. It revisited the notion of knowledge in Computer Science with the focus on Artificial Intelligence and Computer Supported Cooperative Work. Different dimensions and viewpoints of Knowledge Management were presented, which were used to develop the Knowledge Management Cube. The cube unites major aspects of Knowledge Management strategies, and was used to structure and assess subsequent Knowledge Management strategies.

The main part (part II) of this thesis describes the Ontobroker system, its languages, and applications. It showed that with the Ontobroker system and languages the Web may be indeed be transformed into a knowledge base, based on ontologies that define the application domain.

The Ontobroker system focused on practical approaches to ontologies and knowledge representation. For the representation of ontologies a language (SiLRIs F-logic) was chosen based on Logic Programming and Deductive Database principles, since effective reasoning methods are known for these technologies. Means were then investigated to make logic programming languages more suitable for the definition of ontologies. These were:

1. the usage of F-Logic, which allows for modeling of ontologies with representation primitives known from frame based languages, and
2. the definition of an efficient variant of the Lloyd-Topor transformation, which allows for the writing of rules with the full syntax of full-first order logic in the bodies.

A compilation approach for compiling domain specific declarative languages, like Ontobroker SiLRI, was defined for normal logic programs and this approach was demonstrated by combining Chronolog and F-logic. The C-F-logic language helps users to reason about objects and change.

HTML-A, an annotation language for HTML, and XAL, was presented as a generalization and extension of HTML-A. XAL allows for the annotation of XML documents with ontology-based metadata. The metadata contains information about instances of classes defined in the ontology. XAL allows annotations with multiple ontologies in one document, it simplifies the annotation of structures like tables and lists, and provides means to select information with regular expressions. Based on HTML-A OntoPad was developed, to the author's knowledge the first HTML editor that supports the creation of ontology based metadata, leveraging HTML-A, and clarifying the requirements that a metadata annotation tool must possess.

Several query interfaces for querying ontology based metadata were implemented and evaluated. The requirements for such a query interface were identified, such that:

- the query interface has to provide facilities that allow the user to browse and select terms from an ontology. A visualization schema based on hyperbolic geometry was used for the browsing of ontologies. The hyperbolic tree is superior to conventional tree based visualization schemes if the *information scent* of the visualized information is high. It was argued that for high-quality ontologies the information scent is high, and the notion of information scent may be used to determine the quality of an ontology.
- the query interface needs to support the structure of the language. Two visualization schemas were suggested: the first schema was based on the structure of the query language, the second more tailored towards the object-oriented model.

$(KA)^2$, the Knowledge Annotation Initiative of the Knowledge Acquisition community, a real life case study of the usage of Ontobroker, was presented which required collaborative ontology building and annotation of HTML documents.

Ontobroker was compared with other approaches aiming at similar goals, especially with WebKB and SHOE.

The third part of the thesis examined the emerging *Semantic Web*, a Web beyond Ontobroker. The Semantic Web is first and foremost a web of interoperation, based on semi-structured data, ontologies and services. A description of how ontology representation languages may be defined on top of the RDF infrastructure was given, and an argument for why XML may not be suitable as a foundation for the Semantic Web. A quick glance was then taken at two ontology languages for the Semantic Web, to which the author of this thesis contributed: OIL and DAML+OIL.

16.2 Impact of Ontobroker

Ontobroker was one of the first approaches that deployed the Web infrastructure for distributed ontology based Knowledge Management, and was able to show some of the advantages and possibilities of formal metadata on the Web. The DARPA DAML program, a $80M research program of the Defense Advanced Research Projects Agency of the USA, cited Ontobroker as one of the inspirations[1] for the program. The ongoing work, reported in part 3 of the thesis validates many of the directions that were taken in Ontobroker. The origins of the European OntoWeb[2] network can be directly traced back to Ontobroker and (KA).[2] Finally, Ontobroker may also be regarded as a predecessor of the *Semantic Web*.[3]

16.3 Future Work

The work on Ontobroker opens a new field, and as such leads to many new questions, which are now issues in the Semantic Web field. In the following sections two lines of future work are discussed in more detail: *ontology evolution and management*, and *multiple semantics and mediation*.

1. See http://www.oasis-open.org/cover/daml-pipBAA0007.html for a copy of the orginal BAA
2. See http://www.OntoWeb.org
3. See http://www.w3.org/2001/sw/ and http://www.SemanticWeb.org

16.3.1 Ontology Evolution and Management

Experience with the $(KA)^2$ project, executed within the Ontobroker project, has shown that ontologies will change over time due to changing environmental conditions, new knowledge about the world, and so on. Maintaining different versions of an ontology will become necessary to support collaborative development and to provide different extensions of an existing ontology for different communities. Ontologies that change cause the following problems:

1. Data, applications, and other ontologies that extend the changed ontology might become inaccessible, unusable, or inconsistent after the ontology evolves.
2. Managing different versions and branches of an ontology is a laborious task that is a bottleneck for collaborative development of ontologies.

Most of the literature in ontology evolution (cf. [Heflin et al., 1999], [Klein & Fensel, 2001]]) and database-schema evolution (cf. [Roddick, 1995], [Sciore, 1991], [Franconi et al., 2000], [Lu et al., 1996]) focuses on the first problem. However, for the purpose of jointly building ontologies, the second problem is more important than the first: the effort of jointly managing and merging different versions of an ontology greatly increases the cost of establishing a joint ontology.

The problem of managing multiple versions is not unique to ontology management: a similar problem also arose in software engineering in the context of collaborative, parallel development of large software systems. A simple locking mechanism did not improve the situation, since developers tend to forget to unlock files after they are done, essentially eliminating parallel development. *Concurrent Versions System* (CVS), which is based on the *Revision Control System* (RCS)) [Tichy, 1985], and similar systems became an effective solution for software evolution and management. The collaborative internet-based, open-source software-development efforts are unthinkable without CVS-based support.

To enable incremental improvement, collaborative work, and effective management of different versions of ontologies, an *Ontology Versioning System* (OVS) is required [Oliver et al., 1999]. This system will answer structural queries such as: w*hich subclasses of Person were modified in the last month?* To answer structural queries, an OVS has to understand the semantics of the ontology representation language (e.g., the semantics of subclass). Unfortunately, we cannot use current CVS or other software-evolution systems directly. These systems rely on textual representation of information. While focusing on the textual representation works for text documents and the source of software, it does not provide sufficient support for the joint development of ontologies. Comparing textual serialization of ontologies is not sufficient since two ontologies may have completely different textual representations and still have the same semantics.

The development of an OVS has to address the following questions:

1. *Which underlying data model should we use to represent the ontologies?* The data model should be extremely general to support multiple ontology languages that are likely to co-exist on the web (e.g., UML, TopicMaps, DAML+OIL etc.) and it should support structural queries. Graph-based formalisms have proven to be very flexible as an underlying data structure for representing different ontology languages. For example, graph representations have been investigated of Description Logic based Ontology Languages (DAML+OIL), UML, and TopicMaps.
2. *Which primitives are required to capture the collaborative nature of ontology development?* Examples of such primitives include tagging changes with certain names, the ability to undo changes of certain users, the ability to enable social processes (e.g., voting for the introduction

of a new concept), and so on. Other operations include deriving different branches from an existing ontology, merging two different branches, updating a branch and so on. [Lu et al., 1996] sketch the use of a temporal object-oriented database for version control of object-oriented schemata. An OVS should reuse this work and extend it significantly to work with different underlying representations, support collaboration primitives, and allow integration of different ontology versions. The primitives of CVS and RCS deliver a good starting point.

3. *Which primitive data-model operations should OVS support?* In addition to ontology-level operations, it is necessary to provide a model of operations for representing model changes.
4. *How are different versions of the same ontology integrated?* Since the same ontology will be changed by different groups of users independently from one another, there will be a frequent need to reintegrate different versions. Ontology-merging techniques for identifying similarities between different ontologies may provide semi-automatic means for integration of different versions.

16.3.2 Multiple Semantics and Mediation

Multiple ontology languages with different semantics and multiple intersecting ontologies are expected to co-exist on the Semantic Web. As a result, every application that wants to use ontologies developed in different languages has to deal with a plethora of different query and inference languages. This requirement considerably increases the effort of building applications for the Semantic Web.

To remedy this situation, there exists a need for an inference system that provides uniform storage of processing, querying, and inference with ontologies expressed in languages with different semantics. Unfortunately SiLRI is not usable here - the semantics of F-logic are built into the language, and it is not possible to specify that a certain set of axioms should just be used for a particular set of data. Since logic programming and deductive databases provide an adequate basis for reasoning, the provision of a core language based on Horn logic is suggested, which is extended with the notion of *semantic spaces*. Semantic spaces are disjoint from one another and each semantic space contains its own set of axioms. As a result, each semantic space can in effect implement different semantics for data. For example, one semantic space may contain the axioms defining RDF Schema, another one contains the axioms of UML, and the third one the axioms of TopicMaps. Since all the axioms are available in the same system, one can query several different datasets in the same system using different semantics.

The main task for the language is to enable the specification of *mediation system*s. E.g., the semantics of an ontology language may not be expressible using Horn logic axioms. DAML+OIL is an example of a description-logic-based ontology language. While some parts of description logics can be captured using Horn logic, it is not possible for a description logic in total. An application that aims to integrate different ontology-based languages still has to use different inference systems, such as the Horn logic based languages and a description-logic reasoning engine.

A mediation component may help here. The idea is to associate a semantic space not with a set of horn logic axioms, but with an arbitrary software component. In the case of DAML+OIL, a description-logic reasoner plugged into a semantic space of the Horn logic system may provide the reasoning support for a DAML+OIL ontology.

Reasoning support for ontology languages may be only one application of such a mediation component. Other possible applications are the integration of arbitary data sources and software components, similar to the MSL system in TSIMMIS [Papakonstantinou et al., 1996].

16.4 Final Remarks

An ancient Chinese curse says, '*May you live in interesting times*,' and it is true that interesting times are usually turbulent, difficult times. The times we are living in are not only interesting, but also exciting: we are merely in year 10 of the Big-Bang, which created the World Wide Web universe, a global information space previously unknown to humankind. The particles have just begun to cool down and to built more sophisticated structures, with the hope that the emerging universe will be rich and sophisticated.

As an example of what to expect, the following scenario is posited; so far we simply store documents and static data on the Web - aimed at human consumption. With the emergence of Web Services we will also store declarative, machine processable descriptions of how those Web Services may be combined to achieve more sophisticated tasks (first examples of description languages are emerging, e.g., IBM's Web Services Flow Language (WSFL)[1]). Web Services are connected to real life tasks, and these descriptions contain the knowledge of how more sophisticated tasks may be performed. These descriptions will be stored on the Web and are downloadable, understandable and executable by everyone, not only by humans but also by automated agents.

Automated agents will be able to extend their own capabilities with these downloadable descriptions and, given that every imaginable task may be represented as a declarative description and stored on the Web (just as a large portion of the world's knowledge is available on the Web now), an automated agent may be able to extend its capabilities beyond all imagination.

If an automated agent is indeed able to perform every imaginable task, it might be necessary again to discuss and revise the notion of machine intelligence.

1. See http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

References

[AAMODT & NYGÅRD, 1995]

A. Aamodt and M. Nygård (1995). Different roles and mutual dependencies of data, information and knowledge. *Data & Knowledge Engineering, 16*, 191-222.

[ABECKER ET AL., 1997]

A. Abecker, S. Decker, K. Hinkelmann, and U. Reimer (1997). *Workshop on Knowledge-Based Systems for Knowledge Management in Enterprises*, Freiburg, Germany, Document D-97-03, DFKI GmbH.

[ABECKER ET AL., 1998A]

A. Abecker, S. Decker, N. Matta, F. Maurer and U. Reimer: *Workshop on Building, Maintaining, and Using Organizational Memories*. In Conjunction with the 13th European Conference on Artificial Intelligence (ECAI-98). Brighton, UK, 1998. Available as CEUR Workshop No. 14. Retrieved from :http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-14/

[ABECKER, DECKER & KÜHN, 1998B]

A. Abecker, S. Decker and O. Kühn (1998). Organizational Memory. In: *Informatik Spektrum*, *21*(4), 213-214.

[ABECKER ET AL., 1998C]

A. Abecker, A. Bernardi, K. Hinkelmann, O. Kühn and M. Sintek (1998). Toward a Technology for Organizational Memories. I*EEE Intelligent Systems*, May/June, pp. 40-48.

[ABECKER ET AL., 1998D]

A. Abecker, S. Aitken, F. Schmalhofer, and B. Tschaitschian (1998). KARATEKIT: Tools for the Knowledge-Creating Company. In: B. R. Gaines and M. A. Musen (eds.), *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, University of Calgary, Banff, Canada.

[ABECKER & DECKER, 1999]

A. Abecker and S. Decker: Organizational Memory (1999). Knowledge Acquisition, Integration and Retrieval Issues. In: F. Puppe, (ed.), *Knowledge-based Systems: Survey and Future Directions, Proc. 5th German Conference on Knowledge-based Systems*, Würzburg, Lecture Notes in AI, Springer Verlag, 1999.

[ABITEBOUL ET AL., 2000]

S. Abiteboul, P. Buneman, D. Suciu (2000). Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufman.

[ABITEBOUL & VERCOUSTRE, 1999]

S. Abiteboul, A.M. Vercoustre (eds.) (2000). Research and Advanced Technology for Digital Libraries. In: *Proceedings of the Third European Conference on Digital Libraries, ECDL-99*, Paris, France, September 1999. LNCS 1696.

[ABITEBOUL ET AL., 1997A]

S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener (1997). The Lorel query language for semistructured data. *International Journal on Digital Libraries*, *1*(1), 68-88.

[ABITEBOUL ET AL., 1997B]

S. Abiteboul, S. Cluet and T. Milo (1997). Correspondence and Translation for Heterogeneous Data. In: *Proceedings of the International Conference on Database Theory (ICDT)*, pp.351-363, Delphy, Greece.

[ABITEBOUL, 1997]

S. Abiteboul (1997). Querying semi-structured data. In: *Proceedings of the International Conference on Database Theory* (ICDT), Delphy, Greece.

[ABITEBOUL & VIANU, 2000]

S. Abiteboul, V. Vianu: Queries and computation on the web. *Theoretical Computer Science, 239*(2), 231-255.

[ACKERMAN & MALONE, 1990]

M. S. Ackerman, T. W. Malone (1990). Answer Garden: A Tool for Growing Organizational Memory. In A*CM Conference on Office Information Systems* (Cambridge, MA), pp. 31-39.

[ACKERMAN & MCDONALD, 1996]

M. S. Ackerman, D. W. McDonald (1996). Answer Garden 2: Merging Organizational Memory with Collaborative Help. In: *Proceedings of the ACM Conference on Computer-Supported Cooperative Work* (CSCW'96), pp. 97-105.

[ALBRECHT, 1993]

F. Albrecht (1993): Strategisches Management der Unternehmensressource Wissen. In: *Verlag Peter Lang*, Frankfurt am Main, MA.

[ALEMBIC, 1999]

Retrieved from: http://www.mitre.org/technology/alembic-workbench/

[ALLEN, 2000]

Rob Allen (2000): Workflow: An Introduction. In: Layna Fischer (eds.)*: The Workflow Handbook 2001. Workflow Management Coalition (WfMC)*. ISBN 0-9703509-0-2

[AMANN & FUNDULAKI, 1999]

B. Amann and I. Fundulaki (1999). Integrating Ontologies and Thesauri to build RDF Schemas. In: *ECDL-99: Research and Advanced Technologies for Digital Libraries*, Lecture Notes in Computer Science, pp.234-253, Springer-Verlag, Paris.

[ALTHOFF ET AL., 1998]

K.D. Althoff, F. Bomarius and C. Tautz (1998). Using Case-Based Reasoning Technology to Build Learning Software Organizations. In: *[Abecker et al., 1998a]*.

[ANDREOLI ET AL., 1998]

J.M. Andreoli, C. Fernstrom, N. Glance and A. Grasso (1998). The Coordination Technology Area at XRCE Grenoble: The Research in Support of Distributed Cooperative Work, Xerox Research Centre Europe. *ACM SIGGROUP Bulletin,* April.

[ANGELE, 1993]

J. Angele (1993). *Operationalisierung des Models der Expertise mit KARL,* Infix, St. Augustin,.

[ANGELE ET AL., 1996]

J. Angele, S. Decker, R. Perkuhn, and R. Studer (1996). Modeling Problem-Solving Methods in New KARL. In: *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop* (KAW´98), Banff, Canada. Retrieved from: http://ksi.cpsc.ucalgary.ca/KAW/KAW96/angele/angele.html

[ANGELE ET AL., 1998]

J. Angele, D. Fensel, and R. Studer (1998). Developing Knowledge-Based Systems with MIKE, *Journal of Automated Software Engineering*, *5*(4), 389-418.

[ANGUS ET AL., 1998]

J. Angus, J. Patel, J. Harty (1998). Knowledge management: Great concept ... but what is it? In: *Information Week.*

[ARENS ET AL., 1993]

Y. Arens, C. Y. Chee, C.-N. Hsu and C. Knoblock (1993). Retrieving and Integrating Data From Multiple Information Sources. *International Journal of Intelligent Cooperative Information Systems, 2*(2):127−158.

[ARKIN & AGRAWAL, 2001]

Assaf Arkin and Ashish Agrawa (2001). Business Process Modeling Language (BPML).Working Draft 0.4. Retrieved from: http://www.bpmi.org/

[AROCENA & MENDELZON, 1998]

G. Arocena and G. Mendelzon: WebOQL (1998). Restructuring Documents, Databases and Webs. In: *Proceedings of the International Conference on Data Engineering (ICDE-98)*, Orlando, Florida.

[ASHISH & KNOBLOCK, 1997]

N. Ashish and C. Knoblock (1997). Semi-automatic Wrapper Generation for Internet Information Sources. In: *Proceedings of the Conference on Cooperative Information Systems (CoopIS),* Charlston, South Carolina.

[BAADER ET AL., 1991]

F. Baader, H.J. Heinsohn, B. Hollunder, J. Muller, B. Nebel, W. Nutt, and H.J. Profitlich: (1991). Terminological knowledge representation: A proposal for a terminological logic.In: *Technical Memo* TM-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI).

[BAADER, 1999]

F. Baader (1999). Logic-based knowledge representation. In: M.J. Wooldridge and M. Veloso, (eds.), *Artificial Intelligence Today*, Recent Trends and Developments, no.1600 in Lecture Notes in Computer Science, pp.13-41. Springer Verlag.

[BANNON & HUGHES, 1993]

L. J. Bannon, J. A. Hughes (1993). The Context of CSCW. In: K. Schmidt (ed.) *Report of COST14 "CoTech" Working Group 4* (1991-1992), pp.9-36. ISBN 87-550-1924-2

[BARAL & GELFOND, 1994]

C. Baral and Michael Gelfond (1994). Logic Programming and Knowledge Representation. *Journal of Logic Programming*, *19/20*, 73-148.

[BASILI ET AL., 1994]

V.R. Basili, G. Caldiera, and H.-D. Rombach (1994). Experience Factory. In: J. Marciniak (Ed.)*: Encyclopedia of Software Engineering, vol. 1*, John Wiley & Sons, pp.469-476.

[BASILI & ROMBACH., 1998]

V. R. Basili and H. D. Rombach (1998). The TAME Project: Towards Improvement-Oriented Software Environments. In: *IEEE Trans. on Software Engineering*, *14*(6), **(pages?)**.

[BAUER ET AL., 1994]

B.M. Bauer, C. Kohl, H.C. Mayr, and J. Wassermann (1994). Enterprise Modeling using OOA Techniques. In: *Proceedings Connectivity '94: Workflow Management - Challenges, Paradigms and Products*. R. Oldenbourg-Verlag, Munich, pp. 96-111.

[BAUMANN ET AL., 1997]

S. Baumann, M. Malburg, H. Meyer auf'm Hofe, and C. Wenzel (1997). From Paper to a Corporate Memory-A First Step. In: *[Abecker et al., 1997]*.

[BENJAMINS, 1995]

V. R. Benjamins (1995). Problem Solving Methods for Diagnosis And Their Role in Knowledge Acquisition. *International Journal of Expert Systems: Research and Application, 8*(2):3-120.

[BENJAMINS, 1997]

R. Benjamins: Problem-Solving Methods in Cyberspace (1997). In: *Proceedings of the Workshop on Problem-Solving Methods for Knowledge-based Systems (W26)* during the *Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97),* Nagoya, Japan, August 23-29.

[BENJAMINS & FENSEL, 1998A]

V. R. Benjamins and D. Fensel (1998). Special Issue on Problem-Solving Methods. In: *International Journal of Human-Computer Studies (IJHCS),.49*(4):305-313.

[BENJAMINS & FENSEL, 1998B]

V. R. Benjamins and D. Fensel (1998). The ontological engineering initiative (KA). In Guarino, N., (Ed.), *Formal Ontology in Information Systems (FOIS'98)*, IOS Press, pp.287–301.

[BENJAMINS ET AL., 1998A]

V. R. Benjamins, D. Fensel, A. Gomez-Perez, S. Decker, Michael Erdmann, E. Motta, and M. Musen (1998). Knowledge Annotation Initiative of the Knowledge Acquisition Community $(KA)^2$. In *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW´98)*, Banff, Canada, April 18-23.

[BENJAMINS ET AL., 1998B]

V. R. Benjamins, E. Plaza, E. Motta, D. Fensel, R. Studer, B. Wielinga, G. Schreiber, Z. Zdrahal (1998). An Intelligent Brokering Service for Knowledge-Component Reuse on the World-WideWeb. In: *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW´98)*, Banff, Canada, April 18-23.

[BENJAMINS ET AL., 1998C]

V. R. Benjamins, D. Fensel, and A. Gómez Pérez, Knowledge Management through Ontologies (1998). In: *Proceedings of the 2nd International Conference on Practical Aspects of Knowledge Management (PAKM '98)*, Basel, Switzerland, October..

[BENJAMINS & Fensel & Decker & Gomez-Perez, 1999]

V R. Benjamins, D. Fensel, S. Decker, and A. Gomez-Perez: (KA)2 (1999). Building Ontologies for the Internet: a Mid Term Report. *International Journal of Human-Computer Studies (IJHCS), 51,* 687-612.

[BENTLEY ET AL., 1997]

R. Bentley, W. Appelt, U. Busbach, E. Hinrichs, D. Kerr, S. Sikkel, J. Trevor, and G. Woetzel (1997). Basic Support for Cooperative Work on the World Wide Web. *International Journal of Human-Computer Studies: Special issue on Innovative Applications of the World Wide Web, 46*(6), 827-846, June.

[BERNERS-LEE, 1999]

Tim Berners-Lee (1999). *Weaving the Web: The Orginal Design and Ultimate Destiny of the World Wide Web by Its Inventor.* Harper San Francisco.

[BERNERS-LEE ET AL., 1998]

T. Berners-Lee, R. Fielding, L. Masinter (1998). Uniform Resource Identifiers (URI): Generic Syntax. *IETF Request for Comments: 2396*. Retrieved from http://www.ietf.org/rfc/rfc2396.txt

[BIRON & MALHOTRA, 2000]

P. V. Biron, A. Malhotra (Eds.) (2000). XML Schema Part 2: Datatypes, W3C Working Draft 07 April. Retrieved from: http://www.w3.org/TR/2000/WD-xmlschema-2-20000407/

[BLEICHER, 1992]

K. Bleicher (1992). Das Konzept Integriertes Management, 2. Auflage. Campus Verlag, Frankfurt am Main.

[BELCHER & PLACE, 2000]

M. Belcher, E. Place (1999). People Power and the Semantic Web: Building Quality Controlled Portals. In: *WWW9 Poster Session,* Amsterdam. Retrieved from http://www9.org/final-posters/poster64.html

[BÖNING, 1994]

U. Böning (1994). Moderieren mit System, 2. Auflage.Gabler, Wiesbaden.

[BOLL ET AL., 1998]

S. Boll, W. Klas and A. Sheth (1998). Overview on Using Metadata to Manage Multimedia Data. In: [Sheth & Klas, 1998] pp. 1-24.

[BONNER & KIFER, 1993]

A. J. Bonner and M. Kife (1993). Transaction Logic Programming. In: *Proceedings of the 10th International Conference on Logic Programming (ICLP),* Budapest, Hungary, June 21-24.

[BONNER & KIFER, 1995]

A. J. Bonner and M. Kifer (1995). *Transaction Logic Programming,* Technical Report CSRI-323.

[BORGIDA, 1996]

A. Borgida (1996). On the Relative Expressiveness of Description Logics and Predicate Logics, Artificial Intelligence Journal, *82*(1-2), 353-367.

[BORGIDA & PATEL-SCHNEIDER, 1994]

A. Borgida and P. F. Patel-Schneider (1994). A semantics and complete algorithm for subsumption in the CLASSIC description logic. J. of Artificial Intelligence Research, *1*, 277–308.

[BORGHOFF & PARESCHI, 1998A]

U. M. Borghoff and R. Pareschi (Eds) (1998): *Information Technology for Knowledge Management*, Berlin, Heidelberg, New York, Springer.

[BORGHOFF & PARESCHI, 1998B]

U. M. Borghoff and R. Pareschi. Introduction. In:[Borghoff & Pareschi, 1998a] .

[BORST & AKKERMANS, 1997]

W.N. Borst, J.M. Akkermans (1997). Engineering Ontologies. *International Journal of Human-Computer Studies*, *46*(2/3), 365-406.

[BOSAK, 1999]

Jon Bosak (1999). Web Applications of XML. In: XML, Java and the future of the Web, Retrieved from http://metalab.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm

[BOSAK & BRAY, 1999]

Jon Bosak and Tim Bray (1999). XML and the Second-Generation Web. *Scientific American*, May. Retrieved from http://www.sciam.com/1999/0599issue/0599bosak.html.

[BOX ET AL., 2000]

D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk-Nielsen, S. Thatte, D. Winer (2000). Simple Object Access Protocol (SOAP) 1.1, *W3C Note 08 May 2000*, Retrieved from http://www.w3.org/TR/2000/NOTE-SOAP-20000508

[BRACHMAN, 1979]

R. J Brachman (1979). On the Epistemological Status of Semantic Networks. *Associative Networks: Representations and Use of Knowledge by Computers*, Findler, N.V: Academic Press, pp. 3-50.

[BRACHMAN ET AL., 1991]

R. J. Brachman, A. Borgida, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Resnick (1991). Living with CLASSIC: When and how to use a KL-ONE-like language.John Sowa (Ed.) *Principles of Semantic Networks*, San Mateo, CA, Morgan Kaufmann, pp. 401-56.

[BRAY ET AL., 1998]

T. Bray, J. Paoli, and C.M. Sperberg-McQueen (1998). "Extensible Markup Language (XML) 1.0, *W3C Recommendation*. Retrieved from http://www.w3.org/TR/REC-xml

[BRAY ET AL., 1999]

T. Bray, D. Hollander, and Andrew Layman (Eds.) (1999). Namespaces in XML, *World Wide Web Consortium Recommendation*, 14-January. Retrieved from http://www.w3.org/TR/REC-xml-names/

[BRACHMAN & SCHMOLZE, 1985]

R. Brachman and J. Schmolze (1985). An overview of the KL-ONE Knowlege Representation System. *Cognitive Science*, *9*:171-216.

[Breuker & Van de Velde, 1994]

J. Breuker and W. Van de Velde (Eds.) (1994). *The CommonKADS Library for Expertise Modeling*, Amsterdam, IOS Press.

[Brewka & Dix, 1999]

G. Brewka and J. Dix, Knowledge representation with logic programs (1999). *D. M. Gabbay ed., Handbook of Philosophical Logic, 2nd edition, vol. 6, Chapter 6*, Oxford University Press.

[Brewka & Dix, 1998]

G. Brewka and J. Dix (1998). Knowledge representation with logic programs. J. Dix, L. Pereira, and T. Przymusinski, (Eds.) *Logic Programming and Knowledge Representation*, LNAI 1471, pp. 1-55, Berlin, Springer.

[Brickley & Guha, 2000]

D. Brickley, R. Guha (2000_. Resource Description Framework (RDF) Schema Specification 1.0, *W3C Candidate Recommendation 27 March 2000*. Retrieved from http://www.w3.org/TR/PR-rdf-schema/

[Brin & Page, 1998]

S. Brin, L. Page (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. *WWW7 / Computer Networks 30*(1-7), 107-117.

[Bringsjord, 1998]

S. Bringsjord (1998). Logic and Artificial Intelligence: Still Married, Divorced, Separated...?: *Minds and Machines*.Minds *8*, 273-308. Also in: http://www.rpi.edu/~brings/LOG+AI/lai/lai.html

[Brodie, 1984]

M.L. Brodie (1984). On the Development of Data Models. M.L. Brodie et al. (Eds.), *On Conceptual Modeling*, Berlin, Springer-Verlag.

[Buckingham Shum, 1998]

S. Buckingham Shum (1998). Negotiating the Construction of Organisational Memories. In: *[Borghoff & Pareschi, 1998a]*, pp. 55-78.

[Bullinger et al., 1997]

H.-J Bullinger, K. Wörner, and J. Prieto (1997). Wissensmanagement heute. Fraunhofer Institut für Arbeitswirtschaft und Organisation, Stuttgart.

[Catarci et al., 1998]

T. Catarci, L. Iocchi, D. Nardi, and G. Santucci (1998). Accessing the Web: exploiting the DB paradigm. In: *[Abecker et al., 1998a]*.

[Carey et al., 1996]

M. J. Carey, L. M. Haas, V. Maganty, and J. H. Williams (1996). PESTO : An Integrated Query/Browser for Object Databases. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases*, September 3-6, Mumbai (Bombay), India, pp 203-214. Morgan Kaufmann.

[Chen et al., 1993]

W. Chen, M. Kifer, and D.S. Warren (1993). Hilog: a foundation for higher-order logic programming. *Journal of Logic Programming*, *15*,187-230, February.

[CLANCEY, 1983]

W. J. Clancey: Heuristic Classification (1985). *Artificial Intelligence*, *27*, 289—350.

[CLANCEY, 1993]

W. J. Clancey (1993). The Knowledge Level Reinterpreted: Modeling Socio-Technical Systems. *The International Journal of Intelligent Systems, 8*(2).

[CLANCEY ET AL., 1996]

W. J. Clancey, P. Sachs, M. Sierhuis, R. van Hoff: Brahms (1996). Simulating Practice for Work Systems Design, In: *Proceedings of The Pacific Knowledge Acquisition Workshop (PKAW)*.

[CONKLIN & WEIL, 1997]

E.J. Conklin and W. Weil (1997). Wicked Problems: Naming the Pain in Organizations. Retrieved from: *http://www.gdss.com/wicked.htm*

[COVEY, 1989]

R. Covey (1989). The seven habits of highly effective people. New York, Simon & Schuster, Inc..

[CRUZ, 1992]

I. F. Cruz (1992). DOODLE: A Visual Language for Object-Oriented Databases. In: P*roceedings of the 1992 ACM SIGMOD International Conference on Management of Data*, San Diego, California, June 2-5.

[DAGASAN & WITT, 1998]

A. Dagasan, A. Witt (1998). Realisierung eines Wrappers zur Erfassung der Fakten des CIA World Fact Books. *Studienarbeit 767/768*. Institut AIFB, Universität Karlsruhe.

[DAHLGREN, 1995]

K. Dahlgren (1995). A linguistic ontology. *International Journal of Human-Computer Studies*, *43* (5/6), 809-818.

[DALITZ ET AL., 1997]

W. Dalitz, M. Grötschel and J. Lügger (1997). Information Services for Mathematics in the Internet (Math-Net). In: *Proceedings of the 15th IMACS World Congress on Scientific Computation: Modelling and Applied Mathematics*. A. Sydow (Ed.) Wissenschaft und Technik Verlag, *Artificial Intelligence and Computer Science*, *4*, 773-778/

[DAML JOINT COMMITTEE, 2001]

DAML+OIL specification, March 27 2001. Retrieved from http://www.daml.org/2001/03/daml+oil-index

[DANIEL & DECKER ET AL., 1997]

M. Daniel, S. Decker et al (1997). ERBUS-Towards a Knowledge Management System for Designers. In *[Abecker et al., 1997]*.

[DAVENPORT, 1996]

\T. H Davenport (1996). Some Principles of Knowledge Management. Retrieved from http://www.bus.utexas.edu/kman, *Graduate School of Business*, University of Texas at Austin, *Strategy and Business*, February 1996, pp. 34-40.

[DAVENPORT ET AL., 1996]

T.H. Davenport, S.L. Jarvenpaa, and M.C. Beers (1996). Improving Knowledge Work Processes. *Sloan Management Review*, *37*(4), 53-65.

[DAVENPORT & PRUSAK, 1998]

T. H. Davenport and L. Prusak (1998). *Working Knowledge. How Organizations manage what they know.* McGraw-Hill; Harvard Business School Press.

[DAVENPORT, 1997]

Tom Davenport (1997). If only HP knew what HP knows. *Perspectives on Business Innovation Journal*. Issue 1. Retrieved from http://www.cbi.cgey.com/journal/issue1/features/ifonly/

[DAVIS, 1993]

A.M. Davis (1993). *Software requirements: objects, functions, and states*, Prentice Hall.

[DAY ET AL., 1997]

D. Day, J. Aberdeen, L. Hirschman, R. Kozierok, P. Robinson, and M. Vilain (1997). Mixed-Initiative Development of Language Processing Systems. In: *Fifth Conference on Applied Natural Language Processing,1997, Association for Computational Linguistics*, 31 March - 3 April, Washington D.C. .

[DECKER, 1994]

S. Decker (1994): Ueberpruefung von Integritaetsbedingungen in Deduktiven Datenbanken. Diploma Thesis, University of Kaisersautern.

[DECKER, 1998]

S. Decker (1998). On Domain-Specific Declarative Knowledge Representation and Database Languages In: (Eds.) A. Borgida, V. Chaudri, M. Staudt. *Proceedings of the 5th KRDB Workshop (KRDB98)*, Seattle, WA, 31 May.

[DECKER ET AL., 1999]

S. Decker, M. Erdmann, D. Fensel and R. Studer (1999). Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In: R. Meersman et al. (Eds.), *DS-8 Semantic Issues in Multimedia Systems*, Kluwer Academic Publisher.

[DECKER ET AL., 1996]

S. Decker, M. Erdmann, R. Studer (1996). A Unifying View on Business Process Modelling and Knowledge Engineering. In: *Proceedings of the 10th Knowledge Aquisition Workshop (KAW 96)*, Banff, Canada, November.

[DECKER & STUDER, 1998]

S. Decker and R. Studer (1998). Towards an Enterprise Reference Scheme for Building Knowledge Management Systems. In: *Proc. Modellierung 98*, Münster.

[DECKER ET AL., 1997]

S. Decker, M. Daniel, M. Erdmann, and R. Studer (1997). An Enterprise Reference Scheme for Integrating Model-based Knowledge Engineering and Enterprise Modeling. In: E. Plaza and R. Benjamins, (Eds.), *Knowledge Acquisition, Modeling, and Management, 10th European Workshop (EKAW'97)*, Sant Feliu de Guixols, Lecture Notes in Artificial Intelligence 1319, Springer-Verlag.

[DECKER ET AL., 1998]

S. Decker, M. Erdmann, D. Fensel, R. Studer (1998). How to Use Ontobroker. In *Proceedings of the 11th Workshop on Knowledge Aquisition, Modeling and Management (KAW '98)*, Banff, Canada, November.

[DECKER ET AL., 2000A]

S. Decker, J. Jannink, S. Melnik, P. Mitra, S. Staab, R. Studer and G. Wiederhold (2000). An Information Food Chain for Advanced Applications on the WWW. In: *Proceedings of the Fourth European Conference on Research and Advanced Technology for Digital Libraries (ECDL '2000)*, Lisbon, Springer, LNCS.

[DECKER ET AL., 2000B]

S. Decker, P. Mitra, S. Melnik (2000) Framework for the Semantic Web: An RDF Tutorial In: *IEEE Internet Computing*. November/December.

[DECKER ET AL., 2000C]

S. Decker, F. van Harmelen, J. Broekstra, M. Erdmann, D. Fensel, I. Horrocks, M. Klein, S. Melnik (2000). The Semantic Web - on the Roles of XML and RDF . In: *IEEE Internet Computing*. September/October.

[DECKER ET AL., 2000D]

S. Decker, D. Fensel, F. van Harmelen, I. Horrocks, S. Melnik, M. Klein, and J. Broekstra (2000). Knowledge Representation on the Web. In: *Proceedings of the 2000 International Workshop on Description Logics (DL2000)*.

[DE HOOG ET AL., 1994]

R. de Hoog, R. Martil, B. Wielinga, R. Taylor, C. Bright, and W. van de Velde (1994). The CommonKADS model set. *ESPRIT Project P5248 KADS-II/M1/DM.1b/UvA/018/6.0*, University of Amsterdam, Lloyd's Register, Touche Ross Management Consultants & Free University of Brussels, June.

[DE HOOG ET AL., 1994B]

R. de Hoog, B. Benus, C. Metselaar, M. Vogler, and W. Menezes (1994). Organisation model: Model definition document. Deliverable DM6.2c, *ESPRIT Project P5248 KADS-II/M6/M/UvA/ 041/3.0, University of Amsterdam and Cap Programmator*, June.

[DE HOOG ET AL., 1996]

R. de Hoog, B. Benus, M. Vogler, C. Metselaar (1996). The CommonKADS Organisation Model: Content, Usage and Computer Support. *Journal of Expert Systems with Applications*, *11* (1), 29-40.

[DENGEL & HINKELMANN, 1996]

A. Dengel and K. Hinkelmann (1996). The Specialist Board - A Technology Workbench for Document Analysis and Understanding. In: *Integrated Design & Process Technology, Proceedings of the Second World Congress*, Society for Design and Process Science.

[DIENG ET AL., 1998]

R. Dieng, O. Corby, A. Giboin, and M. Ribière (1998). Methods and Tools for Corporate Knowledge Management. In: *Proceedings of the 12th Workshop on Knowledge Aquisition, Modeling and Management*, Banff, Canada.

[DOMINGUE 1998]

J. Domingue (1998). Tadzebao and webonto: Discussing, browsing, and editing ontologies on the web. In: *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management, (KAW'98)*, Banff, Canada.

[DRUCKER, 1993]

P. F. Drucker (1993). Die postkapitalistische Gesellschaft. *Econ Verlag*, Düsseldorf.

[Drucker, 1988]

P. F. Drucker (1988). The Coming of the New Organization. In: *Harvard Business Review*, January-February, pp. 45-53.

[Duursma et al., 1993]

C. Duursma, O. Olsson, and U. Sundin (1993). Task model definition and task analysis process. *Deliverable ESPRIT Project P5248 KADS-II/M5/VUB/RR/004/2.0*, Free University Brussels, December.

[Ebbinghaus, 1994]

H.-D. Ebbinghaus, J. Flum, and W. Thomas (1994). *Mathematical Logic. Undergraduate Texts in Mathematics*. Springer-Verlag, 2nd edition.

[Enderton, 1992]

H.B. Enderton (1992). A mathematical introduction to logic. *Academic Press*, San Diego, California.

[Eppler, 1997]

M. J. Eppler (1997). Knowledge Mapping. Retrieved from *http://www.cck.uni-kl.de/wmk/papers/public/KnowledgeMapping* Universität Kaiserslautern, HEC Universität Genf.

[Erdmann et al., 2000]

M. Erdmann, A. Mädche, S. Staab, H.-P. Schnurr (2000). From Manual to Semi-Automatic Semantic Annotation: About Ontology-based Text Annotation Tools. in: *Proceedings of Workshop on Semantic Annotation and Intelligent Content* at the 18th International Conference on Computational Linguistics (COLING 2000), Luxembourg, August.

[Erdmann, 2001]

M. Erdmann (2001). Ontologien zur konzeptuellen Modellierung der Semantik von XML. *Dissertation*, Institut AIFB, University of Karlsruhe.

[Erdmann & Studer, 2001]

M. Erdmann, R.Studer (2001). How to Structure and Access XML Documents With Ontologies. In: *Data and Knowledge Engineering*, Special Issue on Intelligent Information Integration.

[Eriksson et al., 1995]

H. Eriksson, Y. Shahar, S.W. Tu, A.R. Puerta, and M.A. Musen (1995). Task Modeling with Reusable Problem-Solving Methods. *Artificial Intelligence*, *79*(2), 293-326.

[Eriksson et al., 1999]

H. Eriksson, R. Fergerson, Y. Shahar, and M. A. Musen (1999). Automatic Generation of Ontology Editors. In: *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management* (KAW '99), Banff, Canada.

[Essers & Schreinemakers, 1996]

J. Essers and J. Schreinemakers (1996). Critical Notes on the Use of Knowledge in Knowledge Management. In: R. Green (Ed.), *Knowledge Organization and Change: Advances in Knowledge Organization*. Proceedings of the Fourth International ISKO Conference 15-18 July, Washinghton DC, INDEKS Verlag, pp. 213-215.

[Etzioni, 1997]

O. Etzioni (1997). Moving Up the Information Food Chain: Deploying Softbots on the World Wide Web, *AI Magazine*, *18*(2), 11-18.

[EUZENAT, 1996]

J. Euzenat (1996). Corporate Memory through Cooperative Creation of Knowledge Bases and Hyper-documents. In: *Proceedings of the 10th Workshop on Knowledge Acquisition, Modeling and Management (KAW'96)*, Banff, Canada.

[EVANS, 1994]

J. D. G. Evans (1994). Meno's Puzzle. In: Ioanna Kuaçuradi and R. S. Cohen (Eds.): *The concept of knowledge: the Ankara seminar*, Boston studies in the philosophy of science, v. 170, Kluwer Academic Publisher.

[EVETT ET AL., 1993]

M. Evett, W. Andersen, and J. Hendler (1993). Providing computational effective knowledge representation via massive parallelism. In: *L. Kanal, V. Kumar, H. Kitano, and C. Suttner, editors, Parallel Processing for Artificial Intelligence*, Amsterdam, Elsevier Science.

[FARQUHAR ET AL., 1997]

A. Farquhar, R. Fikes, and J. Rice (1997). The Ontolingua Server: a Tool for Collaborative Ontology Construction. In: *International Journal of Human-Computer Studies (IJHCS)*, *46*(6), 707-728.

[FEGARAS, 1999]

L. Fegaras (1999). VOODOO: a visual object-oriented database language for ODMG OQL. In *Proceedings of the ECOOP Workshop on Object-Oriented Databases*, Lisbon, Portugal, June.

[FELLBAUM, 1998]

Christiane Fellbaum (Ed.) (1998). Wordnet: An Electronic Lexical Database; MIT Press, Also at http://www.cogsci.princeton.edu/~wn, 1998.

[FENSEL ET AL., 1998A]

D. Fensel, S. Decker, M. Erdmann and R. Studer (1998). Ontobroker: The Very High Idea. In: *Proceedings of the 11th International Florida AI Research Symposium (FLAIRS '98)*, Sanibal Island, Florida.

[FENSEL, 1995]

D. Fensel (1995). Formal Specification Languages in Knowledge and Software Engineering. *The Knowledge Engineering Review, 10* (4).

[FENSEL ET AL., 1998B]

D. Fensel, S. Decker, M. Erdmann, and R. Studer: Ontobroker: Or How to Enable Intelligent Access to the WWW. In: *Proceedings of the 11th Workshop on Knowledge Acquisition*, Modeling and management, Banff, 1998

[FENSEL ET AL., 1998C]

D. Fensel, J. Angele, and R. Studer (1998). The Knowledge Acquisition and Representation Language KARL. *IEEE Transactions on Knowledge and Data Engineering 10*(4), 527-550.

[FENSEL ET AL., 1998D]

D. Fensel, S. Decker, M. Erdmann, and R. Studer (1998). Ontobroker: Transforming the WWW into a Knowledge Base. In *Proceedings of the 11th Workshop on Knowledge Acquisition Modeling and Management (KAW´98)*, Banff, Canada, April 18-23.

[FENSEL, 2000]

D. Fensel (2000). Problem-Solving Methods. *Lecture Notes in Computer Science (LNAI)* .VOL. 1791, Springer Verlag.

[FENSEL ET AL., 1997]

D. Fensel, E. Motta, S. Decker, and Z. Zdrahal (1997). Using Ontologies For Defining Tasks, Problem-Solving Methods and Their Mappings. In: E. Plaza et al. (Eds.), *Knowledge Acquisition, Modeling and Management*, Lecture Notes in Artificial Intelligence (LNAI) 1319, Springer-Verlag.

[FENSEL ET AL., 2000]

D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, and M. Klein (2000): OIL in a Nutshel. 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*, Springer-Verlag, October.

[FERNÁNDEZ ET AL., 1998]

M. Fernández, D. Florescu, J. Kang, A. Levy (1998). Catching the Boat with Strudel: Experiences with a Web-Site Management System. In: *Proceedings of the 1998 ACM International Conference on Management of Data* (SIGMOD'98), Seattle, WA.

[FERNANDEZ & GOMEZ-PEREZ, 1999A]

M. Fernandez, A. Gomez-Perez, J. Pazos, and A. Pazos (1999). Ontology of tasks and methods. *IEEE Intelligent Systems and Their Applications*, *14*(1), 37-46.

[FERNÁNDEZ & GOMEZ-PEREZ, 1999B]

M. Fernández, A. Gomez-Perez, A. Pazos, J. Pazos (1999). Building a Chemical Ontology using METHONTOLOGY and the Ontology Design Environment *IEEE Expert: Special Issue on Uses of Ontologies*. January/February pp. 37-46.

[FIELDING ET AL., 1999]

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee (1999).: Hypertext Transfer Protocol -- HTTP/1.1. *IETF Request for Comments: 2616.* June. Retrieved from http://www.ietf.org/rfc/rfc2616.txt

[FIKES & KEHLER, 1985]

R. Fikes and T. Kehler (1985). The role of frame-based representation in reasoning. In: *Communications of the ACM*, *28* (9).

[FIKES ET AL., 1996]

R. Fikes, A. Farquhar, and W. Pratt: R. Fikes, A. Farquhar, W. Pratt (1996). Information Brokers for Gathering Information from Heterogeneous Information Sources; *Proceedings of the Ninth Florida Artificial Intelligence Research Symposium (FLAIRS '96)*, John H. Stewman [Ed.], Key West, Florida, May, pp. 192-197. Also, KSL Technical Report KSL-96-18 .

[FIESER & DOWDEN, 1999]

J. Fieser and B. Dowden (Eds.) (1999). The Internet Encyclopedia Of Philosophy. Retrieved from: *http://www.utm.edu/research/iep/*

[FLORESCU ET AL., 1998]

D. Florescu, A. Levy, A. Mendelzon (1998). Database Techniques for the World-Wide Web. In: *SIGMOD Record*. *27*(3), 59-74.

[FOX ET AL., 1996]

M. S. Fox, M. Barbuceanu, and M. Gruninger (1996). An Organisation Ontology for Enterprise Modelling: Preliminary Concepts for Linking Structure and Behaviour, *Computers in Industry, 29*, 123-134.

[FRANCONI ET AL., 2000]

E. Franconi, F. Grandi and F. Mandreoli (2000). A Semantic approach for Schema Evolution and Versioning in Object-Oriented Databases. *6th International Conference on Rules and Objects in Databases (DOOD '00)*, London, UK, July.

[FREIRE ET AL., 1997]

J. Freire, P. Rao, K. Sagonas, T. Swift, and D. S. Warren (1997). XSB: A System for Efficiently Computing the Well-Founded Semantics. In: *International Workshop on Logic Programming and Non-Monotonic Reasoning.*

[FRIDMAN NOY & HAFNER, 1997]

N. Fridman Noy and C. D. Hafner (1997). The State of the Art in Ontology Design, *AI Magazine*, *18*(3), 53-74.

[FROHN ET AL., 1997]

J. Frohn, R. Himmeröder, P.-Th. Kandzia, G. Lausen, C. Schlepphorst (1997). FLORID - A Prototype for F-Logic. In: *Proceedings of the International. Conference on Data Engineering (ICDE, Exhibition Program)*, Birmingham. IEEE Computer Science Press. Also at http://www.informatik.uni-freiburg.de/~dbis/flogic-project.html.

[FRÖHLICH ET AL., 1998]

P. Fröhlich, W. Neijdl, and M. Wolpers (1998). KBS-Hyperbook - An Open Hyperbook System for Education. In: *10th World Conference on Educational Media and Hypermedia (EDMEDIA'98)*, Freiburg, Germany.

[GARCIA-MOLINA ET AL., 1995]

H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and Jennifer Widom (1995). Integrating and Accessing Heterogeneous Information Sources in TSIMMIS. In: *Proceedings of the AAAI Symposium on Information Gathering*, pp. 61-64, Stanford, California, March.

[GAMMA ET AL., 1995]

E. Gamma, R. Helm, R. Johnson, and J. Vlissides (1995). *Design Patterns*, Addison-Wesley Pub..

[GARLAN & PERRY, 1995]

D. Garlan and D. Perry (Eds.) (1995). *Special Issue on Software Architecture*, *IEEE Transactions on Software Engineering, 21*(4).

[GELDFOND & LIFSCHITZ, 1988]

M. Gelfond and V. Lifschitz (1998). The Stable Model Semantics for Logic Programming. In R.Kowaliski and K. Bowen (Eds.), *5th Conference on Logic Porgramming*, pp. 1070-1080, MIT-Press.

[GENE ONTOLOGY CONSORTIUM, 2001]

The Gene Ontology Consortium (2001). Creating the gene ontology resource: design and implementation. *Genome Research, 11*, 1425-1433.

[GENESERETH & KETCHPEL, 1994]

M.R. Genesereth, S. P. Ketchpel (1994) Software Agents. In: *Communications of the Association for Computing Machinery,* July, pp. 48-53.

[GENESERETH ET AL., 1997]

M. R. Genesereth, A. M. Keller, and O. M. Duschka (1997). Infomaster: An Information Integration System. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Tucson, AZ, May.

[GENESERETH AND NILSSON, 1987]

M. Genesereth and N.J. Nilsson: *Logical Foundations of Artificial Intelligence.* San Mateo, Morgan Kaufmann.

[GIANNOTTI ET AL., 1997]

F. Giannotti, G. Manco, D. Pedreschi (1997). A Deductive Data Model for Representing and Querying Semistructured Data. In: *Proceedings of the Second international Workshop on Logic Programming Tools for Internet Applications*, Leuven. Retrieved from http://www.clip.dia.fi.upm.es/lpnet/proceedings97/lpnet_proc97.html

[GLUSCHKO ET AL., 1999]

R.J. Gluschko, J.M. Tenenebaum, B. Meltzer (1999). An XML Framework for Agent-based E-commerce. *Communications of the ACM, 42*(3), 106-114.

[GOEDEL, 1965]

K. Goedel (1965). On formally undecidable propositions of the principia mathematica and related systems. In M. Davis, (Ed.), *The Undecidable*, pp. 5-38. Hewlett, N.Y., Raven Press.

[GOLDBLATT, 1982]

R. Goldblatt (1982). *Axiomatising the Logic of Computer Science*, Lecture Notes in Computer Science (LNCS) 130, Springer-Verlag, Berlin.

[GOLDMAN ET AL., 1996]

R. Goldman, S. Chawathe, A. Crespo, J. McHugh (1996). A Standard Textual Interchange Format for the Object Exchange Model (OEM). *Technical Report*, Stanford University. Retrieved from http://dbpubs.stanford.edu/pub/1996-5

[GOUBAULT, 1994]

J. Goubault. The complexity of resource-bounded first-order classical logic. In: P. Enjalbert, E.W. Mayr, and K.W. Wagner, (Eds.), *11th Symposium on Theoretical Aspects of Computer Science*, pp. 59-70, Caen, France, February. Springer Verlag LNCS 775.

[GROSSO ET AL., 1999]

W. E. Grosso, H. Eriksson, R. W. Fergerson, J. H. Gennari, S. W. Tu, & M. A. Musen (1999).. Knowledge Modeling at the Millennium (The Design and Evolution of Protege-2000). In: *Twelfth Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Alberta, Canada, October.

[GRUBER, 1993]

T. R. Gruber (1993). A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*, *5*, 199—220.

[GRUBER, 1995]

T. R. Gruber (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human-Computer Studies (IJHCS)*, *43*(5/6), 907—928.

[GRUNINGER & FOX, 1995]

M. Gruninger and M. Fox (1995). Methodology for the design and evaluation of ontologies. In: *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing help in conjunction with IJCAI'95*.

[GUARINO ET AL., 1999]

N. Guarino, C. Masolo, G. Vetere (1999). OntoSeek: Content-Based Access to the Web. In: *IEEE Intelligent Systems* 14(3), May/June, pp. 70-80. Retrieved from: http://www.ladseb.pd.cnr.it/infor/Ontology/Papers/OntoSeek.pdf

[GUARINO, 1998]

N. Guarino (Ed.) (1998). *Proceedings of the first International Conference on Formal Ontologies in Information Systems (FOIS-98)*, Trento, Italy. Frontiers in Artificial Intelligence and Applications, vol. 46, IOS-Press.

[GUARINO, 1997]

N. Guarino (1997). Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. In: M. T. Pazienza, (Ed.), *Summer School on Information Extraction*, Springer Verlag.

[GUELDENBERG, 1998]

S. Gueldenberg (1998). *Wissensmanagment und Wissenscontrolling in lernenden Organisationen.* Deutscher Universitaets Verlag, Weisebaden.

[GUHA, 1993]

R. V. Guha (1993). *Context Dependence of Representations in Cyc*, MCC Technical Report, CYC 066-93.

[GUHA & LENAT, 1990]

R. V. Guha and D. B. Lenat (1990). Cyc: A Midterm Report, *AI Magazine*, *11*,32-59.

[HAARSLEV & MÖLLER, 2001]

V. Haarslev, R. Möller (2001). High Performance Reasoning with Very Large Knowledge Bases: A Practical Case Study In: *Seventeenth International Joint Conference on Artificial Intelligence*, IJCAI-01, August 4-10, Seattle, Washington.

[HAMMER ET AL., 1997]

J. Hammer, H. Garcia-Molina, J. Cho, A. Crespo, R. Aranha (1997). Extracting Semistructured Information from the Web. In: *Proceedings of the Workshop on Management of Semistructured Data*, pp. 18-25, Tucson, Arizona, May 19.

[HANDSCHUH ET AL 2001]

S. Handschuh, S. Staab, and Alexander Mädche (2001). CREAM — Creating relational metadata with a component-based, ontology-driven annotation framework. In: *Proceedings of the First International Conference on Knowledge Capture* (K-CAP).

[HAN ET AL., 1994]

J. Han, L. Liu and Z. Xie (1994). LogicBase: A Deductive Database System Prototype. In: *Proceedings of the. 3rd International Conference on Information and Knowledge Management (CIKM '94)*, Gaithersburg, Maryland, November. pp. 226-233. Also at http://db.cs.sfu.ca/LogicBase/

[HAVELIWALA, 1999]

T. Haveliwala (1999). Efficient Computation of PageRank. *Technical Report Stanford University, Digital Library Project*, Retrieved from http://dbpubs.stanford.edu/pub/1999-31

[HAYES, 1977]

P. Hayes (1997). In defense of logic.In: *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pp. 107-114.

[HEFLIN ET AL., 1999]

J. HEFLIN, J. Hendler, and S. Luke (1999). Coping with Changing Ontologies in a Distributed Environment, in *AAAI Conference Ontology Management Workshop,* pp. 74-79, AAAI Press .

[HEFLIN, 2001]

Jeff Heflin (2001). Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment. *Ph.D. Thesis, University of Maryland, College Park.*

[HEJL, 1991]

P. M. Hejl (1991). Wie Gesellschaften Erfahrungen machen. In: Schmidt, Siegfried J. (Hrsg.), *Gedächtnis - Probleme und Perspektiven der interdisziplinären Gedächtnisforschung*, 2. Auflage, 293-336. Suhrkamp Verlag, Frankfurt am Main.

[HIMMERÖDER ET AL., 1997]

R. Himmeröder, G. Lausen, B. Ludäscher, and C. Schlepphors (1997). On a Declarative Semantics for Web Queries. In: *Proceedings of the International Conference on Deductive and Object-Oriented Databases* (DOOD'97) pp. 386-398.

[HOFFMANN ET AL., 1999]

A. Hoffman, H. Roester, M. Daniel, R. Wegner, S. Decker, and F. Höhn (1999). Entwicklung eines "Ergonomischen Beratungs- und Unterstuetzungssystems fuer Design" (ERBUS) im Rahmen von WORKS: Final Report for the BMBF, Werkstatt fuer Design und Informatik, Chemnitz.

[HOLZNER, 1998]

S. Holzner (1998): *XML complete*, McGraw-Hill Companies, Inc. .

[HORI, 2000]

M. Hori, G. Kondoh1, K. Ono1, S. Hirose1, and S. Singhal: Annotation-based Web Content Transcoding. In: *Proceedings of the 9th WorldWideWeb Conference*, Amsterdam, 2000. Retrieved from: http://www9.org/w9cdrom/169/169.html

[HORROCKS ET AL., 1999]

I. Horrocks, U. Sattler, and S. Tobies (1999). Practical reasoning for expressive description logics. In: H. Ganzinger, D. McAllester, and A. Voronkov, (Eds.), *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR '99)*, number 1705 in Lecture Notes in Artificial Intelligence, pp. 161-180. Springer-Verlag.

[HORROCKS AND PATEL-SCHNEIDER, 1999]

I. Horrocks and P. F. Patel-Schneider (1999). Optimising description logic subsumption. *Journal of Logic and Computation*, *9*(3), 267–293.

[HORROCKS ET AL., 2000]

I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. van Harmelen, M. Klein, S. Staab, R. Studer, and E. Motta (2000). The Ontology Inference Layer OIL. *Technical Report*, Free University of Amsterdam. Retrieved from: http://www.cs.vu.nl/~dieter/oil/Tr/oil.pdf

[HOSCHKA, 2000]

P. Hoschka (Ed.) (2000). Synchronized Multimedia Integration Language (SMIL) 1.0 Specification, *W3C Recommendation* 15 June. Retrieved from http://www.w3.org/TR/REC-smil

[HUCK ET AL., 1999]

G. Huck, I. Macherius, and P; Fankhauser (1999). PDOM: Lightweight Persistency Support for the Document Object Model. In: *Proceedings of the 1999 OOPSLA Workshop "Java and Databases: Persistence Options"*. Held at the 14th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'99) 1-2 November. Denver, Colorado.

[HULL & YOSHIKAWA, 1990]

R. Hull and M. Yoshikawa (1990). ILOG: Declarative Creation and Manipulation of Object Identifiers. In: *Proceedings International Conference on Very Large Databases (VLDB'90)*, August, pp. 455-468. Retrieved from ftp://ftp.cs.colorado.edu/users/hull/ilog:vldb90.ps

[IBENS, 1998]

Ortrun Ibens. SETHEO - The CASC-14 System. In: *Journal of Automated Reasoning*, *18*(1), 117.

[ICD, 1997]

IDC - International Data Corporation (1997). *What is Knowledge Management?* IDC #14910, International Data Corp., Framingham (MA), December.

[ISO 8879, 1986]

ISO (International Organization for Standardization) (1986). ISO 8879:1986(E). Information processing - Text and Office Systems - Standard Generalized Markup Language (SGML). First edition - 1986-10-15. [Geneva]: International Organization for Standardization.

[IWAZUME ET AL., 1996]

M. Iwazume, H. Takeda, and T. Nishida (1996). Ontology-based Information Capturing from the Internet. In: R. Green (Ed), *Knowledge Organization and Change: Advances in Knowledge Organization. Proceedings of the Fourth International ISKO Conference,* pp. 261-272, 15-18 July, Washinghton DC, USA, INDEKS Verlag.

[JAESCHKE, 1996]

P. Jaeschke (1996). Geschäftsprozeßmodellierung mit INCOME, In: G. Vossen, J. Becker: *Geschäftsprozeßmodellierung und Workflowmanagement*, Thomson Publishing.

[JANNINK ET AL., 1999]

J. Jannink, P. Mitra, E. Neuhold, S. Pichai, R Studer, and G. Wiederhold (1999): An Algebra for Semantic Interoperation of Semistructured Data; in *1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX '99)*, Chicago, Nov.

[JARKE ET AL., 1997]

M. Jarke, R. Gallersdörfer, M.A. Jeusfeld, M. Staudt, S. Eherer (1995). ConceptBase - a deductive object base for meta data management. In: *Journal of Intelligent Information Systems, Special Issue on Advances in Deductive Object-Oriented Databases*, 4(2), 167-192.

[JOHNSON, 1999]

E. Johnson, C. R. Ramakrishnan, I. V. Ramakrishnan and P. Rao (1999). A Space Efficient Engine for Subsumption-Based Tabled Evaluation of Logic Programs. *Fuji International Symposium on Functional and Logic Programming*, pp. 284-300.

[KAHAN ET AL., 2001]

J. Kahan, M. Koivunen, E. Prud'Hommeaux, and R. R. Swick (2001). Annotea: An Open RDF Infrastructure for Shared Web Annotations. In: *Proceedings of the 9th WorldWideWeb Conference*, Amsterdam. Retrieved from: http://www9.org/w9cdrom/169/169.html

[KAMIN, 1997]

S. Kamin (Ed.). *First ACM-SIGPLAN Workshop on Domain-Specific Languages*. University of Illinois Computer Science Report, Retrieved from: www-sal.cs.uiuc.edu/~kamin/dsl

[KANGASSALO ET AL., 1995]

H. Kangassalo, H. Jaakkola, S. Ohsuga, and B. Wangler (Eds) (1995): *Information Modelling and Knowledge Bases VI*, IOS Press.

[KANDZIA, 1997]

P. Kandzia (1997). Nonmonotonic Reasoning in Florid. In: *4th International Workshop on Logic Programming and Nonmonotonic Reasoning (LPNMR'97)*, July, Dagstuhl Castle, Germany, Springer LNCS 1265.

[KARNER, 1996]

H. F. Karner (1996). Die personelle und strukturelle Seite des intellektuellen Kapitals. in: U. Schneider (Hrsg.) *Wissensmanagement*, 77-132. Frankfurter Allgemeine Zeitung, Frankfurt am Main.

[KARP ET AL., 1999]

P. D. Karp, V. K. Chaudhri, and J. Thomere (1999). XOL: An XML-Based Ontology Exchange Language. Retrieved from *http://www.ai.sri.com/~pkarp/xol/*

[KASCHEK ET AL., 1995]

R. Kaschek, C. Kohl, H. C. Mayr (1995). Cooperations - An Abstraction Concept Suitable for Business Process Re-Engineering. In: *J. Györkös, M. Krisper, H.C. Mayr (eds.): ReTIS'95 - Re-Technologies for Information Systems*. OCG Lecture Notes 80, Oldenbourg.

[KASHYAP & SHET, 1996]

V. Kashyap and A. Sheth (1996). Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies. In: M. Papazoglou and G. Schlageter (Eds.): *Cooperative Information Systems: Current Trends and Directions*, Academic Press.

[KASHYAP, 1999]

V. Kashyap (1999). Design and Creation of Ontologies for Environmental Information Retrieval. In: *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW '99)*, Banff, Canada, October. Retrieved from: http://sern.ucalgary.ca/KSI/KAW/KAW99/papers/Kashyap1/kashyap.pdf

[KENT, 1999]

R. E. Kent (1999). Conceptual Knowledge Markup Language: The Central Core . In *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management*, Banff. Retrieved from http://sern.ucalgary.ca/KSI/KAW/KAW99/papers.html

[KERBER, 1994]

M. Kerber (1994). On the Translation of Higher-Order Problems into First-Order Logics. In: Tony Cohn, (Ed.), *Proceedings of the 11th ECAI*, pp. 145-149, Amsterdam, The Netherlands.

[KESSLER, 1995]

M. Kesseler (1995). A Schema Based Approach to HTML Authoring. in: *Proceedings of the 4th Inernational. World Wide Web Conference (WWW-4)*. Boston, December. Also: *World Wide Web Journal*. 1/96. Sebastopol, Calif., O'Reilly, 1996. Retrieved from: http://www.w3.org/ Conferences/WWW4/Papers2/145/

[KIF, 1998]

Knowledge Interchange Format (1998). *Draft proposed American National Standard* (dpANS), NCITS.T2/98-004. Retrieved from: http://logic.stanford.edu/kif/dpans.html

[KIFER ET AL., 1995]

M. Kifer, G. Lausen, and J. Wu (1995). Logical Foundations of Object-Oriented and Frame-Based Languages, *Journal of the ACM, 42*, 741-843.

[KINGSTON ET AL., 1997]

J. Kingston, A. Griffith,and T. Lydiard (1997). Multi-Perspective Modelling of the Air Campaign Planning process. In: *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Japan, August.

[KIRIKOVA & BUBENKO, 1994]

M. Kirikova and J.A. Bubenko (1994). Software Requirements Acquisition through Enterprise Modeling. In: *Proceedings of the 6th International Conference on Software Engineering and Knowledge Engineering (SEKE ´94)*, Jurmala.

[KIRSH, 2000]

D. Kirsh (2000). A Few Thoughts on Cognitive Overload. *Intellectica 30*, pp. 19-51. Also at http://icl-server.ucsd.edu/~kirsh/Articles/Overload/published.html

[KLEIN & FANKHAUSER, 1997]

B. Klein and P. Fankhauser (1997). Error tolerant document structure analysis. *International Journal on Digital Libraries 1*, 344-257, Springer.

[KLEIN & FENSEL, 2001]

M. Klein and D. Fensel (2001). Ontology versioning for the Semantic Web. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, Stanford University, California, July 30 - August 1.

[KNUTH, 1984]

D.E. Knuth (1984). Literate programming. *The Computer Journal, 27*, 97-111.

[KONOPNICKI & SHMUELI, 1995]

D. Konopnicki, and O. Shmueli (!995). W3QS: A query stem for the World Wide Web. In: *Proceedings of the International Conference on Very Large Data Bases* (VLDB ‘95) pp. 54-65, Zürich.

[KUSHMERICK ET AL., 1997]

N. Kushmerick, D.S. Weld, and R. Doorenbos (1997). Wrapper induction for information extraction. In: *Proceedings of the International Joint Conference on Artificial Intelligence* (IJCAI).

[KÜHN & ABECKER, 1998]

O. Kühn and A. Abecker (1998). Corporate Memories for Knowledge Management in Industrial Practice: Prospects and Challenges. In: U. M. Borghoff and R. Pareschi (Eds.): *Information Technology for Knowledge Management*, pp. 183-206, Springer, Berlin, Heidelberg, New York.

[KUNZ & RITTEL, 1970]

W. Kunz W and H. Rittel (1970). Issues as Elements of Information Systems. *Working paper 131. Center for Planning and Development Research*, University of California, Berkeley.

[LABROU & FININ, 1999]

Y. Labrou, T.W. Finin (1999). Yahoo! As an Ontology: Using Yahoo! Categories to Describe Documents. In: *Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management*, pp. 180-187, Kansas City, Missouri. November. ACM Press.

[LAGOZE AND HUNTER, 2001]

C. Lagoze and J. Hunter (2001). The ABC Ontology and Model. In: *Journal of Digital Information. Special Issue on Metadata*. Retrieved from: http://jodi.ecs.soton.ac.uk/

[LAKSHMANAN ET AL., 1996]

L. V. S. Lakshmanan, F. Sadri, and I. N. Subramanian (1996). A declarative language for querying and restructuring the Web. In *RIDE'96, pages 12-21. IEEE Computer Society Press.*

[LAMPING & RAO, 1996]

J. Lamping and R. Rao (1995). The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages and Computing*, *6*(4).

[LANDES ET AL., 1998]

D. Landes, K. Schneider and F. Houdek (1998). Organizational Learning and Experience Documentation in Industrial Software Projects. In: *[Abecker et al., 1998a]*.

[LASSILA & SWICK, 1999]

O. Lassila, R. Swick (Eds) (1999). Resource Description Framework (RDF) Model and Syntax Specification, *W3C Recommendation 22 February.* Retrieved from http://www.w3.org/TR/ REC-rdf-syntax/

[LAWRENCE & GILES, 1999]

S. Lawrence, and C. L. Giles (1999). Accessibility of information on the web. *Nature*, *400*, 107-109.

[LIEBOWITZ & WILCOX, 1997]

J. Liebowitz and L.C. Wilcox (1997). *Knowledge Management and Its Integrative Elements*, New York, CRC Press, Boca Raton.

[O'LEARY, 1998A]

D. O'Lear (1998). Enterprise Knowledge Management. In: *IEEE Computer*, *31*(3), 54-61.

[O'LEARY, 1998B]

D. O'Leary (1998). Using AI in Knowledge Management: Knowledge Bases and Ontologies. *IEEE Intelligent Systems*, May/June, pp. 34-39.

[O'LEARY, 1998C]

D. O'Leary (1998). Knowledge Management Systems: Converting and Connecting. *IEEE Intelligent Systems*, May/June, pp. 30-33.

[LENAT & GUHA, 1990

D. B. Lenat and R. V. Guha (1990). Building large knowledge-based systems. *Representation and inference in the Cyc project*. Reading, Massachusetts, Addison-Wesley.

[LEVY ET AL., 1996]

A. Y. Levy, A. Rajaraman, and J. J. Ordille (1996). Query-Answering Algorithms for Information Agents. In: *Proceedings of the National Conference on AI (AAAI '96)*, Portland, Oregon, August 4-8.

[LEVY & ROUSSET, 1996]

A. Y. Levy and M.-C. Rousset (1996). CARIN: A Representation Language Combining Horn Rules and Description Logics. In: *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96)*, Budapest, August 12-16.

[LIEBOWITZ AND WILCOX, 1997]

J. Liebowitz and L.C. Wilcox (Eds.) (1997). *Knowledge Management and Its Integrative Elements*, Boca Raton, New York, CRC Press.

[LIU & ORGUN, 1996]

C. Liu and M. A. Orgun (1996). Clocked Temporal Logic Programming. In: *Proceedings of the 19th Australasian Computer Science Conference*, Melbourne, Australia.

[LLOYD, 1987]

J.W. Lloyd (1987). *Foundations of Logic Programming*, 2nd Edition. Berlin, Springer.

[LLOYD & TOPOR, 1984]

J. W. Lloyd and R. W: Topor (1984). Making Prolog more Expressive, *Journal of Logic Programming*, *3*, 225-240.

[LU ET AL., 1996]

J. Lu, P. Barclay, and J. Kennedy (1996). On Temporal Versioning in Object Oriented Databases, (MoBIS '96), *Modelling Business Information Systems*, Cottbus, Germany, Oct.

[LUDÄSCHER ET AL., 1998]

B. Ludäscher, R. Himmerorder, G. Lausen, W. May, and C. Schlepphorst (1998). Managing Semistructured Data with FLORID: A deductive Object-Oriented Perspective. *Information Systems*, 23(8), 589-613.

[LUKE ET AL., 1997]

S. Luke, L. Spector, D. Rager, and J. Hendler (1997). Ontology-based Web Agents. In: *Proceedings of the First International Conference on Autonomous Agents, (AA '97)*.

[LYMAN ET AL., 2000]

P. Lyman, H. R. Varian, J. Dunn, A. Strygin, K. Swearingen (2000). *How much Information?* University of California at Berkley, Oktober. Rretrieved from http://www.sims.berkeley.edu/research/projects/how-much-info/

[MacGregor, 1990]

R.M. MacGregor (1990). LOOM Users Manual, ISI/WP-22, USC/Information Sciences Institute

[MacGregor, 1994]

R. M. MacGregor (1994). A description classifier for the predicate calculus. In: *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 213–220, Seattle, Washington.

[Marin, 2001]

M. Marin (2001). Workflow Process Definition Interface - XML Process Definition Language. *Workflow Management Coalition Workflow Standard*, May. Retrieved from: http:// www.wfmc.org/standards/docs/xpdl_010522..pdf

[Mark, 1996]

B. Mark (1996). *Special Issue on Data-Mining*, *IEEE-Expert*, *11*(5).

[Maurer & Dellen, 1998]

F. Maurer and B. Dellen (1998). An Internet Based Software Process Management Environment. In: *Proceedings of the. ICSE '98 Workshop on Software Engineering over the Internet (ICSE '98).*

[Maier, 1986]

D. Maier (1986). *A logic for objects*: Technical Report Report TR CS/E-86-012, Oregon Graduate Center, Nov.

[Martin & Eklund, 2001]

P. Martin and P. Eklund (2001). Large-scale cooperatively-built heterogeneous KBs. In: *Proceedings of ICCS'01, 9th International Conference on Conceptual Structures* (Springer Verlag, LNAI 2120, pp. 231-244), Stanford University, California, July 30 to August 3. Retrieved from http://meganesia.int.gu.edu.au/~phmartin/WebKB/doc/papers/iccs01/

[Martin & Eklund, 1999]

P. Martin and P. Eklund (1999). Embedding Knowledge in Web Documents. Proceedings of *WWW8, 8th International World Wide Web Conference*, special issue of *The International Journal of Computer and Telecommunications Networking*, Toronto, Canada, May 11-14.

[May, 2000]

W. May (2000). An Integrated Architecture for Exploring, Wrapping, Mediating and Restructuring Information from the Web. In: *Australian Database Conference (ADC2000)* Jan. 31 - Feb. 3, Canberra, Australia.

[May et al., 1997]

W. May, C. Schlepphorst,and G. Lausen (1997). Integrating Dynamic Aspects into Deductive Object-Oriented Databases. In: *3rd International Workshop on Rules in Database Systems, (RIDS '97)*, pp. 20-34, Skövde, Sweden, 7, Springer LNCS 1312 .

[McCarthy, 1959]

J. McCarthy (1959). Programs with common sense. In: *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pp. 75-91, London. Her Majesty's Office.

[McCarthy & Hayes, 1969]

J. McCarthy and P. Hayes (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence, 4*, 463-502.

[McCarthy, 1989]

J. McCarthy (1989). Artificial Intelligence, Logic and Formalizing Common Sense. In: R. Thomason (Ed.), *Philosophical Logic and Artificial Intelligenc*, Dordrecht, Kluwer Academic.

[McGuinness et al., 2000]

D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder (2000). The Chimaera Ontology Environment. Proceedings of the *The Seventeenth National Conference on Artificial Intelligence (AAAI 2000)*, Austin, Texas, 30 July to - 3 August.

[McHugh et al., 1997]

J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom: Lore: A Database Management System for Semistructured Data. *SIGMOD Record*, *26*(3), 54-66, September. Retrieved from: http://www-db.stanford.edu/pub/papers/lore97.ps

[Mena et al., 1998]

E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth (1998). Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure. In: *[Guarino, 1998]*.

[Mendelzon et al., 1997]

A. O. Mendelzon, G.A. Mihaila, and T. Milo. Querying the World Wide Web. In: *International Journal on Digital Libraries 1*(1), 54-67.

[Miller, 1990]

G.A. Miller (1990). Wordnet. An On-Line Lexical Database. *International Journal of Lexicography 3-4*, 235-312.

[Milton et al., 1999]

N. Milton, N. Shadbolt, H. Cottam, and M Hammersley (1999). Towards a knowledge technology for knowledge management. *International Journal of Human-Computer Studies, Special Issue on Organizational Memory*, 51 (3), 615-641.

[Minsky, 1975]

M. L. Minsky (1975). A Framework for Representing Knowledge. P. Winston (Ed.), *The Psychology of Computer Vision*, New York.

[Mitra et al., 2000]

P. Mitra, M. Kersten and G. Wiederhold (2000). Graph-Oriented Model for Articulation of Ontology Interdependencies. In: *Proceedings of the 7th International Conference on Extending Database Technology*, (EDBT).

[Mitra et al., 2001]

P. Mitra, G. Wiederhold and S. Decker: A Scalable Framework for Interoperation of Information Sources. In: *Proceedings of the 1st Semantic Web Working Symposium*, Stanford. Retrieved from http://www.semanticweb.org/SWWS/

[Motta et al., 1996]

E. Motta, K. O'Hara, N. Shadbolt, A. Stutt, and Z. Zdrahal (1996). Solving VT in VITAL: A Study in Model Construction and Knowledge Reuse. *International Journal of Human-Computer Studies*, *44* (3-4).

[Motta, 1999]

E. Motta (1999). *Reusable Components for Knowledge Modeling*, Amsterdam, IOS Press.

[MUSEN, 1993]

M.A. Musen (1993). An Overview of Knowledge Acquisition. In: J.M. David et al. (Eds.), *Second Generation Expert Systems*, Springer-Verlag.

[NEBEL, 1996]

B. Nebel (1996). Artificial Intelligence: A Computational Perspective. In G. Brewka (Ed.), *Principles of Knowledge Representation*, CSLI publications, Studies in Logic, Language and Information, Stanford.

[NECHES ET AL., 1991]

R. Neches, R. E. Fikes, T. Finin, T. R. Gruber, T. Senator, and W. R. Swartout (1991). Enabling Technology for Knowledge Sharing, *AI Magazine, 12*(3),36-56.

[NEUBERT, 1993]

S. Neubert (1993). Model Construction in MIKE. In: *Current Trends in Knowledge Acquisition-(EKAW '93), 7th European Knowledge Acquisition Workshop*, Toulouse, France, September. Lecture Notes in Artificial Intelligence, pp. 200-219, Springer-Verlag, Berlin-Heidelberg.

[NEUMANN ET AL., 1997]

G. Neumann, R. Backofen, J. Baur, M. Becker, and C. Braun (1997). Information Extraction Core System for Real World German Text Processing. In: *Fifth Conference on Applied Natural Language Processing*, Association for Computational Linguistics, 31 March - 3 April, Washington, D.C. .

[NEWELL, 1982]

A. Newell (1982). The Knowledge Level, *Artificial Intelligence*, *18*, 87—127.

[NICHOLS AND TWIDALE, 1999]

D. M. Nichols and M. B. Twidale (1999). Computer supported cooperative work and libraries. In: *Vine 109*(10-15) (special issue on Virtual communities and information services).

[NONAKA & TAKEUCHI, 1995]

I. Nonaka and H. Takeuchi (1995). *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation.* Oxford University Press.

[NOY & MUSEN, 1999]

N. Fridman Noy, and M. Musen (1999). An Algorithm for Merging and Aligning Ontologies: Automation and Tool Support. In: *Proceedings of the Workshop on Ontology Management at the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, Orlando, FL.

[NWANA & NDUMU, 1999]

H. S. Nwana and D. T. Ndumu (1999). A Perspective on Software Agents Research. In: *The Knowledge Engineering Review*, 14(2), 1-18.

[OLIVER ET AL., 1999]

D. E. Oliver, Y. Shahar, M. A. Musen, and E. H. Shortliffe, (1999). Representation of change in controlled medical terminologies. A*rtificial Intelligence in Medicine*, *15*(1), 53–76.

[OBERWEIS ET AL., 1994]

A. Oberweis, G. Scherrer, W. Stucky (1994). INCOME/STAR: methodology and tools for the development of distributed information systems. *Information Systems 19*(8), 643-660.

[ORGUN, 1996]

M. A. Orgun (1996). On temporal deductive databases. *Computational Intelligence*, *12* (2), 235-259.

[ORGUN & MA, 1996]

M. A. Orgun and W. Ma (1996). An overview of temporal and modal logic programming. In: D. M. Gabbay and H. J. Ohlbach (Eds.), *Proceedings of ICTL '94: The First International Conference on Temporal Logic*, pp.445-479 (Gustav Stresemann Institut, Bonn, Germany, July 11-14). LNAI 827, Springer-Verlag Berlin Heidelberg .

[ODLYZKO, 1997]

A. Odlyzko (1997). The Economics of Electronic Journals, *First Monday 2(8)*. Retrieved from http://www.firstmonday.dk/issues/issue2_8/odlyzko.

[PAGE ET AL., 1998]

L. Page, S. Brin, R. Motwani, and T. Winograd (1998). The PageRank citation ranking: Bringing order to the web, Stanford University, *Unpublished Manuscript.*

[PAPAKONSTANTINOU ET AL., 1995]

Y. Papakonstantinou, H. Garcia Molina, and J. Widom (1995). Object Exchange Across Heterogeneous Information Sources. In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, Taipei, Taiwan, March.

[PAPAKONSTANTINOU ET AL., 1996]

Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina (1996). Object fusion in mediator systems. In: *Proceedings of the International Conferenc*. *on Very Large Data Bases (VLDB)*, pp. 413-424, Bombay, India, September.

[PIROLLI & CARD, 1999]

Pirolli, P. and S.K. Card (1999). Information foraging. *Psychological Review, 106*, 643-675.

[PIROLLI ET AL., 2000]

P. Pirolli, S.K. Card, and M.M. Van Der Wege (2000). The Effect of Information Scent on Searching Information Visualizations of Large Tree Structures. *Advanced Visual Interfaces, AVI 2000*. Palermo, Italy. ACM.

[PIROLLI ET AL., 2001]

P. Pirolli, S.K. Card, and M.M. Van Der Wege (2001). Visual Information Foraging in a Focus + Context Visualization. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 506-513, Seattle, Washington, (CHI).

[POPPER, 1959]

K. R. Popper (1992). *The Logic of Scientific Discovery,* London, (reprint of 1959).

[PRINZ & SYRI ET AL., 1998]

W. Prinz and A. Syri (1998). An Environment for Cooperative Knowledge Processing. In: U. M. Borghoff and R. Pareschi (Eds.)*: Information Technology for Knowledge Management*, Berlin, Heidelberg, New York, Springer, pp. 101-119.

[PROBST ET AL., 1997A]

G. Probst, S. Raub, and K. Romhardt (1999). *Wissen managen: Wie Unternehmen ihre wertvollste Ressource optimal nutzen*, Wiesbaden, Gabler.

[PROBST ET AL, 1997B]

G. Probst, S. Raub, K. Romhardt (1997). Wissen managen. *Frankfurter Allgemeine Zeitung*, Frankfurt am Main, Wiesbaden, Gabler.

[PRZYMUSINSKI, 1988A]

T. C. Przymusinski (1998). Perfect Model Semantics. In: *Proceedings of the International Logic Programming Conference*, pp. 1081-1096, Seattle, Washington.

[PRZYMUSINSKI, 1988B]

T. C. Przymusinski: On the Declarative Semantics of Deductive Databases and Logic Programs. In: J. Minker (Ed.), *Foundations of Deductive Databases and Logic Programming*, Los Altos, CA, Morgan Kaufmann.

[RAGETT ET AL., 1998]

D. Raggett, A. Le Hors, and I. Jacobs (Eds.) (1998). HTML 4.0 Specification. *W3C Recommendation*, 24. April. Retrieved from http://www.w3.org/TR/REC-html40

[RAMACKERS & VERRIJN-STUART, 1995]

G. J. Ramackers, and A. A. Verrijn-Stuart (1995). Conceptual Model Requirements for Integrated Business and Information System Development. In *[Kangassalo et al., 1995]*.

[RAMAKRISHNAN ET AL., 1994]

R. Ramakrishnan, D. Srivastava, S. Sudarshan and P. Seshadri: The CORAL Deductive System. *The VLDB Journal, Special Issue on Prototypes of Deductive Database Systems*, *3*(2), 161-210. Also at http://www.cs.wisc.edu/coral/

[RAO ET AL., 1997]

P. Rao, K. F. Sagonas, T. Swift, D. S. Warren, J. Freire (1997). XSB: A System for Efficiently Computing WFS. In: *Proceedings of Logic Programming and Non-Monotonic Reasoning LPNMR 1997*, Dagstuhl, Germany, July 28-31. Lecture Notes in *Computer Science*, Vol. 1265, Springer 1997, pp. 431-441.

[REHÄUSER & KRCMAR, 1996]

J. Rehäuser, J. H. Krcmar (1996). Wissensmanagement in Unternehmen. In: G. Schreyögg, P. Conrad, (Hrsg.) *Managementforschung 6 - Wissensmanagement*, 1-40. Berlin, Walter de Gruyter.

[REIMER, 1997]

U. Reimer (1997). Knowledge Integration for Building Organisational Memories. In: *[Abecker et al., 1997]*.

[REITER, 1982]

R. Reiter (1982). Towards a Logical Reconstruction of Relational Database Theory. *On Conceptual Modelling* (Intervale) pp. 191-233.

[RIBIÈRE & MATTA, 1998]

M. Ribière and N. Matta (1998). Virtual Enterprise and Corporate Memory. In: *[Abecker et al., 1998a]*.

[RICH & KNIGHT, 1991]

E. Rich and K. Knight (1991). *Artificial Intelligence*, McGraw-Hill, New York, 2nd edition.

[RODDICK, 1995]

J. F. Roddick (1995). A Survey of Schema Versioning Issues for Database Systems. *Information and Software Technology, 37*(7), 383--393.

[ROGERS & RECTOR, 1997]

E. Rogers and A. L. Rector (1997). Terminological Systems: Bridging the Generation Gap. In: *Annual Fall Symposium of American Medical Informatics Association*. Nashville TN, Hanley & Belfus Inc. Philadelphia PA: pp. 610-614.

[ROMHARDT, 1997]

K. Romhardt (1997). Processes of Knowledge Preservation: Away from a Technology Dominated Approach. In: A. Abecker, S. Decker, K. Hinkelmann, U. Reimer (Eds.): Proceedings of the Workshop "*Wissensbasierte Systeme für das Wissensmanagement im Unternehmen*."

[RUMBAUGH ET AL., 1991]

J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen (1991). *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs.

[SAHUGUET & AZAVANT, 1999]

A. Sahuguet and F. Azavant (1999). Building light-weight wrappers for legacy Web data-sources using W4F. In: *Proceedings of the International Conference on Very Large Databases (VLDB)*.

[SCIORE, 1991]

E. Sciore (1991). Using annotations to support multiple kinds of versioning in an object-oriented database system. In: *ACM Transactions on Database Systems (TODS)* Vol. 16(3), September.

[SCHEER, 1994]

A.-W. Scheer (1994). *Business Process Engineering: Reference Models for Industrial Enterprises*, Springer, 2nd ed. .

[SCHENK, 1999]

M. Schenk (1999). Ontology-based semantical annotation of XML. Diploma Thesis, Insitut AIFB, University of Karlsruhe.

[SCHLENOFF ET AL., 2000]

C. Schlenoff, M. Gruninger, F. Tissot, J. Valois, J. Lubell,and J. Lee (2000). The Process Specification Language (PSL): Overview and Version 1.0 Specification, *NISTIR 6459*, National Institute of Standards and Technology, Gaithersburg, MD. Retrieved from: http://www.mel.nist.gov/psl/pubs/PSL1.0/paper.doc

[SCHLEPPHORST, 1997]

C. Schlepphorst (1997). Semi-naive Evaluation of F-logic Programs, *Technical Report 85*, University of Freiburg.

[SCHNEIDER, 1996]

U. Schneider (1996). Management in der wissensbasierten Unternehmung. In: U. Schneider, (Hrsg.) *Wissensmanagement*, 13-48. Frankfurt am Main, Frankfurter Allgemeine Zeitung.

[SCHNYDER, 1989]

A. B. Schnyder (1989). Unternehmungskultur: Die Entwicklung eines Unternehmungskultur-Modells unter Berücksichtigung ethnologischer Erkenntnisse und dessen Anwendung auf die Innovations-Thematik. In: *Europäische Hochschulschriften. Reihe 5. Volks- und Betriebswirtschaft*; 987, S. 227-243.

[SCHREIBER ET AL., 1993]

A. Th. Schreiber, B. J. Wielinga, and J. A. Breuker (Eds.) (1993). *KADS: A Principled Approach to Knowledge-Based System Development, vol 11 of Knowledge-Based Systems Book Series*, London, Academic Press.

[SCHREIBER ET AL., 1994]

A. Th. Schreiber, B. Wielinga, J. M. Akkermans, W. Van De Velde, and R. de Hoog: CommonKADS. A Comprehensive Methodology for KBS Development, *IEEE Expert*, *9*(6), 28-37.

[SCHREIBER & WIELINGA, 1993]

G. Schreiber, B.J. Wielinga (1993). KADS and Conventional Software Engineering. In: *KADS - A Principled Approach To Knowledge Based System Development*, Academic Press.

[SCHREIBER ET AL., 1999]

A. T. Schreiber, J. M. Akkermans, A. A. Anjewierden, R. de Hoog, N. R. Shadbolt, W. V. de Velde, and B. J. Wielinga (1999). *Engineering and Managing Knowledge, The CommonKADSmethodology*. MIT Press.

[SCHÜPPEL, 1996]

J. Schüppel: *Wissensmanagement*. Wiesbaden, Deutscher Universitäts Verlag.

[SHAPIRO & RAPAPORT, 1987]

S. C. Shapiro and William J. Rapaport (1987). SNePS considered as a fully intensional propositional semantic network. In: N. Cercone and G. McCalla (Eds.), *The Knowledge Frontier*, New York, Springer-Verlag, pp. 263-315.

[SHAW & GARLAN, 1996]

M. Shaw and D. Garlan (1996). S*oftware Architectures. Perspectives on an Emerging Discipline*, Prentice-Hall.

[SHETH & KLAS, 1998]

A. Sheth and W. Klas (Eds.) (1998). *Multimedia Data Management: Using Metadata to Integrate and Apply Digital Media*, McGraw-Hill.

[SIMONE & DIVITINI, 1998]

C. Simone and M. Divitini (1998). Ariadne: Supporting Coordination Through a Flexible Use of Knowledge Processes. In: *[Borghoff & Pareschi, 1998a]*, pp. 121-148.

[SOWA, 1984]

J. F. Sowa (1984). *Information Processing in Mind and Machine. Reading*, MA: Addison-Wesley Publ..

[SOWA, 2000]

J. F. Sowa (2000). *Knowledge Representation, Logical, Philosophical, and Computational Foundations*. Pacific Grove, Brooks/Cole.

[SMITH & POULTER, 1999]

H. Smith, and K. Poulter (1999): First bring semantic order to the world of XML. Share the Ontology. In: *XML-based Trading Architectures*. In: *Communications of the ACM 42*(3) March, pp. 110-111.

[STAAB ET AL., 2000]

S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Mädche, H-P. Schnurr, R. Studer, and Y. Sure (2000). Semantic Community Web Portals. In: *WWW9 - Proceedings of the 9th International World Wide Web Conference*, Amsterdam, The Netherlands, May, 15-19, Elsevier.

[STAAB & SCHNURR, 2000]

S. Staab, and H. P. Schnurr (2000). Smart Task Support through Proactive Access to Organizational Memory. In: *Journal of Knowledge-based Systems*. Elsevier. September.

[STEFIK & BOBROW, 1986]

M. Stefik and D.G. Bobrow (1986). Object-Oriented Programming: Themes and Variations. In: *The AI Magazine*, 6 (4), 40 - 62.

[STEFIK, 1995]

M. Stefik (1995). *Introduction to Knowledge Systems*, San Francisco, Morgan Kaufman.

[STEIN & ZWASS, 1995]

E. W. Stein and V. Zwass (1995). Actualizing Organizational Memory With Information Technology. *Information Systems Research*, no. 2, *6*, 85-117.

[STUDER ET AL., 2000]

R. Studer, S. Decker, D. Fensel, and S. Staab. Situation and Prospective of Knowledge Engineering (2000). In: J.Cuena, Y.Demazeau, A. Garcia, J.Treur (Eds.), *Knowledge Engineering and Agent Technology*. IOS Series on Frontiers in Artificial Intelligence and Applications. IOS Press.

[STUDER ET AL., 1998]

R. Studer, D. Fensel, S. Decker, and V. R. Benjamins (1999). Knowledge Engineering: Survey and Future Directions. In: F. Puppe, (ed.), *Knowledge-based Systems: Survey and Future Directions, Proceedings of the. 5th German Conference on Knowledge-based Systems*, Würzburg, Lecture Notes in AI, Springer Verlag.

[SUCIU, 1998]

D. Suciu (1998). An Overview of Semistructured Data. In: *SIGACT News*, *29*(4), 28-38 , December.

[SURE ET AL., 2000]

Y. Sure, A. Mädche, and S. Staab (2000). Leveraging Corporate Skill Knowledge - From ProPer to OntoProper. In: D. Mahling, U. Reimer (Eds.), *Proceedings of the Third International Conference on Practical Aspects of Knowledge Management (PAKM 2000)*, Basel, Switzerland, October.

[SURE, 2001A]

Y. Sure (2001). A Tool supported Methodology for Ontology-based Knowledge Management. In: LLWA 01 - Tagungsband der GI-Workshopwoche "Lernen - Lehren - Wissen - Adaptivität", R. Klinkenberg, S. Rüping, A. Fick, N. Henze, C. Herzog, R. Molitor, O. Schröder (Eds.), *Technical Report Nr. 763, S. 236-242, University of Dortmund*, Oktober.

[SURE & STUDER, 2001]

Y. Sure and R. Studer (2001). OntoEdit. Institute AIFB, University of Karlsruhe, *On-To-Knowledge Deliverable 3*.

[SUTCLIFFE & SUTTNER, 1998]

G. Sutcliffe, C.B. Suttner (1998). The TPTP Problem Library: CNF Release v1.2.1. In: *Journal of Automated Reasoning*. *21*(2), 177-203.

[SUTCLIFFE, 2001]

G. Sutcliffe (2001). The CADE-17 ATP System Competition. *Journal of Automated Reasoning 27*(3), 227-250.

[SVEIBY & LLOYD, 1990]

K. E. Sveiby and T. Lloyd (1990). *Das Management des Know-how*, Frankfurt am Main, Campus Verlag.

[SWARTOUT ET AL., 1996]

W. Swartout, R. Patil, K. Knight, and T. Russ (1996). Toward distributed use of large-scale ontologies. In: Gaines, B. & Musen, M., (Eds.), *Proceedings of the Tenth Knowledge Acquisition for Knowledge-BasedSystems Workshop*.

[THOMPSON ET AL., 2000]

H. S. Thompson, D. Beech, M. Maloney, N. Mendelsohn (Eds.) (2000). *XML Schema Part 1: Structures W3C Working Draft 7 April*. Retrieved from http://www.w3.org/TR/2000/WD-xmlschema-1-20000407/

[TICHY, 1985]

W. F. Tichy (1985). RCS - A system for version control. *Software Practice and Experience*, *15*(7), 637 - 654.

[TOP & AKKERMANS, 1994]

J. Top and H. Akkermans (1994). Tasks and Ontologies in Engineering Modeling, *International Journal of Human-Computer Studies (IJHCS)*, *41*, 585-617.

[TOVE, 1995]

TOVE (1995). Manual of the Toronto Virtual Enterprise. Technical Report, Department of Industrial Engineering, University of Toronto, 1995. Retrieved from: http://www.ie.utoronto.ca/EIL/tove/ontoTOC.html

[TSCHAITSCHIAN ET AL., 1997]

B. Tschaitschian, A. Abecker, and F. Schmalhofer (1997). Information Tuning with KARAT: Capitalizing on existing documents. In: *Proceedings of the European Workshop on Knowledge Acquisition, Modeling and Management (EKAW '97)*, Springer Verlag, LNAI 1319.

[TUTTLE ET AL., 1991]

M.S. Tuttle, D.D. Sherertz, M.S. Erlbaum, W.D. Sperzel, L.F. Fuller, N.E. Olson, S.J. Nelson, J.J. Cimino, and C.G. Chute (1991). Adding your terms and relationships to the UMLS Metathesaurus. In: P.D. Clayton (Ed.), *Proceedings of the Annual Symposium on Computer Applications in Medical Care*. New York: McGraw-Hill, pp. 219-223.

[USCHOLD ET AL., 1998]

M. Uschold, M. King, S. Moralee, Y. Zorgios (1998). The Enterprise Ontology. *Knowledge Engineering Review*, *13*(1), 31-89. Cambridge University Press. Retrieved from: http://www.aiai.edu.ac.uk/~enterprise/enterprise/

[USCHOLD & GRUNINGER, 1996]

M. Uschold, M. Gruninger (1996). Ontologies: Principles, Methods and Applications. In: *Knowledge Engineering Review, 11*(2), 93-155. auch: Technical Report AIAI-TR-191, Artificial Intelligence Applications Institute, University of Edinburgh. Februar 1996. Retrieved from: ftp://ftp.aiai.ed.ac.uk/pub/documents/1996/96-ker-intro-ontologies.ps.gz

[VAGHANI ET AL., 1996]

J. Vaghani, K. Ramamohanarao, D. B. Kemp, Z. Somogyi, P. J. Stuckey, T.S. Leask, and J. Harland (1994). The Aditi deductive database system. In: *The VLDB Journal, 3*(2), 245--288. See also: http://www.cs.mu.oz.au/~kemp/aditi/aditi.html.

[VAN GELDER ET AL., 1991]

A. van Gelder, K. A. Ross, and J. S. Schlipf (1991). The well-founded semantics for general logic programs. *Journal of the ACM, 38*, 620-650.

[VAN HEIJST ET AL., 1998]

G. van Heijst, R. van der Spek and E. Kruizinga (1998). The Lessons Learned Cycle. In: *[Borghoff & Pareschi, 1998a]* .

[VAN HEIJST ET AL., 1997]

G. van Heijst, A. T. Schreiber, and B. J. Wielinga (1997). Using Explicit Ontologies in Knowledge-Based System Development, *International Journal of Human-Computer Studies (IJHCS), 46*(6).

[WADDINGTON, 1996]

P. Waddington. (1996) *Dying for information: an investigation of information overload in the UK and world-wide*. Reuters Business Information, London.

[WAGNER, 1995]

M. P. Wagner (1995). *Groupware und neues Management.* Vieweg Verlag, Wiesbaden, Braunschweig.

[WEIDENBACH, 1997]

C. Weidenbach. SPASS (1997). Version 0.49. *Journal of Automated Reasoning, 18(2), 247-252*. Special Issue on the CADE-13 Automated Theorem Proving System Competition.

[WEITZ, 1999]

W. Weitz (1999). *Integrierte Dokumenten- und Ablaufmodellierung im Electronic Commerce*, Dissertation, University of Karlsruhe (TH), Fakultät für Wirtschaftswissenschaften. Shaker.

[WELTY, 1998]

C. Welty (1998). The Ontological Nature of Subject Taxonomies. In: *International Conference on Formal Ontology in Information Systems (FOIS '98).*

[WIEDERHOLD, 1992]

G. Wiederhold (1992). Mediators in the Architecture of Future Information Systems, *IEEE Computer, 25*(3), 38–49.

[WIEDERHOLD & GENESERETH, 1997]

G. Wiederhold and M. Genesereth (1997). The Conceptual Basis for Mediation Services. In: *IEEE Intelligent Systems*, September/October, pp. 48-47.

[WIELINGA ET AL., 1993]

B. J. Wielinga, W. Van De Velde, A. Th. Schreiber, and J. M. Akkermans (1993): Towards a Unification of Knowledge Modeling Approaches. In J.M. David et al. (Eds.), *Second Generation Expert Systems*, Berlin: Springer-Verlag.

[WIELINGA, ET AL 1994]

B. J. Wielinga, J. M. Akkermans, H. A. Hassan, O. Olsson, K. Orsvärn, A. Th. Schreiber, P. Terpstra, W. Van de Velde, and S. Wells (1994). Expertise model definition document. deliverable *ESPRIT Project P-5248 /KADS-II/M2/UvA/026/5.0*, University of Amsterdam, Free University of Brussels and Netherlands Energy Research Centre ECN, June.

[WIELINGA ET AL., 1997]

B. J. Wielinga, J. Sandberg, and G. Schreiber (1997). Methods and Techniques for Knowledge Management: What has Knowledge Engineering to Offer? *Expert Systems with Applications*, *13*(1), 73-84.

[WILLE & RAMMING, 1999]

David S. Wile, J. Christopher Ramming (Eds.) (1999). Special issue on domain-specific languages. *IEEE Transactions on Software Engineering*, *25*(3), May/June.

[WOLF & REIMER, 1996]

M. Wolf and U. Reimer (Eds.). *First International Conference on Practical Aspects of Knowledge Management (PAKM '96)*, Basel, Switzerland.

[WOLF, 1999]

T. Wolf (1999). Wissensmanagement in Theorie und Praxis. *Diplomarbeit*, Institut AIFB, University of Kaiserslautern.

[WOLF & DECKER & ABECKER, 1999]

T. Wolf, S. Decker, and A. Abecker (1999). Unterstützung des Wissensmanagements durch Informations- und Kommunikationstechnologie. In: *Proc. WI'99 - 4. Int. Tagung Wirtschaftsinformatik*, Saarbrücken, Physica-Verlag, Heidelberg.

[WONDERGEM, 1996A]

B. C. M. Wondergem, P. van Bommel, T.W.C. Huibers, and Th.P. van der Weide (1996). Agents in Cyberspace: Towards a Framework for Multi-Agent Systems in Information Discovery, *Technical Report*, University of Nijmegen, The Netherlands.

[YANG & KIFER, 2000]

G. Yang and M. Kifer (2000). FLORA: Implementing an Efficient DOOD System Using a Tabling Logic Engine. In: *6th International Conference on Rules and Objects in Databases (DOOD)*.

[ZLOOF, 1977]

M. Zloof (1977). Query-by-example: a data base language. *IBM Systems Journal*, *16*(4), 324--343.

Appendix A EBNF Syntax of SiLRI's F-Logic Variant

```
START          ←  {(Query | Rule | DoubleRule | Fact) }* "EOF.
Query          ←  ["FORALL" VarList ] ( "<-" | "?-") Formula ) ".".
Rule           ←  ["FORALL" VarList] MoleculeConjunction ( "<-" | ":-") Formula ".".
DoubleRule     ←  ["FORALL" VarList] MoleculeConj "<->" MoleculeConj ".".
Fact           ←  ["FORALL" VarList] MoleculeConj ".".
MoleculeConj   ←Molecule {("AND" | ",")  Molecule }*.
Formula        ←  ("EXISTS"|"FORALL") VarList Formula.
               |  Formula ("AND" | "OR" | "<-" | "<->" | "->") Formula.
               |  "NOT" Formula .
               |  "(" Formula ")" |
                  Molecule .
VarList        ←  Identifier {"," Identifier}*.
Molecule       ←  FMolecule | PMolecule.
FMolecule      ←  Reference Specification.
Reference      ←  Object ([Specification] ("#" | "##") MethodApplication )*.
Specification  ←  (( ":" | "::" | "<:") Object ["[" [ ListOfMethods] "]" ])
               |  "[" [m=ListOfMethods] "]" .
PMolecule      ←  Identifier [ "(" ListOfPaths ")"  ] .
Path           ←  Object [Specification] ( ("#" | "##") MethodApplication [ Specification ]) *.
ListOfPaths    ←  Path ("," Path)*.
Object         ←ID_Term | "(" Path ")".
Method         ←  MethodApplication MethodResult .
ListOfMethods  ←  Method (";" Method )*.
MethodApplication ← Object [ "@(" ListOfPaths")" ].
MethodResult   ←  (("->" | "=>" | "*->" | "->>" | "=>>" | "*->>") Path).
               |   ( "->>" | "=>>" | "*->>") "{" ListOfPaths "}".
IDTerm         ←  IDENTIFIER [ "(" ListOfPaths ")"  ]
               |  INTEGER_LITERAL
               |  FLOATING_POINT_LITERAL
               |  STRING_LITERAL
               |  "["  [ListOfPaths [ "|" Path ]] "]".
```

Appendix B Lloyd-Topor Transformation Algorithm

The section defines the author's variant of the Lloyd-Topor transformation in the style of [Lloyd, 1987]. As a second step, an easy implementable algorithm is provided that computes the Lloyd-Topor Transformation in a procedural way.

(a). Replace $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge \neg(V \wedge W) \wedge W_{i+1} \wedge \ldots \wedge W_m$
by $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge (\neg V \vee \neg W) \wedge W_{i+1} \wedge \ldots \wedge W_m$

(b). Replace $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge \forall x_1, \ldots, x_n W \wedge W_{i+1} \wedge \ldots \wedge W_m$
by $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge \neg\exists x_1, \ldots, x_n \neg W \wedge W_{i+1} \wedge \ldots \wedge W_m$

(c). Replace $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge \neg\forall x_1, \ldots, x_n W \wedge W_{i+1} \wedge \ldots \wedge W_m$
by $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge \exists x_1, \ldots, x_n \neg W \wedge W_{i+1} \wedge \ldots \wedge W_m$

(d). Replace $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge (V \leftarrow W) \wedge W_{i+1} \wedge \ldots \wedge W_m$
by $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge (V \vee \neg W) \wedge W_{i+1} \wedge \ldots \wedge W_m$

(e). Replace $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge \neg(V \leftarrow W) \wedge W_{i+1} \wedge \ldots \wedge W_m$
by $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge W \wedge \neg V \wedge W_{i+1} \wedge \ldots \wedge W_m$

(f). Replace $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge (V_1 \vee \ldots \vee V_o) \wedge W_{i+1} \wedge \ldots \wedge W_m$
by $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge p(x_1, \ldots, x_n) \wedge W_{i+1} \wedge \ldots \wedge W_m$

$$p(x_1, \ldots, x_n) \leftarrow V_1$$
$$\ldots$$
$$p(x_1, \ldots, x_n) \leftarrow V_o$$

where p is a new, previously unused predicate symbol and the $x_1, \ldots, x_n$ are the free variables that occur in $(V_1 \vee \ldots \vee V_m)$.

(g). Replace $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge \neg(V \vee W) \wedge W_{i+1} \wedge \ldots \wedge W_m$

by $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge \neg V \wedge \neg W \wedge W_{i+1} \wedge \ldots \wedge W_m$

(h). Replace $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge \neg\neg W \wedge W_{i+1} \wedge \ldots \wedge W_m$

by $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge W \wedge W_{i+1} \wedge \ldots \wedge W_m$

(i). Replace $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge \exists x_1, \ldots, x_n W \wedge W_{i+1} \wedge \ldots \wedge W_m$

by $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge W \wedge W_{i+1} \wedge \ldots \wedge W_m$

(j). Replace $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge \neg\exists x_1, \ldots, x_n W \wedge W_{i+1} \wedge \ldots \wedge W_m$

by $A \leftarrow W_1 \wedge \ldots \wedge W_{i-1} \wedge \neg p(y_1, \ldots, y_k) \wedge W_{i+1} \wedge \ldots \wedge W_m$

$p(y_1, \ldots, y_k) \leftarrow \exists x_1, \ldots, x_n W$

where p is a new, previously unused predicate symbol and the $y_1, \ldots, y_k$ are the free variables in $\exists x_1, \ldots, x_n W$.

Proposition B.1
Let P be a general logic program. Then the process of continually applying transformations (a),...,(j) to P terminates after a finite number of steps and results in a normal logic program (called the normal form of P.

Proof
The same construction as in [Lloyd, 1987] proposition 18.2 applies.

The following program realizes a procedural way to perform the above Lloyd-Topor Transformation.

```
translateFormula(ruleHead,formula)
    rules = []
    <disjunctions,todo> = getDisjunctions(+, formula)
    for all disj in disjunctions
        <conjunctions,todo1> = getConjunctions(+,disj)
        todo = todo + todo1
        ruleBody = ""
        for all conj in conjunctions
            if conj isa Literal
                ruleBody = ruleBody ∧ conj
            else
                let x1,...,xn be the free variables in conj
                let p be a new, unused predicate symbol
                todo = todo + [<p(x1,...,xn), conj>]
                ruleBody = ruleBody ∧ p(x1,...,xn)
```

```
        rules = rules + <ruleHead <- ruleBody>
    for all <newRuleHead, newFormula> in todo
        rules = rules + translateFormula(newRuleHead,newFormula)
    return rules
end

getConjunction(pol, formula){
    if(formula = not F )
        return getConjunction(pol^-1, F)
    else if(pol = + & formula = F1 AND F2)
        <conj1, todo1> = getConjunction(+, F1)
        <conj2, todo2> = getConjunction(+, F2)
        return <conj1 + conj2, todo1 + todo2>
    else if(pol = - & formula = F1 OR F2)
        <conj1, todo1> = getConjunction(-, F1)
        <conj2, todo2> = getConjunction(-, F2)
        return <conj1 + conj2, todo1 + todo2>
    else if(pol = - & formula = F1 <- F2)
        <conj1, todo1> = getConjunction(+, F2)
        <conj2, todo2> = getConjunction(-, F1)
        return <conj1 + conj2, todo1 + todo2>
    else if(pol = - & formula = F1 -> F2)
        <conj1, todo1> = getConjunction(+, F1)
        <conj2, todo2> = getConjunction(-, F2)
        return <conj1 + conj2, todo1 + todo2>
    else if(pol = + & formula = F1 <-> F2)
        <conj1, todo1> = getConjunction(+, F1 <- F2)
        <conj2, todo2> = getConjunction(+, F1 -> F2)
        return <conj1 + conj2, todo1 + todo2>
    else if(pol = - & formula = EXISTS x1,...,xn F )
        let y1,...,ym be the free variables in F
        let p be a new, unused predicate symbol
        return <NOT p(y1,...,ym), [< p(y1,...,ym)<- F)]>
    else if(pol = + & formula = EXISTS x1,...,xn F )
        return getConjunction( + , F )
    else if(formula = FORALL x1,...,xn F )
        return getConjunction( pol^-1 , NOT EXISTS x1,...,xn F)
    else if(pol = +)
        return <formula, []>;
    else if(pol = -)
        return <NOT formula, []>;
end

getDisjunction(pol, formula){
    if(formula = not F )
        return getDisjunction(pol^-1, F)
    else if(pol = - & formula = F1 AND F2)
```

<$disj_1$, $todo_1$> = *getDisjunction*(-, F_1)
<$disj_2$, $todo_2$> = *getDisjunction*(-, F_2)
return <$conj_1$ + $conj_2$, $todo_1$ + $todo_2$>
else if(*pol* = + & *formula* = F_1 **OR** F_2)
<$disj_1$, $todo_1$> = *getDisjunction*(+, F_1)
<$disj_2$, $todo_2$> = *getDisjunction*(+, F_2)
return <$conj_1$ + $conj_2$, $todo_1$ + $todo_2$>
else if(*pol* = + & *formula* = F_1 <- F_2)
<$disj_1$, $todo_1$> = *getDisjunction*(-, F_1)
<$disj_2$, $todo_2$> = *getDisjunction*(+, F_2)
return <$conj_1$ + $conj_2$, $todo_1$ + $todo_2$>
else if(*pol* = + & *formula* = F_1 -> F_2)
<$disj_1$, $todo_1$> = *getDisjunction*(- , F_1)
<$disj_2$, $todo_2$> = *getDisjunction*(+, F_2)
return <$conj_1$ + $conj_2$, $todo_1$ + $todo_2$>
else if(*pol* = - & *formula* = F_1 <-> F_2)
<$disj_1$, $todo_1$> = *getDisjunction*(-, F_1 <- F_2)
<$disj_2$, $todo_2$> = *getDisjunction*(-, F_1 -> F_2)
return <$conj_1$ + $conj_2$, $todo_1$ + $todo_2$>
else if(*pol* = - & *formula* = **EXISTS** x_1,...,x_n F)
let y_1,...,y_m be the free variables in F
let p be a new, unused predicate symbol
return <**NOT** $p(y_1,...,y_m)$, [< $p(y_1,...,y_m)$<- F)]>
else if(*pol* = + & *formula* = **EXISTS** x_1,...,x_n F)
return *getDisjunction*(+ , F)
else if(*formula* = **FORALL** x_1,...,x_n F)
return *getDisjunction*(pol^{-1} , **NOT EXISTS** x_1,...,x_n F)
else if(*pol* = +)
return <*formula*, []>;
else if(*pol* = -)
return <**NOT** *formula*, []>;
end

Proposition B.2
Let P be a general logic program. Then the process of continually applying transformations (a),...,(j) to P and the above sketched program results in the same transformation (modular predicate symbol renamings).

Proof

The proof is lengthy but uninspiring. The idea is to use an induction over the length of the input formula. For atomic formulae (and negated atomic formulae) the proposition is easy to verify. What remains to be shown is the proposition of non-atomic formulae of length n, under the assumption that for formulae of length n-1 the proposition is true.

Appendix C EBNF Syntax for XAL

Onto-attribute::=Ontology ':' 'ONTO' Eq Ontoinfos.
Ontology ::=NCName
NCName ::=(Letter | '_') (NCNameChar)*\
To describe 'NCNameChar' see 'Namespaces in XML'.
Ontoinfos ::=' " ' (Ontoinfo)* ' " '
Ontoinfo ::=Tag_annot | Struct_annot | Relation | Declaration
Tag_annot ::=Host (':' 'Class')? ('[' (Attval_pair)* ']')?
Host ::=(web_resource | string_shortcut) (Regular_expr)?
Web_resource::=(URI | URI_shortcut)
Regular_Expr::='(' RegEx ')'
URI_shortcut ::= ('PAGE' | 'TAG' | 'ID(' ID ')' | 'PATH(' Path ')' | Att)
ID ::=Name
Path ::=((.) (Name '[' Integer ']'))+
String_shortcut ::=(('ID(' ID ').')? ('BODY' | Att)) |
(('PATH(' Path ').')? ('BODY' | Att))
Att ::=Name
Class ::=NCName
Attval_pair ::=(Ont_att Eq Value | Ont_att '=' Skolem '[' Attval_pair ']')
Ont_att ::=Name
Value ::=(Host | String | Number)
Number ::=Converter '(' (String | String_shortcut (Regular_Expr)?) ')'
Converter ::=('FLOAT' | 'DOUBLE' | 'LONG' | 'SHORT' | 'BYTE' | ‘INTEGER' | 'BOOLEAN')
Struct_annot ::=Host_gen (':' Class)? '[' (Attval_pair_gen)+ ']'
Host_gen ::='{' (Name | Path) '}' ('.' ('BODY' | Att))? (Regular_expr)?
Attval_pair_gen ::= (Ont_att Eq Value_gen | Ont_att '=' Skolem '[' Attval_pair_gen ']')
Skolem ::='S (' Host ',' (Value)* ')'
Value_gen ::=(String_gen | Number_gen | Host)
String_gen ::=(String | Host_gen)
Number_gen ::=Converter '(' String_gen (Regular_expr)? ')'
Relation ::=Name '(' (Host | String | Number)+ ')'
Declaration ::='DECLARATION' '(' (',' Name '=' String)+ ')'

Appendix D $(KA)^2$ Ontology

```
/*****

Ontology for KA-Community
Version 0.992, February 18th, 1999

All classes in this ontology are sub-sub-classes of
the top level class Object.

*****/

/*****
Firstly, the is-a hierarchy is presented,
attribute definitions follow below.

The is-a hierachy ensures attribute inheritance
and subset relationship.

*****/

Object[].

Organization :: Object.
  Enterprise :: Organization.
  University :: Organization.
  Department :: Organization.
  Institute :: Organization.
  ResearchGroup :: Organization.

Person :: Object.
  Employee :: Person.
    AcademicStaff :: Employee.
      Researcher :: AcademicStaff.
        PhDStudent :: Researcher.
      Lecturer :: AcademicStaff.
    AdministrativeStaff :: Employee.
      Secretary :: AdministrativeStaff.
      TechnicalStaff :: AdministrativeStaff.
  Student :: Person.
    PhDStudent :: Student.

Publication :: Object.
  Book :: Publication.
  Article :: Publication.
    TechnicalReport :: Article.
    JournalArticle :: Article.
    ArticleInBook :: Article.
    ConferencePaper :: Article.
    WorkshopPaper :: Article.
  Journal:: Publication.
    SpecialIssue :: Journal.
  OnlinePublication :: Publication.

Project :: Object.
  ResearchProject :: Project.
  DevelopmentProject :: Project.
    SoftwareProject :: DevelopmentProject.

Event :: Object.
  Conference :: Event.
  Workshop :: Event.
  Activity :: Event.
  SpecialIssue :: Event.
  Meeting :: Event.
  ResearchTopic :: Object.

Product :: Object.


/*****
Now the attribute definitions were added.
*****/


Object[keyword =>> STRING].


/*****
Organization
*****/

Organization[
name =>> STRING;
location =>> STRING;
employs =>> Person;
publishes =>> Publication;
technicalReport =>> TechnicalReport;
carriesOut =>> Project;
develops =>> Product;
finances =>> Project].

University[
student =>> Student;
hasParts =>> Department].

Department[
hasParts =>> Institute].

Institute[
hasParts =>> ResearchGroup].

ResearchGroup[
member =>> Researcher;
head =>> Employee].


/*****
Person
*****/

Person[
photo =>> STRING;
firstName =>> STRING;
middleInitial =>> STRING;
lastName =>> STRING;
name =>> STRING;
address =>> STRING;
email =>> STRING;
phone =>> STRING;
fax =>> STRING].

Employee[
affiliation =>> Organization;
worksAtProject =>> Project;
headOf =>> Project;
headOfGroup =>> ResearchGroup].
```

```
AcademicStaff[
supervises =>> PhDStudent;
publication =>> Publication;
organizerOrChairOf =>> Event;
memberOfPC =>> Event;
editor =>> Publication].

Researcher[
researchInterest =>> ResearchTopic;
memberOf =>> ResearchGroup;
cooperatesWith =>> Researcher].

Secretary[
secretaryOf =>> ResearchGroup].

Student[studiesAt =>> University].

PhDStudent[
supervisor =>> AcademicStaff].
// Multiple inheritance from researcher and from student


/*****
Publication
*****/

Publication[
author =>> Person;
title =>> STRING;
year =>> NUMBER;
abstract =>> STRING;
onlineVersion =>> OnlinePublication;
describesProject =>> Project].

Book[
publisher =>> Organization;
editor =>> Person;
containsArticle =>> ArticleInBook].

TechnicalReport[
series =>> STRING;
number =>> NUMBER;
organization =>> Organization].

JournalArticle[
journal =>> Journal;
firstPage =>> NUMBER;
lastPage =>> NUMBER].

ArticleInBook[
book =>> Book;
firstPage =>> NUMBER;
lastPage =>> NUMBER].

ConferencePaper[
conference =>> Conference;
proceedingsTitle =>> STRING;
firstPage =>> NUMBER;
lastPage =>> NUMBER].

WorkshopPaper[
workshop =>> Workshop;
proceedingsTitle =>> STRING;
firstPage =>> NUMBER;
lastPage =>> NUMBER].

Journal[
editor =>> Person;
publisher =>> Organization;
volume =>> NUMBER;
number =>> NUMBER;
containsArticle =>> JournalArticle].

OnlinePublication[
onlineVersionOf =>> Publication;
type =>> STRING].


/*****
Project
*****/

Project[
title =>> STRING;
member =>> Employee;
head =>> Employee;
isAbout =>> ResearchTopic;
projectInfo =>> Publication;
carriedOutBy =>> Organization;
product =>> Product;
financedBy =>> Organization].


/*****
Event
*****/

Event[
eventTitle =>> STRING;
location =>> STRING;
date =>> STRING;
programCommittee =>> Person;
orgCommittee =>> Person;
publication =>> Publication;
hasParts =>> Event;
atEvent =>> Event].

Conference[
series =>> STRING;
number =>> NUMBER].

Workshop[
series =>> STRING;
number =>> NUMBER].


Meeting[
participant =>> Person;
date =>> STRING;
title =>> STRING].


/*****
ResearchTopic
*****/

ResearchTopic[
Name =>> STRING;
Description =>> STRING;
Approaches =>> STRING;
ResearchGroups =>> ResearchGroup;
Researchers =>> Researcher;
```

```
RelatedTopics =>> ResearchTopic;
SubTopics =>> ResearchTopic;
Events =>> Event;
Journals =>> Journal;
Projects =>> Project;
ApplicationAreas =>> STRING;
Products =>> Product;
Bibliographies =>> Publication;
MailingLists =>> STRING;
Webpages =>> STRING;
InternationalFundingAgencies =>> Organization;
NationalFundingAgencies =>> Organization;
AuthorOfOntology =>> Researcher;
DateOfLastModification =>> STRING].

/*****
Product
*****/

Product[
producedBy =>> Project;
developedBy =>> Organization;
productName =>> STRING;
ProductPublication =>> Publication;
ProductFAQ =>> STRING;
ProductMailingList =>> STRING].


/*****
These inference rules copy values from one object to
another, i.e. defines the symmetric attributes.
*****/

FORALL O,C,A,V,T V:T <- C[A=>>T] AND O:C[A->>V].


/*****  Organization-Rules  *****/

FORALL Person1, Organization1
Organization1:Organization[employs ->> Person1] <->
Person1:Employee[affiliation ->> Organization1].

FORALL Publication1, Organization1
Organization1:Organization[publishes ->> Publication1] <->
Publication1:Publication[publisher ->> Organization1].
// actually Publication should be union(Book, Journal)

FORALL TechnicalReport1, Organization1
Organization1:Organization[technicalReport ->>
TechnicalReport1] <->
TechnicalReport1:TechnicalReport[organization ->>
Organization1].

FORALL Organization1, Project1
Organization1:Organization[carriesOut ->> Project1] <->
Project1:Project[carriedOutBy ->> Organization1].

FORALL Organization1, Product1
Organization1:Organization[develops ->> Product1] <->
Product1:Product[developedBy ->> Organization1].

FORALL Person1, Group1
Group1:ResearchGroup[member ->> Person1] <->
Person1:Researcher[memberOf ->> Group1].

FORALL Employee1, Group1
    Employee1:Employee[headOfGroup ->> Group1] <->
    Group1:ResearchGroup[head ->> Employee1].


/*****  Person-Rules not allready listed above *****/

FORALL Person1, Person2
Person1:Researcher[cooperatesWith ->> Person2] <-
Person2:Researcher[cooperatesWith ->> Person1].

FORALL Person1, Publication1
Publication1:Publication[author ->> Person1] <->
Person1:Person[publication ->> Publication1].

FORALL Person1, Event1
Event1:Event[orgCommittee ->> Person1] <->
Person1:Person[organizerOrChairOf ->> Event1].

FORALL Person1, Event1
Event1:Event[programCommittee ->> Person1] <->
Person1:Person[memberOfPC ->> Event1].

FORALL Person1, Publication1
Publication1:Publication[editor ->> Person1] <->
Person1:Person[editor ->> Publication1].
//actually Publication should be union(book, journal)

FORALL Employee1, Project1
Project1:Project[member ->> Employee1] <->
Employee1:Employee[worksAtProject ->> Project1].

FORALL Employee1, Project1
Project1:Project[head ->> Employee1] <->
Employee1:Employee[headOf ->> Project1].

FORALL Person1, Person2
Person1:PhDStudent[supervisor ->> Person2] <->
Person2:AcademicStaff[supervises ->> Person1].

FORALL Person1, Topic1
Topic1:ResearchTopic[isWorkedOnBy ->> Person1] <->
Person1:Researcher[researchInterest ->> Topic1].

FORALL Person1, University1
University1:University[student ->> Person1] <->
Person1:Student[studiesAt ->> University1].


/*****  Publication-Rules not allready listed above *****/

FORALL Publication1, Publication2
Publication2:OnlinePublication[onlineVersionOf ->>
Publication1] <->
Publication1:Publication[onlineVersion ->> Publication2].

FORALL Publication1, Project1
Project1:Project[projectInfo ->> Publication1] <->
Publication1:Publication[describesProject ->> Project1].

FORALL Publication1, Publication2
Publication2:Book[containsArticle ->> Publication1] <->
Publication1:ArtcileInBook[book ->> Publication2].

FORALL Publication1, Publication2
Publication2:Journal[containsArticle ->> Publication1] <->
Publication1:JournalArticle[journal ->> Publication2].

FORALL Publication1, Event1
```

```
Event1:Conference[publication ->> Publication1] <->
Publication1:ConferencePaper[conference ->> Event1].
// publication in Conference is inherited

FORALL Publication1, Event1
Event1:Workshop[publication ->> Publication1] <->
Publication1:WorkshopPaper[workshop ->> Event1].
// publication in Workshop is inherited


/*****  Project-Rules not allready listed above *****/

FORALL Topic1, Project1
Project1:Project[isAbout ->> Topic1] <->
Topic1:ResearchTopic[dealtWithIn ->> Project1].

FORALL Product1, Project1
Project1:Project[product ->> Product1] <->
Product1:Product[producedBy ->> Project1].

FORALL Org1, Project1
Project1:Project[financedBy ->> Org1] <->
Org1:Organization[finances ->> Project1].


/*****  Event-Rules not allready listed above *****/

FORALL Event1, Event2
Event1:Event[hasParts ->> Event2] <->
Event2:Event[atEvent ->> Event1].




/
**********************************************************
***************
KA specific parts of the ontology
**********************************************************
***************/

ResearchTopic :: Object.
  KAthroughMachineLearning :: ResearchTopic.
    Abduction :: KAthroughMachineLearning.
    CaseBaseReasoning :: KAthroughMachineLearning .
        CooperativeKnowledgeAcquisition  ::  KAthroughMachine-
Learning .
    KnowledgeBasedRefinement :: KAthroughMachineLearning .
    KnowledgeDiscoveryInDatasets :: KAthroughMachineLearning
.
    DataMining :: KAthroughMachineLearning .
    LearningApprenticeSystems :: KAthroughMachineLearning .
    ReinforcementLearning :: KAthroughMachineLearninghrough-
MachineLearning .
  Reuse :: ResearchTopic.
    Ontologies :: Reuse.
      TheoreticalFoundations :: Ontologies.
      SoftwareApplications :: Ontologies.
      Methodologies :: Ontologies.
    PSMs :: Reuse.
      PSMevaluation :: PSMs.
      PSMlibraries :: PSMs.
      PSMnotations :: PSMs.
      AutomatedPSMgeneration :: PSMs.
      SysiphusIIIExperiment :: PSMs.
      WebMediatedPSMSelection :: PSMs.
    SoftwareReuse :: Reuse.
  SpecificationLanguages :: ResearchTopic.
    SpecificationMethodology :: SpecificationLanguages.
    SpecificationOfControlKnowledge :: SpecificationLanguages.
    SupportToolsForFormalMethods :: SpecificationLanguages.
     AutomatedCodeGenerationFromSpecification :: Specification-
Languages.
    ExecutableSpecificationLanguages :: SpecificationLanguages.
  ValidationAndVerification :: ResearchTopic.
    AnomalyDetection :: Validation_And_Verification.
                    AnomalyRepairAndKnowledgeRevision      ::
Validation_And_Verification.
    Formalisms :: Validation_And_Verification.
    Methodology :: Validation_And_Verification.
    V_And_V_Of_MAS :: Validation_And_Verification.
  KnowledgeManagement :: ResearchTopic.
  KAMethodologies :: ResearchTopic.
  EvaluationOfKA :: ResearchTopic.
  KnowledgeElicitation :: ResearchTopic.



Product :: Object.
  KAMethodology :: Product.
    Guideline :: KAMethodology.
    PaperLibrary :: KAMethodology.
    Ontology :: KAMethodology.
    PSM :: KAMethodology.
    ModelingLanguage :: KAMethodology.
    SpecificationLanguage :: KAMethodology.
  ComputerSupport :: Product.
    Editor :: ComputerSupport.
    TransformationTool :: ComputerSupport.
    PSMLibrary :: ComputerSupport.
    OntologyLibrary ::ComputerSupport.
    Validator :: ComputerSupport.
    Verifier :: ComputerSupport.
    ImplementationEnvironment :: ComputerSupport.
    ElicitationTool :: ComputerSupport.
    InternetTool :: ComputerSupport.
    NLParser :: ComputerSupport.


/*****
The subconcepts of Journal, i.e. IJHCS, DKE, and IEEE_Expert
contain each issue of the corresponding journal as an instance.
*****/

IJHCS :: Journal[
title ->> {"International Journal of Human Computer Studies"}] .
DKE :: Journal[
title ->> {"Data and Knowledge Engineering"}] .
IEEE_Expert :: Journal[
title ->> {"IEEE Expert"}].

SEKE :: Conference[
series ->> {"SEKE"};
eventTitle ->> {"Intl. Conference on Software and Knowledge En-
gineering"}].

KAW :: Workshop[
series ->> {"KAW"};
location ->> {"Banff, Canada"};
eventTitle ->> {"Workshop on Knowledge Acquisition, Modeling,
and Management"}].
EKAW :: Workshop[
series ->> {"EKAW"};
eventTitle ->> {"European Workshop on Knowledge Acquisition,
```

Modeling, and Management"}].
KEML :: Workshop[
series ->> {"KEML"};
eventTitle ->> {"Knowledge Engineering: Methods _And_ Languages"}].