Knowledge Engineering: Survey and Future Directions

Rudi Studer[1], Dieter Fensel[1], Stefan Decker[1], and V. Richard Benjamins[2]
[1]Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
{studer, decker, fensel}@aifb.uni-karlsruhe.de
http://www.aifb.uni-karlsruhe.de

[2]Dept. of Social Science Informatics (SWI),
University of Amsterdam, Roetersstraat 15, 1018 WB Amsterdam, The Netherlands
richard@swi.psy.uva.nl, http://www.swi.psy.uva.nl/usr/richard/home.html

Abstract. This paper provides an overview of important developments in the field of Knowledge Engineering. We discuss the paradigm shift from a transfer to a modeling approach and discuss two prominent methodological achievements: problem-solving methods and ontologies. To illustrate these and additional concepts we outline several modeling frameworks: CommonKADS, MIKE, PROTÉGÉ-II, and D3. We also discuss two fields which have emerged in the last few years and are promising areas for applying and further developing concepts and methods from Knowledge Engineering: Intelligent Information Integration and Knowledge Management.

1 Introduction

In its early days, research in Artificial Intelligence was focused on the development of formalisms, inference mechanisms and tools, for operationalizing Knowledge-based Systems (KBS). Typically, the development efforts were restricted to the realization of small KBSs in order to study the feasibility of new approaches.

Although these studies brought forth rather promising results, the transfer of this technology for building commercial KBSs failed in many cases. Just as the software crisis resulted in the establishment of the discipline Software Engineering, the unsatisfactory situation in the construction of KBSs made clear the need for more methodological approaches. Thus the goal of Knowledge Engineering (KE) is similar to that of Software Engineering: constructing KBSs in a systematic and controlable manner. This requires an analysis of the building and maintenance process itself and the development of appropriate methods, languages, and tools suitable for developing KBSs.

Subsequently, we will first give an overview of some important achievements in KE: we discuss the paradigm shift from the so-called *transfer approach* to the so-called *modeling approach*. This paradigm shift is occasionally also considered as the transfer from first generation expert systems to *second generation expert systems* [22]. In addition, we will discuss the notion of *problem-solving methods* and *ontologies*. In Section 3 we will present some modeling frameworks which have been developed in recent years: D3 [78], CommonKADS [81], MIKE [4], and PROTÈGÈ-II [77]. This section concludes with a brief description of the IBROW^3 framework for configuring problem-solvers on the Web. In Section 4 and 5 we will discuss two new fields that have emerged during the last years: Intelligent Information Integration and Knowledge

Management. Both fields provide promising perspectives for the future development of KE methods.

In KE much effort has also been invested in developing methods and supporting tools for knowledge elicitation (compare [29]). In the VITAL approach [82], e.g., a collection of elicitation tools, like repertory grids (see [43]), are offered for supporting the elicitation of domain knowledge. However, a discussion of the various elicitation methods is beyond the scope of this paper. A more detailed description of some concepts and approaches discussed in this paper may be found in [85].

2 Achievements

In this section we will discuss some major concepts which were developed in the KE field in the last fifteen years. We will first outline the paradigm shift from the transfer to the modeling approach and then discuss two fundamental concepts in that modeling framework: Problem-Solving Methods and Ontologies.

2.1 Knowledge Engineering as a Modeling Process

In the early 1980s the development of a KBS was seen as a *process* of transfering 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 [67]. Typically, this knowledge was implemented in some type of production rules which were executed by an associated rule interpreter.

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 [18]. 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 development of small prototypical systems, but it failed to produce large, reliable and maintainable knowledge bases. Furthermore, it was recognized that the assumption of the transfer approach, that is that knowledge acquisition is the collection of already existing knowledge elements, was false 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 *Knowledge Level* notion [70]. This knowledge level proposes the modeling of knowledge independent from its implementation and to structure 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 [20].

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 the further acquisition of knowledge.
- The modeling process is dependent on the subjective interpretations of the knowledge engineer. Therefore this process is typically *faulty* and an evaluation of the model with respect to reality is indispensable for the creation of an adequate model.

2.2 Problem-Solving Methods

Originally, KBSs used simple and generic inference mechanisms to infer outputs for provided cases. The knowledge was assumed to be given “declaratively“ by a set of Horn clauses, production rules, or frames. Inference engines like unification, forward or backward resolution, and inheritance dealt with the dynamic part of deriving new information. However, human experts use knowledge about the dynamics of the problem-solving *process* and such knowledge is required to enable problem-solving in practice and not only in principle [40]. [18] provided several examples where knowledge engineers implicitly encoded control knowledge by ordering production rules and premises of these rules which together with the generic inference engine, delivered the desired dynamic behaviour. Making this knowledge explicit and regarding it as an important part of the entire knowledge contained by a KBS, is the rationale that underlies *Problem-Solving Methods (PSMs)*. PSMs refine the generic inference engines mentioned above to allow a more direct control of the reasoning process. PSMs describe this control knowledge independent from the application domain, enabling reuse of this strategic knowledge for different domains and applications. Finally, PSMs abstract from a specific representation formalism as opposed to the general inference engines that rely on a specific representation of the knowledge. In the meantime, a large number of such PSMs have been developed and libraries of such methods provide support in their reuse for constructing new applications.

Reuse is a promising way to reduce development costs of software and knowledge-based systems. The basic idea is that a KBS can be constructed from ready-made parts instead of being built up from scratch. Research on PSMs has adopted this philosophy and several PSM libraries have been developed. In the following, we sketch several issues involved in developing such libraries:

- There are currently several libraries of PSMs. They all aim at facilitating the knowledge-engineering process, yet they differ in various ways. In particular, libraries differ along dimensions such as universality, formality, granularity, and size. The type of a library is determined by its characterization in terms of these above dimensions. Each type has a specific role in the knowledge engineering process and has strong and weak points.
- There are several alternatives for organizing a library and each of them has

consequences for indexing PSMs and for their selection. Determining the ‘best’ organizational principle for such libraries is still an issue of debate.

- Whatever the organizational structure of the library, PSMs are used to realize tasks (tasks describe the *what*, PSMs describe the *how*) by applying domain knowledge. Therefore, there are two possible reasons why a PSM cannot be applied to solve a particular problem: (1) if its requirements on domain knowledge are not fulfilled, or (2) if it cannot deliver what the task requires, that is, if its competence or functionality is not sufficient for the task. Methods for weakening and strengthening PSMs are discussed in [87].
- Traditionally, PSMs are described in an operational style. They are described as decomposing a task into a set of subtasks by introducing their dataflows and knowledge roles and by defining some control on how to execute the subtasks. However, these are not the most important aspects from the standpoint of reuse. As mentioned earlier, two main aspects decide about the applicability of a PSM in a given application: whether the competence of the method is able to achieve the goal of the task and whether the domain knowledge required by the method is available. [6] discussed the characterizations of PSMs by their functionality where the functionality is defined in terms of assumptions over available domain knowledge. Meanwhile several papers have appeared providing declarative characterizations of PSMs (e.g. [9], [39], [21]) and thus dealing with an important line of future work on PSMs.

Offering brokering services for accesing Web-based PSM libraries and configuring KBSs from these libraries is the goal of the IBROW^3 project [10] (cf. Section 3.5).

2.3 Ontologies

Since the beginning of the 1990s ontologies have become a popular research topic and have been investigated by several Artificial Intelligence research communities, including KE, natural-language processing and knowledge representation. More recently, the notion of ontology is also becoming widespread in fields such as intelligent information integration, intelligent information retrieval on the Internet, and knowledge management (see Sections 4 and 5). The reason that ontologies have become so popular is in a large part due to what they promise: a shared and common understanding of a domain that can be communicated across people and computers.

Many definitions of ontologies have been given in the last decade, but one that, in our opinion, best characterizes the essence of an ontology is based on the definition in [50]: *An ontology is a formal, explicit specification of a shared conceptualization.* A ‘conceptualization’ refers to an abstract model of some phenomenon in the world by identifying the relevant concepts of that phenomenon. ‘Explicit’ means that the type of concepts used and the constraints on their use are explicitly defined. ‘Formal’ refers to the fact that the ontology should be machine readable, which excludes natural language. ‘Shared’ reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group. Basically, the role of ontologies in the knowledge engineering process is to facilitate the construction of a domain model. An ontology provides a vocabulary of terms and relations with which a domain can be modeled.

Especially because ontologies aim at consensual domain knowledge, their development is often a cooperative process involving different people, possibly at different locations. People who agree to accept an ontology are said to *commit* themselves to that ontology.

Depending on their generality level, different types of ontologies that fulfil different roles in the process of building a KBS can be identified ([89], [53]). Among others, we can distinguish the following ontology types:

- *Domain ontologies* capture the knowledge valid for a particular type of domain (e.g. electronic, medical, mechanic, digital domain).
- *Generic or commonsense ontologies* aim at capturing general knowledge about the world and provide basic notions and concepts for things like time, space, state, event etc. ([75], [42]). As a consequence, they are valid across several domains. For example, an ontology about mereology (part-of relations) is applicable in many technical domains [12].
- *Representational ontologies* do not commit themselves to any particular domain. Such ontologies provide representational entities without stating what should be represented. A well-known representational ontology is the *Frame Ontology* [50], which defines concepts such as frames, slots and slot constraints allowing us to express knowledge in an object-oriented or frame-based way.

The ontologies mentioned above all capture static knowledge in a problem-solving independent way. KE is, however, also concerned with problem-solving knowledge, therefore the so-called *method* and *task ontologies* are also useful types of ontologies ([38], [84]). Task ontologies provide terms specific for particular tasks (e.g. ’hypothesis’ belongs to the diagnosis task ontology) and method ontologies provide terms specific to particular PSMs [48] (e.g. ‘correct state’ belongs to the Propose-and-Revise method ontology). Task and method ontologies provide a reasoning point of view on domain knowledge. In this manner, these ontologies help to solve the ‘interaction problem’ [15], which states that domain knowledge cannot be independently represented from the way in which it will be used in problem solving and vice versa. Method and task ontologies enable us to make explicit the interaction between problem-solving and domain knowledge through assumptions ([9], [40]).

Part of the research on ontology is concerned with envisioning and constructing a technology which enables the large-scale reuse of ontologies on a world-wide level. In order to enable as much reuse as possible, ontologies should be small modules with a high internal coherence and a limited amount of interaction between the modules. This requirement and others are expressed in design principles for ontologies ([51], [52], [88]).

Assuming that the world is full of well-designed modular ontologies, constructing a new ontology is a matter of assembling existing ones. In [33] the Ontolingua server is described which provides different kinds of operations for combining ontologies: inclusion, restriction, and polymorphic refinement. The inclusion of one ontology in another, e.g., has the effect that the composed ontology consists of the union of the two ontologies (their classes, relations, axioms). The SENSUS system [86] provides a means for constructing a domain specific ontology from given commonsense ontologies. The basic idea is to use so-called seed elements which represent the most important domain concepts for identifying the relevant parts of a top-level ontology. The selected parts are then used as a starting point for extending the ontology with further domain-specific concepts. Another approach for the systematic reuse of commonsense ontologies is the KARO approach (Knowledge Acquisition Environment with Reusable Ontologies) [75] which offers formal, linguistic, and graphical methods for retrieving and adapting concept definitions from a given ontology. The supplementation of graphical methods with formal and linguistic means, i.e. with classification mechanisms and natural language processing features, achieves a flexible way for reusing ontologies. The SKC project (Scalable Knowledge Composition) [55] aims at developing an algebra for systematically composing ontologies from already existing ones. It aims at offering union, intersection, and difference as basic operations.

Various kinds of formal languages are used for representing ontologies, among others description logics (see e.g. LOOM [63] or CYCL [61]), Frame Logic [56], and Ontolingua [50], which is based on KIF (Knowledge Interchange Format) [45], and is basically a first-order predicate logic extended with meta-capabilities to reason *about* relations.

2.4 Specification Approaches in Knowledge Engineering

Over the last ten years a number of specification languages have been developed for describing KBSs. These specification languages can be used to specify the knowledge required by the system as well as the reasoning process which uses this knowledge to solve the task assigned to the system. On the one hand, these languages should enable a specification which abstracts from implementation details. On the other hand, they should enable a detailed and precise specification of a KBS at a level which is beyond the scope of specifications in natural language. This area of research is quite well documented by a number of workshops and comparison papers based on these workshops. Surveys of these languages can be found in [41], [34] provides a comparison to similar approaches in software engineering. [91] provide insights as to how these languages can be applied in the broader context of *knowledge management*.

As mentioned above, we can roughly divide the development of knowledge engineering into a knowledge transfer and a knowledge modelling period. In the former period, knowledge was directly encoded using rule-based implementation languages or frame-based systems. The (implicit) assumption was that these representation formalisms are adequate to express the knowledge, reasoning, and functionality of a KBS in a way which is understandable for humans and computers. Serious difficulties arose, however [19]:

- different types of knowledge were represented uniformly,
- other types of knowledge were not represented explicitly,
- the level of detail was too high to present abstract models of the KBS,
- and knowledge level aspects were constantly mixed with aspects of the

implementation.

As a consequence, such systems were hard to construct and to maintain when they become larger or were used over a longer period of time. As a consequence, many research groups worked on more abstract description means for KBSs. Some of them were still executable (like the generic tasks [16]) whereas others combined natural language descriptions with semiformal specifications. The most prominent approach in the latter area are the KADS and CommonKADS approaches [81] that introduced a conceptual model (the *Expertise Model*) to describe KBSs at an abstract and implementation independent level. As explained below, the *Expertise Mode*l distinguishes different knowledge types (called layers) and provides different primitives for each knowledge type (for example, knowledge roles and inference actions at the inference layer) to express the knowledge in a structured manner. A semiformal specification language CML [79] arose that incorporates these structuring mechanisms in the knowledge level models of KBSs. However, the elementary primitives of each model were still defined by using natural language.

Using natural language as a device to specify computer programs has well known advantages and disadvantages. It provides freedom, richness, easiness in use and understanding, which makes it a comfortable tool in sketching what one expects from a program. However, its inherent vagueness and implicitness make it often very hard to answer questions as to whether the system really does what is expected, or whether the model is consistent or correct (cf. [54]). Formal specification techniques arose that overcome these shortcomings. Usually they were not meant as a replacement for semiformal specifications but as a possibility to improve the precision of a specification when required. Meanwhile, around twenty different approaches can be found in the literature ([41], [34]). Some of them aim mainly at formalization. A formal semantics is provided that enables the unique definition of knowledge, reasoning, or functionality along with manual or automated proofs. Other approaches aim at operationalization, that is, the specification of a system can be executed which enables prototyping in the early phase of system development. Here, the evaluation of the specification is the main interest. They help to answer the question as to whether the specification really specifies what the user is expecting or the expert is providing. Some approaches aim at formalizing and operationalizing (cf. e.g. the specification language KARL [35]), however they have to tackle conflicting requirements that arise from these two goals.

Specification languages for KBSs arose to formalize their conceptual models. They use the structuring principles of semiformal specifications and add formal semantics to the elementary primitives and their composition (cf. [27], [28]). As introduced above, the *Expertise Model* [81] describes the different types of knowledge required by a KBS as well as the role of this knowledge in the reasoning process of the KBS. Based on this, specification languages provide formal means for precisely defining:

- the goals and the process necessary to achieve them,
- the functionality of the inference actions, and
- the precise semantics of the different elements of the domain knowledge.

Definitions in natural language are supplemented by formal definitions to ensure unambiguity and preciseness. The structure of the conceptual models organizes the formal specification in a natural manner, improving understandability and simplifying the specification process.

3 Modeling Frameworks

In this section we will describe different modeling frameworks which address various aspects of model-based KE approaches: D3 [78] introduces the notion of configurable role-limiting methods, CommonKADS [81] is noted for having defined the structure of the Expertise Model, MIKE [4] puts emphasis on a formal and executable specification of the Expertise Model as the result of the knowledge acquisition phase, and PROTÉGÉ-II [31] exploits the notion of ontologies.

It should be clear that there are further approaches which are well known in the KE community, like e.g VITAL [82] and EXPECT [49]. However, a discussion of these approaches is beyond the scope of this paper.

3.1 D3: Configurable Role-Limiting Methods

Role-Limiting Methods (RLM) [64] were one of the first attempts to support the development of KBSs by exploiting the notion of a reusable problem-solving method. The RLM approach may be characterized as a shell approach. Such a shell comes with an implementation of a specific PSM and can thus only be used to solve a type of tasks for which the PSM is appropriate. The given PSM also defines the generic roles that knowledge can play during the problem-solving process and it completely determines the knowledge representation for the roles such that the expert only has to instantiate the generic concepts and relationships which are defined by these roles. Therefore, the acquisition of the required domain specific instances may be supported by (graphical) interfaces which are custom-tailored for the given PSM.

In order to overcome this inflexibility of RLMs, the concept of configurable RLMs has been proposed. *Configurable Role-Limiting Methods* (CRLMs) as discussed in [76] and implemented in D3 [78] exploit the idea that a complex PSM may be decomposed into several subtasks where each of these subtasks may be solved by different methods. In [76], various PSMs for solving classification tasks, like *Heuristic Classification* or *Set-covering Classification*, were analysed with respect to common subtasks. This analysis resulted in the identification of shared subtasks like „data abstraction” or „hypothesis generation and test”. Within the CRLM framework a predefined set of different methods are offered for solving each of these subtasks. Thus a PSM may be configured by selecting a method for each of the identified subtasks. In that way the CRLM approach provides a means for configuring the shell for different types of tasks. It should be noted that each method offered for solving a specific subtask has to meet the knowledge role specifications that are predetermined for the CRLM shell, i.e. the CRLM shell comes with a fixed scheme of knowledge types. As a consequence, the introduction of a new method into the shell typically involves the modification and/or extension of the current scheme of knowledge types [76]. Having a fixed scheme of knowledge types and predefined communication paths between the various components is an important restriction which distinguishes the CRLM framework from more flexible configuration approaches such as CommonKADS (see Section 3.2).

The D3 system places strong emphasis on supporting graphical knowledge acquisition methods that allow domain experts to build up domain models themselves [7]. D3 offers a variety of form-based or graphical editors covering the basic types of editors which are needed for knowledge acquisition:

- *Object Forms* are used to specify single objects together with their attributes and their respective values.
- *Object-Attribute Tables* are similar to relational database tables and allow entering a collection of objects which all have the same atomic attributes.
- *Object-Object Tables* are tables in which both rows and columns are labeled by objects. These tables support the specification of simple relations between objects.
- *Object-Relation Tables* are complex tables in which rows are labeled by object attributes and columns by relations. This type of table provides a means for entering relations between several attributes of an object.
- *Hierarchies* are used for entering taxonomic relations between objects or for specifying decision tables.

All these different editors are integrated in a uniform user interface supporting a flexible switching between the different editor types. Of course, grapical means cannot conveniently cope with arbitrary n-ary relations between objects. Therefore, D3 includes graphical abstractions, which visualize such complex relations, but result in some type of information loss [7].

In order to reduce the effort for implementing such graphical interfaces D3 includes META-KA (Meta Knowledge Acquisition System) which generates the required graphical editors from declarative specifications [44]. META-KA is based on the object oriented representation of the knowledge types that are used for the system internal knowledge representation. This internal representation is enriched by information concerning the layout of the tables or the navigation structure that defines, e.g., the menus and dialogue buttons which are offered to the user.

D3 was used in several application projects for developing KBSs. Examples are medical diagnosis and tutoring systems or service-support systems for printing machines (see [78] for more details).

3.2 The CommonKADS Approach

A well-known knowledge engineering approach is *KADS* [80] and its further development to *CommonKADS* [81]. A basic characteristic of KADS is the construction of a collection of models, where each model captures specific aspects of the KBS to be developed as well as its environment. In CommonKADS we distinguish the *Organization Model*, the *Task Model*, the *Agent Model*, the *Communication Model*, the *Expertise Model* and the *Design Model*. Whereas the first four models aim at modeling the organizational environment the KBS will operate in and the tasks that are performed in the organization, the expertise and design model describe (non-) functional aspects of the KBS under development.

Subsequently, we will briefly discuss each of these models and then provide a detailed description of the Expertise Model:

- Within the *Organization Model* the organizational structure is described together with a specification of the functions that are performed by each organizational unit. Furthermore, it identifies the deficiencies of the current business processes and possibilities for improving these processes by introducing KBSs.
- The *Task Model* provides a hierarchical description of the tasks which are performed in the organizational unit in which the KBS will be installed. This includes a specification of which agents are assigned to the different tasks.
- The *Agent Model* specifies the capabilities of each agent involved in the execution of the tasks at hand. In general, an agent can be a human or some kind of software system, e.g. a KBS.
- Within the *Communication Model* the various interactions between the different agents are specified. Among other things, it specifies which type of information is exchanged between the agents and which agent is initiating the interaction.

A major contribution of the KADS approach is its proposal for structuring the *Expertise Model*, which distinguishes three different types of knowledge required to solve a particular task. Basically, the three different types correspond to a static view, a functional view and a dynamic view of the KBS to be built. Each type of knowledge is modeled in a different layer of the *Expertise Model*:

- *Domain layer*: At the domain layer the domain specific knowledge needed to solve the task at hand is modeled. This includes a conceptualization of the domain in a domain ontology (see Section 2) and a declarative theory of the required domain knowledge. One objective for structuring the domain layer is to model it as reusable as possible for solving different tasks.
- *Inference layer*: At the inference layer the reasoning process of the KBS is specified by exploiting the notion of a PSM. The inference layer describes the *inference actions* of which the generic PSM is composed as well as the *roles* which are played by the domain knowledge within the PSM. The dependencies between inference actions and roles are specified in what is called an *inference structure.* Furthermore, the notion of roles provides a domain independent view of the domain knowledge.
- *Task layer*: The task layer provides a decomposition of tasks into subtasks and inference actions, including a goal specification for each task and a specification of how these goals are achieved. The task layer also provides a means for specifying the control over the subtasks and inference actions that are defined at the inference layer.

CML (Conceptual Modeling Language) [79], a semi-formal language with a graphical notation, is offered to to describe an *Expertise Model.* CML is oriented towards providing a communication basis between the knowledge engineer and the domain expert during the model construction process.

The clear separation of the domain specific knowledge from the generic description of the PSM at the inference and task layer enables, in principle, two kinds of reuse: on the one hand, a domain layer description may be reused for solving different tasks by different PSMs, on the other hand, a given PSM may be reused in a different domain by defining a new view of another domain layer. This reuse approach is a weakening of the strong interaction problem hypothesis [15]. In [81] the notion of a *relative interaction hypothesis* is defined to indicate that some kind of dependency exists between the structure of the domain knowledge and the type of task which is to be solved. To achieve a flexible adaptation of the domain layer to a new task environment, the notion of layered ontologies is proposed: *Task* and *PSM ontologies* may be defined as viewpoints of an underlying domain ontology. Within CommonKADS a library of reusable and configurable components, which can be used to build up an *Expertise Model*, has been defined [13].

In essence, the *Expertise Model* and *Communication Model* capture the functional requirements for the target system. Based on these requirements the *Design Model* is developed, which specifies among other things the system architecture and the computational mechanisms for realizing the inference actions. KADS aims at achieving a *structure-preserving design*, i.e. the structure of the *Design Model* should reflect the structure of the *Expertise Model* as much as possible [81].

All the development activities which result in a stepwise construction of the different models are embedded in a cyclic and risk-driven life cycle model similar to Boehm’s spiral model [11].

The knowledge models of CommonKADS can also be used in a Knowledge Management environment for supporting the knowledge acquisition process and for structuring the knowledge according to different knowledge types [91]. This is a nice example of how methods from KE may contribute to new fields like Knowledge Management.

3.3 The MIKE Approach

The *MIKE approach* (*Model-based and Incremental Knowledge Engineering*) (cf. [4], [5]) provides a development method for KBSs covering all steps from the initial elicitation through specification to design and implementation. MIKE proposes the integration of *semiformal* and *formal specification techniques* and *prototyping* into an engineering framework. The integration of prototyping and support for an incremental and reversible system development process into a model-based framework is actually the main distinction between MIKE and CommonKADS [81]:

- MIKE takes the *Expertise Model* of CommonKADS as its general model pattern and provides a smooth transition from a semiformal representation, the *Structure Model*, to a formal representation, the *KARL Model*, and further to an implementation oriented representation, the *Design Model*. The smooth transition between the different representation levels of the *Expertise Model* is essential for enabling incremental and reversible system development in practice.
- In MIKE the executability of the *KARL Model* enables the validation of the *Expertise Model* by means of prototyping. This considerably enhances the

integration of the expert in the development process.

In MIKE, the entire development process is divided into a number of subactivities: *Elicitation*, *Interpretation*, *Formalization/Operationalization*, *Design,* and *Implementation*. Each of these activities deals with different aspects of the system development.

The knowledge acquisition process starts with *Elicitation*. Methods like structured interviews [29] are used for acquiring informal descriptions of the knowledge about the specific domain and the problem-solving process itself. The resulting knowledge, expressed in natural language, is stored in so-called *knowledge protocols*.

During the *Interpretation* phase the knowledge structures which are identified in the *knowledge protocols* are represented in a semi-formal variant of the *Expertise Model*: the *Structure Model* [69]. All structuring information in this model, like the data dependencies between two inferences, is expressed in a fixed, restricted language while the basic building blocks, e.g. the description of an inference, are represented by unrestricted texts. This representation provides an initial structured description of the emerging knowledge structures and can be used as a communication basis between the knowledge engineer and the expert. Thus the expert can be integrated into the process of structuring the knowledge.

The *Structure Model* is the foundation for the *Formalization/Operationalization* process which results in the formal *Expertise Model*: the *KARL Model*. The *KARL Model* has the same conceptual structure as the *Structure Model* whereby the basic building blocks, which have been represented as natural language texts, are now expressed in the formal specification language *KARL* [35]. This representation avoids the vagueness and ambiguity of natural language descriptions and thus helps to obtain a clearer understanding of the entire problem-solving process. The *KARL Model* can be directly mapped to an operational representation because KARL (with some small limitations) is an executable language.

The result of the knowledge acquisition phase, the *KARL Model*, captures all functional requirements for the final KBS. During the *Design* phase additional non-functional requirements are considered [60]. These non-functional requirements include efficiency and maintainability, but also the constraints imposed by target software and hardware environments. Efficiency is already partially covered in the knowledge acquisition phase, but only to the extent that it determines the PSM. Consequently, functional decomposition is already part of the earlier phases in the development process. Therefore, the design phase in MIKE constitutes the equivalent of detailed design and unit design in software engineering approaches. The *Design Model* which is the result of this phase is expressed in the language *DesignKARL* [58]. DesignKARL extends KARL by providing additional primitives for structuring the *KARL Model* and for describing algorithms and data types. DesignKARL additionally allows the description the design process itself and the interactions between design decisions.

The *Design Model* captures all functional and non-functional requirements posed to the KBS. In the *Implementation* process the *Design Model* is implemented in the target hardware and software environment.

The result of all phases is a set of several interrelated refinement states of the *Expertise Model*. All these different model variants are explicitly connected to each other via different types of links and thus ensure traceability of (non-)functional requirements

The entire development process, i.e. the sequence of knowledge acquisition, design, and implementation, is performed in a cycle guided by a *spiral model* [11] as process model. Every cycle produces a prototype of the KBS which may be evaluated by testing it in the real target environment. The results of the evaluation are used in the next cycle to correct, modify, or extend this prototype.

The MIKE approach as described above is restricted to modeling the KBS under development. To capture the embedding of a KBS in a business environment, the MIKE approach has been extended by new models which define different views of an enterprise. Main emphasis is put on a smooth transition from business modeling to the modeling of problem-solving processes [24].

3.4 The PROTÉGÉ-II Approach

The *PROTÉGÉ-II approach* [31] aims at developing a tool set and methodology for the construction of domain-specific knowledge-acquisition tools [30] and knowledge-based systems from reusable components, i.e. PSMs and knowledge bases.

In PROTÉGÉ-II a PSM comes with a so-called *method ontology* (cf. [31], [84]): such a method ontology defines the concepts and relationships that are used by the PSM for providing its functionality. The *Board-Game Method* [31], e.g., uses among other things the notions of ‘pieces’, ‘locations’, and ‘moves’ to provide its functionality, that is to move pieces between locations on a board. In this way, a method ontology corresponds to the generic terminology as introduced by the collection of knowledge roles of a PSM (compare Section 2).

A second type of reusable components are domain knowledge bases which provide the domain specific knowledge to solve a given task. Knowledge bases are accompanied by *domain ontologies* that define the concept and relationships which are used within the domain knowledge base.

Both PSMs and domain ontologies are reusable components for constructing a KBS. However, due to the interaction problem the interdependence between domain ontologies and PSMs with their associated method ontologies has to be taken into account when constructing a KBS from reusable components. Therefore, PROTÉGÉ-II proposes the notion of separate *mediators* [90] to adapt PSMs and knowledge bases to each other [47]. In contrast to PSMs and knowledge bases, mediators are specifically for solving an application task since they are tailored towards adapting a PSM to a knowledge base to solve a particular task.

Rather recently, PROTÉGÉ-II has been embedded in a CORBA environment [73] in order to make the different components accessible across different software environments. In essence, each of the three components comes with an *interface definition* specified in the CORBA Interface Definition Language. Thus components which run on different software platforms may communicate with each other and may therefore be reused for building up a KBS.

Mediators may provide either a static or a dynamic mediation between PSMs and knowledge bases. *Static* mediation means that all the knowledge that is needed by the PSM is made available by the mediator when the first ’LoadKB’ request is issued by the PSM. Thus all further knowledge requests can be handled by the mediator without any further access to the underlying knowledge base. *Dynamic* mediation allows for access to the knowledge base at run-time. Such a flexible mediation may be needed in cases in which the kind of knowledge that is needed by the PSM is dependent on user-provided run-time inputs.

The PROTÉGÉ-II approach has a long tradition in generating knowledge-acquisition tools from ontologies [30]. For generating a knowledge acquisition tool one first has to specify an ontology, i.e the concepts and their corresponding attributes. PROTÉGÉ-II takes such an ontology as input and generates as output a knowledge-acquisition tool that allows domain specialists to enter instances of the domain concepts, i.e. domain facts. The PROTÉGÉ-II component for generating knowledge-acquisition tools is comparable to the META-KA component of D3 (see Section 3.1): META-KA offers more types of graphical editors, its degree of automation in generating a tool is, however, lower compared to PROTÉGÉ-II.

3.5 IBROW^3: An Intelligent Brokering Service for Knowledge-Component Reuse on the World-Wide Web

The World Wide Web is changing the nature of software development to a distributive plug & play process. This requires a new method for managing software by so-called *intelligent software brokers*. The European IBROW^3 project developed an intelligent brokering service that enables third party knowledge-component reuse through the WWW [10]. Suppliers provide libraries of knowledge components adhering to some standard, and customers can consult these libraries -- through intelligent brokers -- to configure a KBS suited to their needs by selection and adaptation. For achieving these goals, IBROW^3 integrates research on heterogeneous databases, interoperability, and web technology with knowledge-system technology and ontologies. The broker can handle web requests for classes of KBS by accessing libraries of reusable PSMs in the Web and by selecting, adapting, and configuring these methods in accordance with the domain at hand. The main focus of the current work is reasoning knowledge, i.e. PSMs. The development of the description language *UPML* (*Unified Problem-solving Method Description Language*) summarized a decade of research on specification languages for KBSs. The language relies on a newly developed architecture for KBSs that uses (stacks of) KBSs adapters to express component connection and refinement [36]. This architecture provides a structured way for developing, adapting, and reusing PSMs. The broker supports component selection, adaptation, and combination through a user-guided browsing process and deductive inferences that relate goal descriptions of tasks to competence descriptions of PSMs. Summing up in a nutshell, IBROW^3 integrates the results of KE in the new computational environments that are currently arising.

4 Intelligent Information Integration and Information Services

The growth of on-line information repositories (such as the Internet) has made information presentation and access much simpler. However, it has also become a cliche to observe that this growth has vastly complicated tasks involving the finding and synthesizing of precisely that information which someone is looking for. A tourist planning a trip to Paris, for example, can not simply use a Web browser today to call up a map showing the Italian restaurant closest to the Eiffel Tower. And the tourist is likely to become even more frustrated upon realizing that the Web in fact contains all the necessary information: maps of Paris, lists of restaurants and tourist sites, etc. The problems is not one of information distribution, but of information integration.

Constructing tools to simplify access to the wealth of available information constitutes a significant challenge to computer science [68]. One specific challenge to be addressed is the development of methods and tools for integrating partially incompatible information sources. In that context the notion of mediators is proposed as a middle layer between information sources and applications [90]. Among other things, mediators rely on the notion of ontologies for defining the conceptualization of the underlying information sources. Therefore, methods for constructing and reusing ontologies are directly relevant for developing mediators.

TSIMMIS [17] stands for The Stanford-IBM Manager of Multiple Information Sources. The goal of the TSIMMIS Project is to develop tools that facilitate the rapid integration of heterogeneous information sources that may include both structured and semistructured data. TSIMMIS uses a pattern-based approach to extract information from different sources. All extracted information is represented in OEM, a semi-structured data model which allows the user to cope with the varying structures found in different Web sources. Mediators that integrate information from different sources use these OEM structures as input and deliver OEM data as answers to queries.

[46] describes the *Infomaster system,* which is a generic tool for integrating different types of information sources, like e.g. relational databases or Web pages. Each information source is associated with a wrapper that hides the source specific information structure of the information sources. Internally, Infomaster uses the Knowledge Interchange Format for representing knowledge. Infomaster uses so-called base relations for mediating between the conceptual structure of the information sources and the user applications. The collection of base relations may be seen as a restricted domain ontology for integrating the different heterogeneous information sources.

Shopping agents [74] free the user from the need to search and visit several on-line stores when trying to buy a product via the WWW. At the request of the human client they visit several of these stores and use wrappers to extract the provided product information, and a mediator summarizes and integrates the results. These agents have become commercial products in the meantime and are integrated into the services of web search engines (cf. www.excite.com).

Ontologies are also used for supporting the semantic retrieval of information from the World-Wide Web. The *SHOE approach* [62] annotates Web pages with ontological information which can then be exploited for answering queries. Thus, the syntactic based retrieval of information from the Web, as is familiar from the various Web search engines, is replaced by a semantic based retrieval process. A further step is taken in the *Ontobroker project* ([37], [25]) which uses a more expressive ontology combined with a corresponding inference mechanism. Thus, the search metaphor of SHOE is replaced by an inference metaphor for retrieving information from the Web since the inference mechanism can use complex inferences as part of the query answering process.

Such systems will soon spread throughout the WWW based on newly emerging standards for information representation. The extendable Mark-up Language XML [93] provides semantic information as a by-product of defining the structure of the document. XML provides a tree structure to describe documents. The different leaves of the tree have a well-defined tag and a context through which the information can be understood. That is, structure and semantics of document are interweaved. The Metadata initiative of the W3C (the standardization committee of the WWW) developed the Resource Description Framework RDF[1] [66] which provides a means for adding semantics to a document without making any assumptions about the structure of this document. It is an XML application (i.e., its syntax is defined in XML) customized for adding meta information to Web documents. It is currently under development as a W3C standard for content descriptions of web sources and will be used by other standards like PICS-2, P3P, and DigSig. The Dublin Core [23] is a set of fifteen metadata elements intended to facilitate the discovery of electronic resources. Originally conceived for the author-generated description of Web resources, it has also attracted the attention of formal resource description communities such as museums and libraries. It corresponds to a simple ontology that fixes the content of meta-descriptions.

5 Knowledge Management and Organizational Memories

Knowledge Management (KM) receives more and more interest. Companies recognize that in the knowledge economy what organizations know is becoming more important than the traditional sources of economic power - capital, land, plants, and labor. It is important to recognize that KM is a multidisciplinary application area and that single solutions from one discipline do usually not work in a complex environment. Disciplines involved are e.g. management sciences, sociology, document management, ergonomics, computer supported cooperative work (cf. [26], [92] and Knowledge Engineering: Exploiting and protecting intellectual assets (cf. [71]) is obviously related to the aims of Knowledge Engineering. Usually, the task of KE is the „engineering“ of knowledge with the construction of a KBSs in mind. This is not necessarily true in Knowledge Management: the outcome of a Knowledge Management Strategy may not be a KBS, not even a computer-based system. Even changes in the culture of an organization can support Knowledge Management. However, from an IT-point of view an *Organizational Memory Information System (OMIS)* (cf. [83], [1], [57]) plays an important role in KM.

1. See http://www.w3c.org/RDF.

5.1 Technological Support for Knowledge Management

Knowledge Engineering provides strong support for the tasks of building an OMIS, e.g. for elicitation of the content, for building and maintenance, and for knowledge retrieval and utilization. Thus it is no wonder that many knowledge elicitation tools and techniques exist which can be used inside an OMIS, e.g. for initial acquisition of knowledge or for an incremental use during the life-cycle of an OMIS. However, as mentioned before Knowledge Elicitation tools are not the only part of an OMIS, instead many other system components have to be integrated. To support these integration tasks and to enable, e.g., ontology-based knowledge retrieval [8] the construction of three different kinds of ontologies is suggested ([1], [91]):

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

These ontologies provide a basis for interoperability at several layers and enable more sophisticated use of the knowledge in an OMIS. The construction of these ontologies is itself a knowledge engineering task and can be supported by tools and techniques.

5.2 Knowledge Engineering vs. Knowledge Management

KE can provide a basis for Knowledge Management, but it is imporant to have the following differences between an OMIS and a KBS in mind:

- A KBS focuses on the solution of a single task. 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, e.g. documents, hypertext (cf. [32]), and formal knowledge bases. Typically, informally represented knowledge is usually much more important than formally represented knowledge. In KM the final consumers of the knowledge are human beings, not the system itself.
- An OMIS integrates different kinds of knowledge (e.g. „Best Practices“ and „Experiences“ (cf. [3], [59]), „Design Rationales“ [14], Process Knowledge [65]] at different levels of representation. Because KBS aim at solving single tasks, their knowledge requirements are very homogenous.
- Groupware and knowledge dissemination techniques are usually not part of a KBS, but are essential for an OMIS because the knowledge inside the system has to be communicated to the employees. In addition an OMIS has to integrate many different preexisting *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 KE can contribute to a methodology for building an OMIS and supporting a *Knowledge Management Strategy* (cf. [24], [91]). The CommonKADS-Methodology, e.g., can help to identify and analyze a company's knowledge-intensive work processes. So we can conclude, that KE does not only provide tools and methodologies for building KBSs based on PSMs, but can also serve as to a great extend as a general framework for building an OMIS. For more details and references cf. [72].

6 Conclusion and Related Work

During the last decade, research in Knowledge Engineering resulted in several important achievements. Notable developments are:

- Within the framework of model-based Knowledge Engineering, model structures have been defined which clearly separate the different types of knowledge which are important in the context of Knowledge-based Systems (KBSs). The *Expertise Model* is the most prominent example of these models.
- The clear separation of the notions of task, problem-solving method, and domain knowledge provides a promising basis for making the reuse-oriented development of KBSs more feasible.
- The integration of a strong conceptual model is a distinctive feature of formal specification languages in Knowledge Engineering.

The notion of knowledge level models which evolved in Knowledge Engineering during the last decade is a promising general concept for modeling knowledge. Thus, other fields, like Knowledge Management or Intelligent Information Integration, that heavily rely on building up knowledge models may directly profit from these notions and the available modeling tools.

Reusing software components that are available in some kind of Web repositories becomes an interesting perspective for developing application systems. Again the Knowledge Engineering concepts of ontologies and problem-solving methods are a promising starting point for developing intelligent software brokering services. The framework of the IBROW^3 approach [10] is a first step in developing and implementing such brokering services.

Acknowledgement

Richard Benjamins was partially supported by the Netherlands Computer Science Research Foundation with financial support from the Netherlands Organisation for Scientific Research (NWO).

References

[1] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kühn, and M. Sintek. Towards a Technology for Organizational Memories. *IEEE Intelligent Systems & Their Applications*,13:(3), 1998.

[2] A. Abecker and S. Decker, Organizational Memory: Knowledge Acquisition, Integration and Retrieval Issue, in: F. Puppe, ed., *Knowledge-based Systems: Survey and Future Directions, Proc. 5th German Conference on Knowledge-based Systems*, Wuerzburg, Lecture Notes in AI, Springer Verlag, 1999.

[3] K.-D. Althoff, F. Bomarius and C. Tautz, Using Case-Based Reasoning Technology to Build Learning Software Organizations, in: *Proc. of the 1st Workshop Building, Maintaining, and Using Organizational Memories (OM-98)*, *13th European Conference on AI (ECAI '98)*, Brighton, 1998. http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/ Vol-14/

[4] J. Angele, D. Fensel, D. Landes, and R. Studer, Developing Knowledge-Based Systems with MIKE, *Journal of Automated Software Engineering* 5, 4 (October 1998), 389-418.

[5] J. Angele, D. Fensel, and R. Studer, Domain and Task Modeling in MIKE, in: A. Sutcliffe et al., eds., *Domain Knowledge for Interactive System Design*, Chapman & Hall, 1996.

[6] H. Akkermans, B. Wielinga, and A.Th. Schreiber, Steps in Constructing Problem-Solving Methods, in: N. Aussenac et al., eds., *Knowledge Acquisition for Knowledge-based Systems, 7th European Workshop (EKAW'93)*, Toulouse, Lecture Notes in AI 723, Springer-Verlag, 1993.

[7] S. Bamberger, U. Gappa, F. Klügl, and F. Puppe, Komplexitätsreduktion durch grafische Wissensabstraktion, in: P. Mertens and H. Voss, eds., *Expertensysteme 97 (XPS'97), Proc. in Artificial Intelligence* 6, infix, St. Augustin, 1997.

[8] V. R. Benjamins, D. Fensel, and A. Gómez Pérez, Knowledge Management through Ontologies, in: *Proceedings of the 2nd International Conference on Practical Aspects of Knowledge Management (PAKM '98)*, Basel, Switzerland, October 1998.

[9] V. R. Benjamins, D. Fensel, and R. Straatman, Assumptions of Problem-solving Methods and Their Role in Knowledge Engineering, in: W. Wahlster, editor, *Proc. ECAI-96*, pages 408-412. J. Wiley & Sons, Ltd., 1996.

[10] V.R. Benjamins, E. Plaza, E. Motta, D. Fensel, R. Studer, B. Wielinga, G. Schreiber, Z. Zdrahal, and S. Decker: IBROW3: An Intelligent Brokering Service for Knowledge-Component Reuse on the World-Wide Web, in: *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling, and Management (KAW '98)*, Banff, Canada, April 1998. See http://www.swi.psy.uva.nl/projects/IBROW3/home.html.

[11] B.W. Boehm, A Spiral Model of Software Development and Enhancement, *Computer* 21, 5 (May 1988), 61-72.

[12] W. N. Borst and J. M. Akkermans, Engineering Ontologies, *International Journal of Human-Computer Studies*, 46 (2/3):365-406, 1997.

[13] J. A. Breuker and W. van de Velde, eds., *The CommonKADS Library For Expertise Modelling*, IOS Press, Amsterdam, 1994.

[14] S. Buckingham Shum, Negotiating the Construction and Reconstruction of Organisational Memories, *Journal of Universal Computer Science*, 3(8), *Special Issue on Information Technology for Knowledge Management*, Springer Science Online, 1997.

[15] T. Bylander and B. Chandrasekaran, Generic Tasks in Knowledge-based Reasoning: The Right Level of Abstraction for Knowledge Acquisition, in: B. Gaines and J. Boose, eds., *Knowledge Acquisition for Knowledge Based Systems*, Vol. 1, Academic Press, London, 1988.

[16] B. Chandrasekaran, Generic Tasks in Knowledge-based Reasoning: High-level Building Blocks for Expert System Design, *IEEE Expert* 1, 3 (1986), 23-30.

[17] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom: The TSIMMIS Project: Integration of Heterogeneous Information Sources, in: *Proceedings of IPSJ Conference, Tokyo, Japan, October 1994*, 7-18.

[18] W.J. Clancey, The Epistemology of a Rule-Based Expert System - a Framework for Explanation, *Artificial Intelligence* 20 (1983), 215-251.

[19] W.J. Clancey, From Guidon to Neomycin and Heracles in Twenty Short Lessons, in: A. van

Lamsweerde, ed., *Current Issues in Expert Systems,* Academic Press, 1987.

[20] W.J. Clancey, The Knowledge Level Reinterpreted: Modeling How Systems Interact, *Machine Learning* 4, 1989, 285-291.

[21] F. Cornelissen, C.M. Jonker, and J. Treur; Compositional Verification of Knowledge-based Systems: A Case Study for Diagnostic Reasoning, 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, 1997.

[22] J.-M. David, J.-P. Krivine, and R. Simmons, eds., *Second Generation Expert Systems*, Springer-Verlag, Berlin, 1993.

[23] The Dublin Core Initiative, *http://purl.org/metadata/dublin_core.*

[24] S. Decker, M. Daniel, M. Erdmann, and R. Studer, 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, 1997.

[25] S. Decker, M. Erdmann, D. Fensel, and R. Studer: Ontobroker: Ontology-based Access to Distributed and Semi-Structured Information, in: *Proc. 8th IFIP 2.6 Working Conf. on Database Semantics (DS-8)*, Rotorua, January 1999.

[26] R. Dieng, O. Corby, A. Giboin and M. Ribière, Methods and Tools for Corporate Knowledge Management, in: *Proc. of the 11th Knowledge Acquisition, Modeling and Management for Knowledge-based Systems Workshop (KAW'98)*, Banff, 1998.

[27] H. Ehrig and B. Mahr, eds., *Fundamentals of Algebraic Specifications 1*, Springer-Verlag, Berlin, 1985.

[28] H. Ehrig and B. Mahr, eds., *Fundamentals of Algebraic Specifications 2*, Springer-Verlag, Berlin, 1990.

[29] H. Eriksson, A Survey of Knowledge Acquisition Techniques and Tools and their Relationship to Software Engineering, *Journal of Systems and Software* 19, 1992, 97-107.

[30] H. Eriksson, A. R. Puerta, and M. A. Musen, Generation of Knowledge Acquisition Tools from Domain Ontologies, *Int. J. Human-Computer Studies* 41, 1994, 425-453.

[31] H. Eriksson, Y. Shahar, S.W. Tu, A.R. Puerta, and M.A. Musen, Task Modeling with Reusable Problem-Solving Methods, *Artificial Intelligence* 79 (1995), 293-326.

[32] J. Euzenat, Corporate Memory through Cooperative Creation of Knowledge Bases and Hyper-documents, in: *Proc. of the 10th Knowledge Acquisition, Modeling and Management for Knowledge-based Systems Workshop (KAW'96)*, Banff, 1996.

[33] A. Farquhar, R. Fikes, and J. Rice, The Ontolingua Server: A Tool for Collaborative Ontology Construction, *International Journal of Human-Computer Studies*, 46:707-728, 1997.

[34] D. Fensel, Formal Specification Languages in Knowledge and Software Engineering, *The Knowledge Engineering Review* 10, 4, 1995.

[35] D. Fensel, J. Angele, and R. Studer, The Knowledge Acquisition and Representation Language KARL, *IEEE Transactions on Knowledge and Data Engineering* 10 (4), 527-550, 1998.

[36] D. Fensel; The Tower-of-Adapter Method for Developing and Reusing Problem-Solving Methods, in: E. Plaza et al., eds., *Knowledge Acquisition, Modeling and Management*, Lecture Notes in Artificial Intelligence 1319, Springer-Verlag, Berlin, 97-112, 1997.

[37] D. Fensel, S. Decker, M. Erdmann and R. Studer: Ontobroker: The Very High Idea, in: *Proceedings of the 11th International Flairs Conference (FLAIRS-98)*, Sanibel Island, 131-135, May 1998.

[38] D. Fensel and R. Groenboom, Specifying Knowledge-based Systems with Reusable Components, in: *Proceedings 9th Int. Conference on Software Engineering and Knowledge Engineering (SEKE '97)*, Madrid 1997.

[39] D. Fensel and A. Schönegge, Using KIV to Specify and Verify Architectures of Knowledge-Based Systems, in: *Proceedings of the 12th IEEE International Conference on Automated Software Engineering (ASEC-97)*, Incline Village, Nevada, November 1997.

[40] D. Fensel and R. Straatman, The Essence of Problem-Solving Methods: Making Assumptions for Efficiency Reasons, in: N. Shadbolt et al., eds., *Advances in Knowledge Acquisiiton,* Lecture Notes in Artificial Intelligence 1076, Springer-Verlag, Berlin, 1996.

[41] D. Fensel and F. van Harmelen, A Comparison of Languages which Operationalize and Formalize KADS Models of Expertise, *The Knowledge Engineering Review* 9, 2, 1994.

[42] N. Fridman-Noy and C.D. Hafner, The State of the Art in Ontology Design, *AI Magazine*, 18(3):53-74, 1997.

[43] B. Gaines and M.L.G. Shaw, New Directions in the Analysis and Interactive Elicitation of Personal Construct Systems, *Int. J. Man-Machine Studies* 13 (1980), 81-116.

[44] U. Gappa, Grafische Wissensakquisitionssysteme und ihre Generierung, Ph.D. Theses in Artificial Intelligence (DISKI 100), infix, St. Augustin.

[45] M.R. Genesereth and R.E. Fikes, Knowledge Interchange Format, Version 3.0, Reference Manual. Technical Report, Logic-92-1, Computer Science Dept., Stanford University, 1992. http://www.cs.umbc.edu/kse/.

[46] M.R. Genesereth, A.M. Keller, and O.M. Duschka, Infomaster: An Information Integration System, in: *Proc. ACM SIGMOD Conference*, Tucson, 1997.

[47] J.H. Gennari, H. Cheng, R.B. Altman, and M.A. Musen: Reuse, CORBA, and Knowledge-Based Sysems, *Int. J. on Human-Computer Studies* 49*,* 1998.

[48] J.H. Gennari, S.W. Tu, T.E. Rothenfluh, and M.A. Musen, Mappings Domains to Methods in Support of Reuse, *Int. J. on Human-Computer Studies* 41 (1994), 399-424.

[49] Y. Gil and C. Paris, Towards Method-independent Knowledge Acquisition, *Knowledge Acquisition* 6, 2 (1994), 163-178.

[50] T.R. Gruber, A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition* 5, 2, 1993, 199-221.

[51] T.R. Gruber, Towards Principles for the Design of Ontologies used for Knowledge Sharing, *International Journal of Human-Computer Studies*, 43:907-928, 1995.

[52] N. Guarino, Formal Ontology, Conceptual Analysis and Knowledge Representation, *International Journal of Human-Computer Studies*, 43(2/3):625-640, 1995.

[53] N. Guarino, ed., *Formal Ontology in Information Systems*, IOS Press, Amsterdam, 1998.

[54] F. van Harmelen and D. Fensel, Formal Methods in Knowledge Engineering, *The Knowledge Engineering Review* 9, 2, 1994.

[55] J. Jannink, S.Pichai, D. Verheijen, and G. Wiederhold; Encapsulation and Composition of Ontologies, in: *Proc. AAAI Workshop AI and Information Integration*, Madison, July 1998.

[56] M. Kifer, G. Lausen, and J. Wu, Logical Foundations of Object-Oriented and Frame-Based Languages, *Journal of the ACM* 42 (1995), 741-843.

[57] O. Kühn and A. Abecker, Corporate Memories for Knowledge Management in Industrial Practice: Prospects and Challenges, *J. of Universal Computer Science* 3, 8 (August 1977), Special Issue on Information Technology for Knowledge Management, Springer Science Online. URL: http://www.iicm.edu/jucs_3_8/corporate_memories_for_knowledge.

[58] D. Landes, DesignKARL - A Language for the Design of Knowledge-based Systems, in: *Proc. 6th International Conference on Software Engineering and Knowledge Engineering (SEKE'94)*, Jurmala, Lettland, 1994, 78-85.

[59] D. Landes, K. Schneider and F. Houdek, Organizational Learning and Experience Documentation in Industrial Software Projects, in: *Proc. of the 1st Workshop Building, Maintaining, and Using Organizational Memories (OM-98)*, *13th European Conference on AI (ECAI '98)*, Brighton, 1998.
http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-14/

[60] D. Landes and R. Studer, The Treatment of Non-Functional Requirements in MIKE, in: W. Schaefer et al., eds., *Proc. of the 5th European Software Engineering Conference (ESEC´95)*, Sitges, Lecture Notes in Computer Science 989, Springer-Verlag, 1995.

[61] D. B. Lenat and R. V. Guha, *Representation and Inference in the Cyc Projec,* Addison-Wesley, 1990.

[62] S. Luke, L. Spector, D. Rager, and J. Hendler, Ontology-based Web Agents, in: *Proc. 1st Int. Conf. on Autonomous Agents*, 1977.

[63] R. MacGregor, Inside the LOOM Classifier, *SIGART Bulletin*, 2(3):70-76, June 1991.

[64] S. Marcus, ed., *Automating Knowledge Acquisition for Experts Systems*, Kluwer Academic Publisher, Boston, 1988.

[65] F. Maurer and B. Dellen; An Internet Based Software Process Management Environment, in: *Proc. ICSE 98 Workshop on Software Engineering over the Internet*, 1998.

[66] E. Miller, An Introduction to the Resource Description Framework, *D-Lib Magazine*, May 1998.

[67] M.A. Musen, An Overview of Knowledge Acquisition, in: J.-M. David et al., eds., *Second Generation Expert Systems*, Springer-Verlag, 1993.

[68] J. Myplopoulos and M. Papazoglou, Cooperative Information Systems, Guest Editors' Introduction, *IEEE Intelligent Systems* 12, 5 (September/October 1997), 28-31.

[69] S. Neubert, Model Construction in MIKE, in: N. Aussenac et al., eds., *Knowledge Acquisition for Knowledge-based Systems, Proc. 7th European Workshop (EKAW'93)*, Toulouse, Lecture Notes in Artificial Intelligence 723, Springer-Verlag, 1993.

[70] A. Newell, The Knowledge Level, *Artificial Intelligence* 18, 1982, 87-127.

[71] I. Nonaka and H. Takeuchi, *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*, Oxford University Press, 1995.

[72] D. O'Leary, Enterprise Knowledge Management, *IEEE Computer*, 31(3):54-61, 1998.

[73] R. Orfali, D. Harkey, and J. Edwards, eds., *The Essential Distributed Objects Survival Guide*, John Wiley & Sons, New York, 1996.

[74] M. Perkowitz and O. Etzioni, Adaptive Web Sites: An AI Challenge, in: *Proceedings of the 15th International Joint Conference on AI (IJCAI-97)*, Nagoya, Japan, August 1997.

[75] Th. Pirlein and R. Studer, Integrating the Reuse of Commonsense Ontologies and Problem-Solving Methods, *Int. Journal of Expert Systems: Research and Applications*, 1999, in press.

[76] K. Poeck and U. Gappa, Making Role-Limiting Shells More Flexible, in: N. Aussenac et al., eds., *Knowledge Acquisition for Knowledge-Based Systems, Proc. 7th European Knowledge Acquisition Workshop (EKAW´93)* Toulouse, Lecture Notes in Artificial Intelligence 723, Springer-Verlag, 1993.

[77] A. R. Puerta, J. W. Egar, S. W. Tu, and M. A. Musen, A Multiple-Method Knowledge Acquisition Shell for the Automatic Generation of Knowledge Acquisition Tools, *Knowledge Acquisition* 4, 1992, 171-196.

[78] F. Puppe, U. Gappa, K. Poeck, and S. Bamberger: *Wissensbasierte Diagnose- und Informationssysteme*, Springer-Verlag, Berlin, 1996.

[79] A.Th. Schreiber, B. Wielinga, H. Akkermans, W. van de Velde, and A. Anjewierden, CML: The CommonKADS Conceptual Modeling Language, in: Steels et al., eds., *A Future of Knowledge Acquisition, Proc. 8th European Knowledge Acquisition Workshop (EKAW'94)*,

Hoegaarden, Lecture Notes in Artificial Intelligence 867, Springer-Verlag, 1994.

[80] A.Th. Schreiber, B. Wielinga, and J. Breuker, eds., *KADS. A Principled Approach to Knowledge-Based System Development*, Knowledge-Based Systems, vol 11, Academic Press, London, 1993.

[81] A.Th. Schreiber, B.J. Wielinga, R. de Hoog, H. Akkermans, and W. van de Velde, CommonKADS: A Comprehensive Methodology for KBS Development, *IEEE Expert*, December 1994, 28-37.

[82] N. Shadbolt, E. Motta, and A. Rouge, Constructing Knowledge-based Systems, *IEEE Software* 10, 6, 34-38.

[83] E.W. Stein, Organizational Memory: Review of Concepts and Recommandations for Management, *International Journal of Information Management,* 15:17-32, 1995.

[84] R. Studer, H. Eriksson, J.H. Gennari, S.W. Tu, D. Fensel, and M.A. Musen, Ontologies and the Configuration of Problem-Solving Methods, in: *Proc. of the 10th Knowledge Acquisition for Knowledge-based Systems Workshop (KAW'96)*, Banff, 1996.

[85] R. Studer, R. Benjamins, and D. Fensel; Knowledge Engineering: Principles and Methods. *Data & Knowledge Engineering* 25 (1988), 161-197.

[86] B. Swartout, R. Patil, K. Knight, and T. Russ, Toward Distributed Use of Large-scale Ontologies, in: B. R. Gaines and M. A. Musen, editors, *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, pages 32.1-32.19, Alberta, Canada, 1996. SRDG Publications, University of Calgary. http://ksi.cpsc.ucalgary.ca:80/KAW/KAW96/KAW96Proc.html.

[87] A. ten Teije and F. van Harmelen, Characterizing Approximative Problem-solving: from Partially Fulfilled Preconditions to Partially Achieved Functionality, in: *Proceedings of the 13th European Conference on AI (ECAI-98)*, Brighton, UK, August 1998.

[88] M. Uschold and M. Gruninger, Ontologies: Principles, Methods, and Applications, *Knowledge Engineering Review*, 11(2):93-155, 1996.

[89] G. van Heijst, A. Th. Schreiber, and B. J. Wielinga, Using Explicit Ontologies in KBS Development, *International Journal of Human-Computer Studies*, 46(2/3):183-292, 1997.

[90] G. Wiederhold and M. Genesereth, The Conceptual Basis for Mediation Services, *IEEE Intelligent Systems* 12, 5 (September/October 1997), 38-47.

[91] B.J. Wielinga, J. Sandberg, and G. Schreiber, Methods and Techniques for Knowledge Management: What has Knowledge Engineering to Offer, *Expert Systems with Applications* 13, 1 (1997), 73-84.

[92] K. Wiig, R. de Hoog and R. van der Spek, Supporting Knowledge Management: A Selection of Methods and Techniques, *Expert Systems With Applications* 13 (1997), 15-27.

[93] Extensible Markup Language (XML) 1.0, *http://www.w3.org/TR/REC-xml.*