Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods

Asunción Gómez Pérez
DIA
Technical University of Madrid
Spain
asun@delicias.dia.fi.upm.es

V. Richard Benjamins
SWI
University of Amsterdam
The Netherlands
richard@swi.psy.uva.nl

Abstract

Ontologies and problem-solving methods are promising candidates for reuse in Knowledge Engineering. Ontologies define domain knowledge at a generic level, while problem-solving methods specify generic reasoning knowledge. Both type of components can be viewed as complementary entities that can be used to configure new knowledge systems from existing, reusable components. In this paper, we give an overview of approaches for ontologies and problem-solving methods.

1 Introduction

In 1991, the ARPA Knowledge Sharing Effort [NFF+91] envisioned a new way in which intelligent systems could be built. They proposed the following: "Building knowledge-based systems today usually entails constructing new knowledge bases from scratch. It could be done by assembling reusable components. Systems developers would then only need to worry about creating the specialized knowledge and reasoners new to the specific task of their system. This new system would interoperate with existing systems, using them to perform some of its reasoning. In this way, declarative knowledge, problem-solving techniques and reasoning services would all be shared among systems. This approach would facilitate building bigger and better systems cheaply..."

Since then considerable progress has been made in developing the conceptual bases needed for building technology that allows knowledge-component reuse and sharing. However, we are still far from the ultimate objective. To enable sharing and reuse of knowledge and reasoning behavior across domains and tasks, Ontologies and Problem-Solving Methods (PSMs) have been developed. Ontologies are concerned with static domain knowledge and PSMs with dynamic reasoning knowledge. The integration of ontologies and PSMs is a possible solution to the "interaction problem" [BC88], which hampered reuse in the eighties. The interaction problem states that representing knowledge for the purpose of solving some problem is strongly affected by the nature of the problem and the inference strategy to be applied to the problem. Through ontologies and PSMs this interaction can be made explicit in the notion of assumptions and taken into consideration. PSMs and ontologies can be seen as complementary reusable components to construct knowledge systems from reusable components. In order to build full applications of information and knowledge systems from reusable components, both PSMs and ontologies are required in a tightly integrated way.

Ontologies aim at capturing domain knowledge in a generic way and provide a commonly agreed understanding of a domain, which may be reused and shared across applications and groups [CJB99]. Ontologies provide a common vocabulary of an area and define -with different levels of formality- the meaning of the terms and the relations between them. Ontologies are usually organized in taxonomies and typically contain modeling primitives such as classes, relations, functions, axioms and instances [Gru93]. Popular applications of ontologies include knowledge management, natural language generation, enterprise modeling, knowledge-based systems, ontology-based brokers, and interoperability between systems.

Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5) Stockholm, Sweden, August 2, 1999

(V.R. Benjamins, B. Chandrasekaran, A. Gomez-Perez, N. Guarino, M. Uschold, eds.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/

Problem-solving methods (PSMs) describe the reasoning process of a knowledge-based system (KBS) in an implementation- and domain-independent manner. A PSM defines a way of how to achieve the goal of a task. It has inputs and outputs and may decompose a task into subtasks. In addition, a PSM specifies the data flow between its subtasks. Control knowledge determines the execution order and iterations of the subtasks of a PSM.

In the following sections, we will discuss several aspects of both ontologies and problem-solving methods. At the end of the paper, we will mention some directions for future work in this area.

2 Ontologies

The aims of this section are to provide answers to the following questions: What is an ontology? What principles should I follow to build an ontology? What are the components of an ontology? What types of ontologies exist? How are ontologies organized in libraries? What methods should I use to build my own ontology? Which techniques are appropriate for each step? How do software tools support the process of building and using ontologies? What are the most well-known ontologies? What are the uses of ontologies? Which principles should I use to select the best ontology for my application? To answer the above questions, the section is organized as follows. First, the theoretical foundations of the ontological engineering field will be presented. This will be followed by a presentation of some existing ontologies. The second part will address methodologies for building ontologies. The third part will present tools for building ontologies. Finally, the last will be related to uses of ontologies in applications.

2.1 Theoretical Foundations

What is an ontology?

The word ontology has been taken from Philosophy, where it means a systematic explanation of Existence. In the Artificial Intelligent field, first Neches and colleagues [NFF+91] defined an ontology as follows "An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary". We can say that this definition tells us how to proceed to build an ontology, giving us vague guidelines: identify basic terms and relations between terms, identify rules to combine them, provide definitions of such terms and relations. Note that according to this definition, an ontology includes not only the terms that are explicitly defined in it, but also terms that can be inferred using rules. Later, in 1993, Gruber's definition [Gru93] becomes famous "an ontology is an explicit specification of a conceptualization", being this definition the most referenced in the literature. In 1997, Borst [Bor97] slightly modify Gruber's definition saying that: "Ontologies are defined as a formal specification of a shared conceptualization". These two definitions have been explained by Studer and Colleagues [SBF98] as follows: " Conceptualization refers to an abstract model of some phenomenon in the world by having identified 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. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group."

Based on the definition of Gruber, many definitions of what an ontology is have been proposed in the literature. In 1995, Guarino and Giaretta [GG95] collected seven definitions and provided corresponding syntactic and semantic interpretations. Other definitions are: "an ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base" [SPKR97], and "An ontology provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base." [BLC96].

As a main conclusion to this section, we can say that the literature provides several definitions of the word ontology. Different definitions provide different and complementary points of view of the same reality.

What principles should I follow to build ontologies?

Here we summarize some design criteria and a set of principles that have been proved useful in the development of ontologies.

- Clarity and Objectivity [Gru95], which means that the ontology should provide the meaning of defined terms by providing objective definitions and also natural language documentation.
- Completeness [Gru95], which means that a definition expressed in terms of necessary and sufficient conditions is preferred over a partial definition (defined only through necessary or sufficient condition).
- Coherence [Gru95], to permit inferences that are consistent with the definitions.
- Maximum monotonic extendibility [Gru95]. It means that new general or specialized terms should be included in the ontology in a such way that is does not require the revision of existing definitions.
- Minimal ontological commitments[1] [Gru95], which means to make as few claims as possible about the world being modeled, giving the parties committed to the ontology freedom to specialize and instantiate the ontology as required.

[1] "Ontological commitments refer to agreement to use the shared vocabulary in a coherent and consistent manner. They guarantee consistency, but not completeness of an ontology" [GO94].

- Ontological Distinction Principle [BGM96], which means that classes in an ontology should be disjoint.
- Diversification of hierarchies to increase the power provided by multiple inheritance mechanisms [AGLP98].
- Modularity [BLC96] to minimize coupling between modules.
- Minimization of the semantic distance between sibling concepts [AGLP98] which means that similar concepts are grouped and represented using the same primitives.
- Standardization of names whenever is possible [AGLP98].

What are the components of ontologies?

Knowledge in ontologies is formalized using five kinds of components: classes, relations, functions, axioms and instances [Gru93]. Classes in the ontology are usually organized in taxonomies. Sometimes, the notion of ontology is diluted, in the sense that taxonomies are considered to be full ontologies [SBF98].

- Concepts are used in a broad sense. A concept can be anything about which something is said and, therefore, could also be the description of a task, function, action, strategy, reasoning process, etc.
- Relations represent a type of interaction between concepts of the domain. They are formally defined as any subset of a product of n sets, that is: R: C1 x C2 x ... x Cn. Examples of binary relations include: subclass-of and connected-to.
- Functions are a special case of relations in which the n-th element of the relationship is unique for the n-1 preceding elements. Formally, functions are defined as: F: C1 x C2 x ... x Cn-1 $\rightarrow$ Cn. Examples of functions are Mother-of and Price-of-a-used-car that calculates the price of a second-hand car depending on the car-model, manufacturing date and number of kilometers.
- Axioms are used to model sentences that are always true.
- Instances are used to represent elements.

Once the main components of ontologies have been represented, the ontology can be implemented in a various languages: highly informal, semi-informal, semi-formal and rigorously formal languages [Usc96].

What types of ontologies already exist?

Nowadays, it is easy to get information from organizations that have ontologies on the WWW. Many ontologies like Ontolingua ontologies at the Ontology Server[2] [FFR97] and WordNet[3] [Mil90] at Princenton are freely available over the Internet. Other ontologies, like Cyc ontologies[4] [LG90], are partially freely available on the web. However, the majority of ontologies have been developed by companies for their own use and are not available. The Ontology Page[5] (also known as TOP) and (Onto)2Agent[6] [AGLP98] (an ontology-based www broker that helps to select ontologies) might help to select ontologies.

This section does not seek to give an exhaustive typology of ontologies as presented in [vSW97, MVI95]. However, it presents the most commonly used types of ontologies.

- Knowledge Representation ontologies [vSW97] capture the representation primitives used to formalize knowledge in knowledge representation paradigms. The most representative example is the Frame-Ontology [Gru93], which captures the representation primitives used in frame-based languages. It allows other ontologies to be specified using frame-based conventions. It is implemented in KIF 3.0 [GF92].
- General/Common ontologies [Gua98] include vocabulary related to things, events, time, space, causality, behavior, function, etc.

  The CYC ontology [LG90] is a common sense ontology that provides a vast amount of fundamental human knowledge. The Cyc ontology is divided into many micro-theories. Cyc Ontologies are implemented in CycL language.
- Top-Level Ontologies provide general notions under which with all the terms in existing ontologies are related. Examples of top level ontologies are: Sowa's boolean lattice [Sow97], PANGLOSS [KL94], Penman Upper Level [BKMW90], Cyc [LG90], Mikrokosmos [Mah96] and Guarino's top level proposal [Gua98].
- Meta-ontologies, also called Generic Ontologies or Core Ontologies [vSW97] are reusable across domains. The Mereology ontology [Bor97] could be the most typical example. It defines the part-of relation and its properties. This relation allows to express that devices are assembled of components, each of which might -on its turn- be decomposed in subcomponents.

[2] http://www-ksl.stanford.edu:5915 or the European mirror site at http://www-ksl-svc-lia.dia.fi.upm.es:5915
[3] http://www.tio.darpa.mil/Summaries95/B370-Princenton.html
[4] http://www.cyc.com/
[5] http://www.medg.lcs.mit.edu/doyle/top
[6] http://delicias.dia.fi.upm.es/REFERENCE_ONTOLOGY/

- Domain ontologies [MVI95, vSW97] are reusable in a given domain. They provide vocabularies about the concepts within a domain and their relationships, about the activities that take place in that domain, and about the theories and elementary principles governing that domain.

  In the domain of engineering ontologies, the EngMath ontology [GO94] and PhysSys [Bor97] deserve special mention. EngMath is an Ontolingua ontology developed for mathematical modeling in engineering. PhysSys is an engineering ontology for modeling, simulating and designing physical systems.

  In the domain of enterprise modeling process, the Enterprise Ontology[7] [Usc96] is a collection of terms and definitions relevant to business enterprises. Ontologies built at the TOVE [GF95] (Toronto Virtual Enterprise)[8] project are: Enterprise Design Ontology, Project Ontology, Scheduling Ontology, or Service Ontology.

  An illustrative example of ontologies for Knowledge Management is the $(KA)^2$ ontology[9] [BFDGP97], to be used by the Knowledge Annotation Initiative of the Knowledge Acquisition Community. This ontology is being built jointly and distributively with people at different locations.

- The most illustrative linguistic ontologies are the Generalized Upper Model [BMF95], WordNet [Mil90] and Sensus [SPKR97]. The Generalized Upper Model (GUM[10]) is a general task and domain-independent linguistic ontology. To make it portable across different languages (English, German, Spanish, Italian, etc.), the GUM ontology only includes the main linguistic concepts and how they are organized across languages, and omits details that differentiate languages. WordNet is a lexical database for English based on psycholinguistic principles. Its information is organized in units called "synsets", which are sets of synonyms that are interchangeable in a particular context and are used to represent different meanings. SENSUS is a natural language based ontology whose goal is to provide a broad conceptual structure for work in machine translation. It was developed by merging and extracting information from existing electronic resources.

- Task ontologies [MVI95] provide a systematic vocabulary of the terms used to solve problems associated with tasks that may or may not be from the same domain. They include generic names, generic verbs generic adjectives and others in the scheduling tasks.

- Domain-Task ontologies are task ontologies reusable in a given domain, but not across domains.

- Method ontologies provide definitions of the relevant concepts and relations used to specify a reasoning process to achieve a particular task [CJB99].

- Application ontologies [vSW97] contain the necessary knowledge for modeling a particular application.

Meta-ontologies, domain ontologies and applications ontologies capture static knowledge in a problem-solving independent way, where as PSMs ontologies, task ontologies and domain-task ontologies are concerned with problem solving knowledge. All these kind of ontologies can be combined to build a new ontology. The reusability-usability trade-off problem [KBD+91] applied to the ontology field states that the more reusable an ontology is, the less usable it is, and vice versa. The first thing to do to model a new ontology using existing ontologies from the library is to decide which knowledge representation paradigm to use to formalize knowledge, which will then be committed to into a knowledge representation ontology. Having selected the knowledge representation ontology, the next step is to decide whether general/common ontologies are needed in the new ontology. If they are required, new ontologies are built and entered into the library or reused from the library. This is when knowledge-component modeling starts. Simultaneously, domain knowledge and problem-solving knowledge can be modeled. So, when domain knowledge is modeled, first generic ontologies, then domain ontologies, and finally application domain ontologies are built. When problem-solving knowledge is modeled, first Task and PSMs ontologies, then domain task ontologies and finally application domain task ontologies are modeled. Method and task ontologies allow the assumption-based interaction between problem-solving and domain ontologies to be explicitly stated [BFS96, BPG96, FS98].

2.2 Methodologies for building ontologies

The ontology building process is a craft rather than an engineering activity. Each development team usually follows its own set of principles, design criteria and phases in the ontology development process. The absence of commonly agreed on guidelines and methods hinders the development of shared and consensual ontologies within and between teams, the extension of a given ontology by others and its reuse in other ontologies and final applications. If ontologies are built on a small scale, some activities can be skipped. But, if you intend to build large-scale ontologies with some guarantees of correctness and completeness, it is advisable to steer clear of anarchic constructions and to follow a methodological approach.

[7] http://www.aiai.ed.ac.uk/project/enterprise
[8] http://www.ie.utoronto.ca/EIL
[9] http://www.aifb.uni-karlsruhe.de/WBS/broker/KA2.html
[10] http://www.darmstadt.gmd.de/publish/komet/gen-um/newUM.html

Uschold's methodology [UG96, Usc96] is based on the experience of building the Enterprise Ontology, which includes a set of ontologies for enterprise modeling, and proposes the following steps: (1) identify the purpose and scope of the ontology; (2) build the ontology by capturing knowledge, coding knowledge and integrating the knowledge with existing ontologies; (3) evaluate the ontology; (4) documentation; and (5) guidelines for each phase.

Grüninger and Fox's methodology [GF95] is based on the experience of building an enterprise modeling ontology in the framework of the TOVE project. Essentially, it involves building a logical model of the knowledge that is to be specified in the ontology. This model is not built directly. First, the specifications that are to be met by the ontology are described informally by identifying a set of competency questions, and this description is then formalized in a language based on first-order predicate calculus. The competency questions are the basis for a rigorous characterization of the knowledge that the ontology has to cover, and they specify the problem and what constitutes a good solution to the problem. By a composition and decomposition mechanism, competency questions and their answers can be used to answer more complex competency questions in other ontologies, allowing the integration of ontologies.

The METHONTOLOGY framework [GP98, FGPPP99] enables the construction of ontologies at the knowledge level. It includes: (a) the identification of the ontology development process, which refers to which tasks (planning, control, specification, knowledge acquisition, conceptualization, integration, implementation, evaluation, documentation, configuration management, etc.) one should carry out when building ontologies; (b) a life cycle based on evolving prototypes, which identifies the stages through which the ontology passes during its lifetime; and (c) the methodology itself, which specifies the steps to be taken to perform each activity, the techniques used, the products to be output and how they are to be evaluated. The main phase is the conceptualization phase. During both specification and conceptualization, a process of integration was completed using in-house and external ontologies. This framework is partially supported by a software environment called Ontology Design Environment (ODE) [BFGPGP98], [FGPPP99]. Several ontologies have been developed using METHONTOLOGY and ODE: CHEMICALS [FGPPP99], Environmental pollutants ontologies [GPR99], the Reference-Ontology [AGLP98] and the restructured version of the $(KA)^2$ ontology [BFGPGP98]. This methodology has been proposed to build ontologies by the Foundation for Intelligent Physical Agents (FIPA[11]).

All these methodologies have in common that they start from the identification of the purpose of the ontology and the need for domain knowledge acquisition. However, having acquired a significant amount of knowledge, Uschold's methodology proposes coding in a formal language and METHONTOLOGY proposes expressing the idea as a set of intermediate representations (IR). Then the ontology is generated using translators. These IRs bridge the gap between, on the one hand, how people see a domain and, on the other hand, the languages in which ontologies are formalized. These intermediate representations provide a user-friendly approach for both knowledge acquisition and evaluation by computer scientists and domain experts who are not knowledge engineers [ABB+98].

The need for ontology evaluation is also identified in the three above methodologies. Uschold's methodology includes this activity but it does not state how it should be carried out. Grüninger and Fox propose identifying a set of competency questions. Once the ontology has been expressed formally, it is compared against this set of competency questions. Finally, METHONTOLOGY proposes that evaluation activities be carried out throughout the entire lifetime of the ontology development process. Most of the evaluation is done in the conceptualization phase.

As main conclusion at this point we can say that each group has and uses its own methodology and there does not yet exist a common methodology that everybody agrees on. Therefore, additional research has to be performed in this direction.

2.3 Languages and environments for building ontologies

Which are the most commonly used languages to build ontologies?

Basically, several representation systems have been reported for formalizing ontologies under a frame-based modeling approach, a logic-based approach or even both. The most representative languages are Ontolingua [Gru93], CycL [LG90], Loom [Mac91] and FLogic [KLW95].

Ontolingua is a language based on KIF and on the Frame Ontology, and is the ontology-building language used by the Ontology Server. The Ontolingua language allows ontologies to be built in any of the following three manners: (1) using KIF expressions; (2) using exclusively the Frame Ontology vocabulary; (3) using both languages at the same time, depending on ontology developer preferences. In any case, the Ontolingua definition is composed of a heading, an informal definition in natural language, and a formal definition written in KIF or using the frame ontology vocabulary. A GFP [CFF+97] application is required in order to reason with Ontolingua Ontologies.

CycL is Cyc's knowledge representation language. CycL is a declarative and expressive language, similar to first-order predicate calculus with extensions. CycL uses a form of circumscription, includes the unique names assumption, and can make use of the closed world assumption where appropriate. CycL has an inference engine to perform several kinds of reasonings.

[11] http://www.fipa.org

LOOM is a high-level programming language based on first-order logic which belongs to the KL-ONE family. The LOOM language provides: an expressive and explicit declarative model specification language, a powerful deductive support, several programming paradigms, and knowledge-base services.

FLogic is an integration of frame-based languages and first-order predicate calculus. It includes objects (simple and complex), inheritance, polymorphic types, query methods and encapsulation. Its deductive system works with the theory of predicate calculus and structural and behavioral inheritance.

How do software tools support the process of building and using ontologies?

The main tools for building ontologies are: The Ontology Server [FFR97], Ontosaurus[12] [SPKR97], ODE [BFGPGP98, FGPPP99] and Tadzebao and Webonto [Dom98].

The Ontology Server is the best known environment for building ontologies in the Ontolingua language. It is a set of tools and services that support the building of shared ontologies between geographically distributed groups. It was developed in the context of the ARPA Knowledge Sharing Effort by the Knowledge Systems Laboratory at Stanford University. The ontology server architecture provides access to a library of ontologies, translators to languages (Prolog, CORBA's IDL, CLIPS, Loom, KI) and an editor to create and browse ontologies. There are three modes of interaction: remote collaborators that are able to write and inspect ontologies; remote applications that may query and modify ontologies stored at the server over the Internet using the generic frame protocol; and stand-alone applications.

Ontosaurus is being developed by the Information Sciences Institute at the University of South California. It consists of two parts: an ontology server that uses Loom as knowledge representation system and an ontology browser server that dynamically crates html pages (including image and textual documentation) that displays the ontology hierarchy and it uses html forms to allow the user to edit the ontology. Translators from loom to Ontolingua, KIF, KRSS and C++ have also been developed.

ODE (Ontology Design Environment) is being developed by the Computer Science School at Universidad Politécnica de Madrid. The main advantage of ODE is the conceptualization module for building ontologies, which allows the ontologist to develop the ontology at the knowledge level using a set of intermediate representations (IRs) that are independent of the target language in which the ontology will be implemented. Once the conceptualization is complete, the code is generated automatically using ODE code generators (Ontolingua, FLogic and a relational database). So, non-experts in the languages in which ontologies are implemented could specify and validate ontologies using this environment.

Tadzebao and WebOnto are complementary tools that are being developed by the Knowledge Media Institute at The Open University. Tadzebao enables knowledge engineers to hold synchronous and asynchronous discussion about ontologies and WebOnto supports the collaborative browsing, creation and editing of ontologies.

2.4 Applications that use ontologies

Although ontologies can be used to communicate between systems, people, and organizations, interoperate between systems, and support the design and development of knowledge-based and general software systems [Usc96], the number of applications built that use ontologies to model the application knowledge is small. That is, many times such ontologies have been built just for a given application without special consideration for sharing and reuse. Several problems make difficult the reuse of existing ontologies in applications [AGLP98]: Ontologies are dispersed over several servers; the formalization differs depending on the server on which the ontology is stored; ontologies on the same server are usually described with different levels of detail; and there is no common format for presenting relevant information about the ontologies so users can decide which ontology best suits their purpose. These problems are probably the cause for the relatively small number of known applications until now. Several applications that use ontologies can be found in the proceedings of the workshop on Applications of ontologies and PSMs held in conjunction with ECAI98.

There exist several applications that use natural language ontologies. The GUM is being used in natural language generation applications in different languages: Penman [BKMW90], KOMET [Bat94], TechDoc [Ros94], AlFresco [SCC$^+$93], OntoGeneration [ABB$^+$98], and the language of [FvdR98]. WordNet is used by Hermes [Hoe98] and OntoSeek [GMV99].

In the domain of enterprise modeling, the Enterprise tool set (see:
http://www.aiai.ed.ac.uk/project/enterprise for more information) is the most relevant environment built using the Enterprise ontology. The Enterprise Design Workbench and the Integrated Supply Chain Management Project use TOVE Ontologies.

Recently, ontologies are being used by www brokers in different domains. Ontobroker[13] [FDES98] for knowledge management in the context of the Knowledge Annotation Initiative of the Knowledge Acquisition Community, (Onto)2Agent [AGLP98] for selecting ontologies that satisfy a given set of constraints and Chemical OntoAgent [AGLP98] for teaching chemistry.

[12] http://indra.isi.edu:8000

[13] http://www.aifb.uni-karlsruhe.de/WBS/broker/

In the domain of information systems design, Comet [WMK95] supports the design of software systems, and Cosmos [WMK95] supports engineering negotiation. Both systems give design feedback to their users.

KACTUS [SWJ95] was an ESPRIT project on modeling knowledge of complex technical systems for multiple use and the role of ontologies to support it.

Plinius [vdVSM94] is a semi-automatic knowledge acquisition system from natural language text in the domain of ceramic materials, their properties and their production processes.

3 Problem-Solving Methods

Problem-Solving Methods (PSMs) are nowadays recognized as valuable components for constructing knowledge-based systems (KBSs). This is manifested by the fact that the notion of PSM is present in leading knowledge engineering frameworks such as Task Structures [CJS92], Role-Limiting Methods [Mar88a], CommonKADS [SWdH+94], Protégé [Mus93], MIKE [AFS98], Components of Expertise [Ste90], EXPECT [SG95], GDM [TvHWS93] and VITAL [DMW93]. PSMs describe the reasoning process of a knowledge-based system (KBS) in an implementation- and domain-independent manner.

Work on PSMs covers different areas such as the identification of task-specific PSMs (for diagnosis, planning, assessment, etc.), how to store and index PSMs in libraries, how to formalize PSMs, etc. The issues involved in reusing PSMs include finding the right PSM (that does -part of- the job), checking whether it is applicable in the situation at hand, and modifying it to fit the domain. In order to reuse PSMs successfully in a real-life application, one has to understand these processes. A PSM may be characterized as follows:

- A PSM specifies which inference steps have to be carried out for achieving the goal of a task.
- A PSM defines one or more control structures over these steps.
- Knowledge roles specify the role that domain knowledge plays in each inference step. These knowledge roles define a domain-independent generic terminology. There are two types of roles: static roles describe the domain knowledge needed by the PSM; dynamic roles form the input and output of inference steps.

PSMs play an important role in knowledge engineering and knowledge acquisition. They can for instance be used to efficiently achieve goals of tasks through the application of domain knowledge [FS98], they can guide the acquisition process of domain knowledge, and they can facilitate KBS development through their reuse.

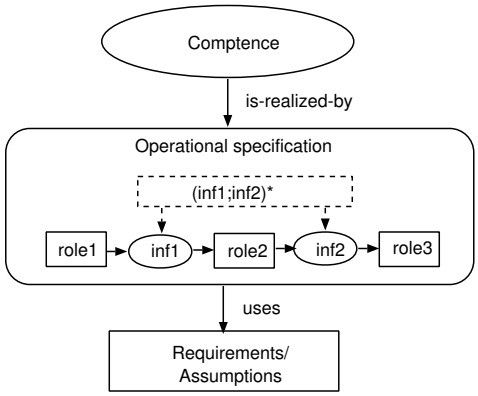

Figure 1: The architecture of a PSM.

Before discussing different approaches to PSMs, we will briefly present a general architecture of PSMs (taken from[BFS96]).

3.1 Architecture of PSMs

Most approaches agree that a PSM consists of three related parts, describing *what* a PSM can achieve, *how* it achieves it and what it *needs* to achieve it, respectively referred to as the PSM's *competence*, *operational specification* and *requirements/assumptions* (see Figure 1).

Competence The competence of a PSM is a declarative description of the input-output behavior and describes what can be achieved by the PSM.

Operational specification The operational specification of a PSM describes the reasoning process which delivers the specified competence if the required knowledge is provided. It consists of inference steps and the knowledge and control-flow between them. The inference steps specify the reasoning steps that together accomplish the competence of the method. They are described by their input/output relation and can be achieved by either a method (which means that a PSM can be hierarchically decomposed) or a primitive inference (an atomic reasoning step which is not further decomposed). The knowledge flow takes place through dynamic roles, which are stores that act as input and output of inferences. Finally, the control of a PSM describes the order of execution of the inference steps. Control knowledge can be specified in advance, if known, or can be opportunistically determined at run time depending on the dynamic problem-solving situation [Ben95]. Problem-solving methods can be used to efficiently achieve goals of tasks through the application of domain knowledge [FS98]. They can play several roles in the knowledge engineering process, such as guiding the acquisition process of domain knowledge and facilitating KBS development through their reuse.

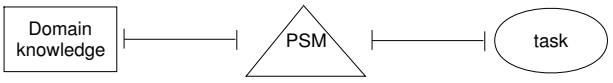

Figure 2: The two possible gaps that may prevent a PSM for being applied to its context.

Requirements/Assumptions Requirements/assumptions of a PSM describe the domain knowledge needed by the PSM to achieve its competence. Examples of such requirements in a parametric design task include the availability of heuristics that link violated constraints to possible repair actions (fixes), and the fact that a preference relation must describe a complete ordering. The requirements describe what a PSM expects in return for the competence it provides.

The internal relationship between the competence and operational descriptions of the method is that it has to be ensured that, assuming that the knowledge requirements are satisfied, the operational description describes a way to achieve the competence [FS97].

A PSM in context

PSMs can be used to realize tasks by applying domain knowledge. Thus, the external context of a PSM is formed by two parties: a task to be realized and domain knowledge to be applied. When we want to use a PSM to build a knowledge-based system, we have thus to connect the PSM with both the task and the domain knowledge. Since PSMs are generic, reusable components, they may not always fit perfectly in the context, or, in other words, there may be gaps (see Figure 2).

These gaps can exist for several reasons. In both directions (i.e. towards the domain knowledge and the task) the PSM may use different terminology than that of the domain knowledge and task, in which case a renaming process can bridge the gap. In the direction of the task, it may happen that the PSM's competence is not strong enough to realize what is specified by the task. In this case, to bridge the gap, the task may be weakened by making simplifying assumptions. Towards the domain knowledge, the knowledge required by the PSM may not be fully given by the domain knowledge, in which case additional knowledge needs to be acquired or can be assumed to exist.

3.2 Issues in PSM research

Problem-solving methods play an important role in knowledge acquisition and knowledge engineering where they have several purposes:

- KBS construction (knowledge engineering): a PSM can be helpful to describe the process of *creating* a problem solver that achieves the goal of a particular task. Often this implies a task decomposition approach.
- KBS specification (reasoning): a PSM can describe an efficient *reasoning* process that achieves the goal of a task. In this sense, a PSM concerns the product of the creation process, and is related to the design model of a KBS.
- Cognitive modeling: a PSM can describe a cognitive model of human problem-solving. An interesting question is to what extent PSMs can be used to generate cognitively adequate explanations of the reasoning process of a knowledge-based system.

PSM development

Work in this area is concerned with how PSMs are constructed in the first place. One way to do this, is by analyzing human problem-solving behavior and representing this behavior computationally. This has been traditionally the focus of Cognitive Psychology. Another way to do this, is to perform reverse engineering of existing expert systems, as has been performed by Clancey [Cla85] when he "discovered" Heuristic Classification. These two ways of developing PSMs essentially involve a creative activity, for which no methodological support exists. In the last decade, several methodologies have been developed to support knowledge modeling and the development of knowledge-based systems, such as CommonKADS [SAA+99, SWdH+94], Protégé [Mus93], MIKE [AFS98], Components of Expertise [Ste90], GDM [TvHWS93] and VITAL [DMW93].

Other approaches propose principled or even semi-automatic approaches to PSM development. One can for example start with specifying the global required competence of the problem-solving method and then step-by-step refine this competence description into an operational problem solver [WAS98]. Another approach views the construction process of PSMs as a specific type of a configuration problem [tTvHSW98] and applies a well-known problem-solving method to solve this problem: propose-critique-modify. Coming up with PSMs is one thing, but coming up with correct PSMs is another (a PSMs is correct if it actually provides what is specified in its competence). Formal methods are applied to develop such correct PSMs [PG98, FS97].

Reuse and libraries of PSMs

When PSMs have been successfully developed for a particular application, it is worthwhile to formulate the PSMs at a generic level. That is, the reusable parts of the PSMs are identified and stored in a repository or a library. When building a new application, this library can then be consulted, preventing the system engineer from developing a complete new system from scratch. Generally, reuse of PSMs includes the following questions: which generic PSMs exist and how should a library of these methods be

organized? How can PSMs be indexed in a way to support their selection for a given application? How can we support the process of adapting a generic PSM to the specific circumstances of a given application? How can individual PSMs from a library be configured into a coherent problem solver? PSM libraries are of central importance if our aim is to reuse as much as possible in a correct way.

Current work in the PSM area focuses on method-description languages such as UPML [FBMW99, GGM98]. Problem-solving methods that reside in libraries can be annotated with such languages, so that they become more accessible to others (people and software agents).

3.3 Libraries of PSMs

PSMs represent a kind of best practice in KBS construction (cf. design patterns in object-oriented approaches [GHJV95]). Instead of that knowledge engineers have to construct problem solvers from scratch, they can benefit from previous successful experiences of other developers. The use of best-practice components has as benefits that they reflect years of experience, enabling thorough validation and verification of the components, which enhances the quality of the software. Once we have a collection of such reasoning patterns, interesting issues arise such as how to structure and organize the collection and how to index the components.

3.3.1 Types of PSM libraries

Currently, there exist several libraries with PSMs. They all aim at facilitating the knowledge-engineering process, yet they differ in various ways. In particular, libraries differ along dimensions such as generality, formality, granularity and size.

- The generality dimension describes whether PSMs in a library are developed for a particular task. Task-specific libraries contain PSMs that are specialized in solving (parts of) a specific task such as diagnosis or design. Their "task-specificness" resides mainly in the terminology in which the PSMs are formulated. Examples include libraries for design [Cha90, MZ98], assessment [VL93], diagnosis [Ben93] and planning [BVB96, BHB97]. The CommonKADS library can be viewed as an extensive collection of task-specific PSMs [BvdV94]. Task-independent libraries provide problem-solving methods that are not formulated in task-specific terminology [Abe93].

- The formality dimension divides the libraries in informal, formal and implemented ones. Implemented libraries provide operational specifications of PSMs, which are directly executable [PETM92, GTRM94]. Formal libraries allow for formal verification of properties of PSMs [Abe93, Abe95, BA97, tT97]. Finally, informal libraries provide structured textual representations of PSMs. Note that within the informal approaches, PSM descriptions can vary from just textual descriptions [Cha90], to highly structured descriptions using diagrams [Ben93].

- The granularity dimension distinguishes between libraries with complex components, in the sense that the PSMs realize a complete task [MZ98], and libraries with fine-grained PSMs that realize a small part of the task. Several libraries contain both large and small building-blocks where the former are built up from the latter [Ben93, Cha90, BVB96].

- The size dimension. The most comprehensive general library is the CommonKADS library [BvdV94] which contains PSMs for diagnosis, prediction of behavior, assessment, design, planning, assignment and scheduling and engineering modeling. The most extensive library for diagnosis [Ben93] contains 38 PSMs for realizing 14 tasks related to diagnosis. The library for parametric design [MZ98] consists of five PSMs, several of them being variations of Propose & Revise [Mar88b]. The design library of [Cha90] mentions about 15 PSMs.

The type of a library is determined by its characterization in terms of the above dimensions. Each type has a specific role in the knowledge engineering process and has strong and weak points. The more general (i.e. task-neutral) PSMs in a library are, the more reusable they are, because they do not make any commitment to particular tasks. However, at the same time, applying such a PSM in a particular application requires considerable refinement and adaptation. This phenomenon is known as the reusability–usability trade-off [KBD+91]. Recently, research has been conducted to overcome this dichotomy by introducing adapters that gradually adapt task-neutral PSMs to task-specific ones [FG97] and by semi-automatically constructing the mappings between task-neutral PSMs and domain knowledge [BBvH96].

Libraries with informal PSMs provide above all support for the conceptual specification phase of the KBS, that is, they help significantly in constructing the reasoning part of the expertise model of a KBS [SWB93]. Because such PSMs are informal, they are relatively easy to understand and malleable to fit a particular application. The disadvantage is – not surprisingly – that still much work has to be done before arriving at an implemented system. Libraries with formal PSMs are particularly important if the PSMs need to have some guaranteed properties, e.g. for use in safety-critical systems such as nuclear power plants. Their disadvantage is that they are hard to understand for humans [BH95] and limit the expressiveness of the knowledge engineer. Apart from the possibility to prove properties, formal PSMs have the additional advantage of be-

ing a step closer to an implemented system. Libraries with implemented PSMs allow the construction of fully operational systems. The other side of the coin is, however, that the probability that operational PSMs exactly match the requirements of the knowledge engineer, is lower.

Developing a KBS using libraries with coarse-grained PSMs, amounts to selecting the most suitable PSM and then adapt it to the particular needs of the application [MZ98]. The advantage is that this process is quite simple as it involves only one component. The disadvantage is, however, that it is unlikely that such a library will have broad coverage, since each application might need a different (coarse-grained) PSM. The alternative approach is to have a library with fine-grained PSMs, which are then combined together (i.e. configured) into a reasoner, either manually [PETM92] or automatically [Ben95, BHB97].

3.3.2 Organization of libraries

There are several alternatives for organizing a library and each of them has consequences for indexing PSMs and for their selection. Finding the "best" organization principle for such libraries is still an issue of debate. In the following, we will present some organization principles.

Several researchers propose to organize libraries as a *task–method decomposition* structure [CJS92, PETM92, Ste93], and some available libraries are organized in this way [Ben93, Bre94a, BVB96]. According to this organization structure, a task can be realized by several PSMs, each consisting of primitive and/or composite subtasks. Composite subtasks can again be realized by alternative methods, etc. Principles for library design according to this principle are discussed in [Ors96b, Ors96a]. In a library organized according to the task-method principle, PSMs are indexed, based on two factors: (1) on the competence of the PSMs – which specifies what a PSM *can* achieve, and (2) on their assumptions – which specify the assumptions under which the PSM can be applied correctly, such as its requirements on domain knowledge [HY98]. Selection of PSMs from such libraries first considers the competence of PSMs (selecting those whose competences match the task at hand), and then the assumptions of PSMs (selecting those whose assumptions are satisfied).

Libraries can also be organized, based on the functionality of PSMs, in which case PSMs with similar functionality are stored together. In addition, the functionality of PSMs can be configured from pre-established parameters and values [tT97].

Another criterion to structure libraries of PSMs is based on assumptions, which specify under what conditions PSMs can be applied. Assumptions can refer to domain knowledge (e.g. a certain PSM needs a causal domain model) or to task knowledge (a certain PSM generates locally optimal solutions). To our knowledge, there does not exist a library organized following this principle, but work is currently being performed to shed more light on the role of assumptions in libraries for knowledge engineering [BFS96, FB98, FG97].

A last proposal to organize libraries of PSMs is based on a *suite* of so-called problem types (or tasks, for the purpose of this article tasks and problem types are treated as synonyms) [Bre94a, Bre94b]. The suite describes problem types according to the way that problems *depend* on each other. The solution to one problem forms the input to another problem. For example, the output of a prediction task is a certain state, which can form the input to a monitoring task that tries to detect problems, which on their turn can be the input to a diagnosis task. It turns out that these problem dependencies recur in many different tasks. According to this principle, PSMs are stored under the problem type they can solve. Selection of PSMs in such a library would first identify the problem type involved (or task), and then look at the respective PSMs for this task.

3.4 Industrial applications

Building KBSs from reusable components in an academic setting is one thing. Doing the same for real industrial applications is another. So far, several industrial applications have been built, but only a few have been reported in the literature. Unilever reports on the successful use of a library with diagnostic problem-solving methods for building a knowledge-based system for diagnosing chemical production processes [SA97, SA98]. A road traffic management knowledge-based system [MHC98] is operational in the cities of Madrid and Barcelona in Spain. IBM, Japan reports a knowledge system for job scheduling of production processes [HY98]. The system has been built by using a domain-oriented library of scheduling problem-solving methods. Metrics show that a significant percentage of existing code has been reused in the new application. Knowledge-based systems for plant classification, service support for printing machines, and rheumatology have been developed from reusable methods, as reported in [Pup98].

4 Conclusions and future work

In this paper, we reviewed recent work in the area of ontologies [ST99, UT98, vSW97, GP95] and problem-solving methods [BF98]. The current state-of-the-art is that there is now a fairly good understanding of what ontologies are and what they do. Current work takes this body of existing work and starts from that in new directions.

In the ontology world, emphasis is now put on integration of heterogeneous ontologies and on characterizing and brokering ontologies on the WWW. Also, efforts are made to connect to the object-oriented world and to databases. Ontologies are clearly becoming more and more important in a large variety of areas.

Also, work in the problem-solving method world moves on. Several libraries of methods exist and

efforts are made to make these libraries accessible and interoperable. The European IBROW project (http://www.swi.psy.uva.nl/projects/IBROW3/home.html) aims at building a brokering service that can configure knowledge system out of reusable PSMs that reside in libraries on the Internet.

The integration of ontologies and PSMs is also a promising new direction, especially when ontologies and databases are integrated. The PSMs can then provide reasoning services on top of these databases, which can lead to dynamically configured active databases rather than passive repositories of static knowledge waiting to be queried.

5 Relevant links

For an extensive collection of (alphabetically ordered) links to work on ontologies and problem-solving methods, including proceedings and events, see: http://www.cs.utexas.edu/users/mfkb/related.html. The homepage of the PSM mailing list can be accessed at: http://www.swi.psy.uva.nl/mailing-lists/kaw-psm/home.html.

A list of relevant workshops that are accessible on the WWW is included below:

- Applications of Ontologies and Problem-Solving Methods, ECAI'98 (European Conference on AI), http://delicias.dia.fi.upm.es/WORKSHOP/ECAI98/index.html
- Building, Maintaining, and Using Organizational Memories, ECAI'98, http://www.aifb.uni-karlsruhe.de/WBS/ECAI98OM/
- Formal Ontologies in Information Systems (FOIS'98), http://krr.irst.itc.it:1024/fois98/program.html
- Intelligent Information Integration, ECAI'98, http://www.tzi.de/grp/i3/ws-ecai98/
- Sharable and Reusable Components for Knowledge Systems, KAW'98 (Workshop on Knowledge Acquisition, Modeling, and Management), http://ksi.cpsc.ucalgary.ca/KAW/KAW98/KAW98Proc.html
- Ontological Engineering, AAAI Spring Symp. Series, Stanford, Calif., 1997, http://www.aaai.org/Symposia/Spring/1997/sss-97.html
- Problem-Solving Methods, IJCAI'97 (Int'l Joint Conf. AI), http://www.aifb.uni-karlsruhe.de/WBS/dfe/PSM/main.html
- Ontological Engineering, ECAI'96, http://wwwis.cs.utwente.nl:8080/kbs/EcaiWorkshop/homepage.html
- Sharable and Reusable Ontologies, KAW'96, http://ksi.cpsc.ucalgary.ca/KAW/KAW96/KAW96Proc.html
- Sharable and Reusable Problem-Solving Methods, KAW'96, http://ksi.cpsc.ucalgary.ca/KAW/KAW96/KAW96Proc.html

References

[ABB+98] G. Aguado, J. Bateman, A. Bañon, S. Bernardos, M. Fernández, A. Gómez-Pérez, E. Nieto, A. Olalla, R. Plaza, and A. Sanchez. ONTOGENERATION: Reusing domain and linguistic ontologies for spanish text generation. In A. Gomez-Perez and V. R. Benjamins, editors, *Proceedings of the Workshop on Applications of Ontologies and Problem-Solving Methods, held in conjunction with ECAI-98*, pages 1–10, Brighton, UK, August 1998. ECAI.

[Abe93] M. Aben. Formally specifying re-usable knowledge model components. *Knowledge Acquisition*, 5:119–141, 1993.

[Abe95] M. Aben. *Formal Methods in Knowledge Engineering*. PhD thesis, University of Amsterdam, Amsterdam, 1995.

[AFS98] J. Angele, D. Fensel, and R. Studer. Developing knowledge-based systems with mike. *Journal of Automated Software Engineering*, to appear, 1998.

[AGLP98] J. Arpirez, A. Gómez-Pérez, A. Lozano, and S. Pinto. (onto)2agent: An ontology-based www broker to select ontologies. In A. Gomez-Perez and V. R. Benjamins, editors, *Proceedings of the Workshop on Applications of Ontologies and Problem-Solving Methods, held in conjunction with ECAI-98*, pages 16–24, Brighton, UK, August 1998. ECAI.

[BA97] V. R. Benjamins and M. Aben. Structure-preserving KBS development through reusable libraries: a case-study in diagnosis. *International Journal of Human-Computer Studies*, 47:259–288, 1997.

[Bat94] J. A. Bateman. KPML: The KOMET-Penman (multilingual) development environment. Technical report, GMD/IPSI, Darmstadt (Germany), 1994.

[BBvH96] P. Beys, V. R. Benjamins, and G. van Heijst. Remedying the reusability-usability tradeoff for problem-solving methods. In B. R. Gaines and M. A. Musen, editors, *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, pages 2.1–2.20, Alberta, Canada, 1996. SRDG Publications, University of Calgary.

[BC88] 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, editors, *Knowledge Acquisition for Knowledge Based Systems*, volume 1, pages 65–77. Academic Press, London, 1988.

[Ben93] V. R. Benjamins. *Problem Solving Methods for Diagnosis*. PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, 1993.

[Ben95] V. R. Benjamins. Problem-solving methods for diagnosis and their role in knowledge acquisition. *International Journal of Expert Systems: Research and Applications*, 8(2):93–120, 1995.

[BF98] V. R. Benjamins and D. Fensel. Editorial: Problem-solving methods. *International Journal of Human-Computer Studies*, 49(4):305–313, October 1998. Special issue on Problem-Solving Methods.

[BFDGP97] V. R. Benjamins, D. Fensel, S. Decker, and A. Gomez-Perez. $(KA)^2$: Building ontologies for the internet: a mid term report. *International Journal of Human-Computer Studies*, page in press, 1997.

[BFGPGP98] M. Blázquez, M. Fernández, J. M. García-Pinar, and A. Gómez-Pérez. Building ontologies at the knowledge level using the ontology design environment. In *Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management, KAW'98*, Banff, Canada, 1998.

[BFS96] 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.

[BGM96] S. Borgo, N. Guarino, and C. Masolo. Stratified ontologies: the case of physical objects. In *Proceedings of the Workshop on Ontological Engineering, held in conjunction with ECAI-96*, pages 5–15, Budapest, August 1996. ECAI.

[BH95] J. P. Bowen and M. G. Hinchey. Ten commands of formal methods. *IEEE Computer*, 28(4):56–63, 1995.

[BHB97] Barros, L. Nunes de, J. Hendler, and V. R. Benjamins. Par-KAP: a knowledge acquisition tool for building practical planning systems. In M. E. Pollack, editor, *Proc. of the 15th IJCAI*, pages 1246–1251, Japan, 1997. International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers, Inc. Also published in Proceedings of the Ninth Dutch Conference on Artificial Intelligence, NAIC'97, K. van Marcke, W. Daelemans (eds), University of Antwerp, Belgium, pages 137–148.

[BKMW90] J. A. Bateman, R. T. Kasper, J. D. Moore, and R. A. Whitney. A general organization of knowledge for natural language processing: the penman upper model. Technical report, USC/ISI, Marina del Rey, CA (USA), 1990.

[BLC96] A. Bernaras, I. Laresgoiti, and J. Corera. Building and reusing ontologies for electrical network applications. In *Proceedings of the 12th ECAI*, pages 298–302, 1996.

[BMF95] J. A. Bateman, B. Magnini, and G. Fabris. The generalized upper model knowledge bare: Organization and use. In N. J. I. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing.*, pages 60–72. IOS Press, Amsterdam, NL, 1995.

[Bor97] W. N. Borst. *Construction of Engineering Ontologies*. PhD thesis, University of Twente, Enschede, 1997.

[BPG96] V. R. Benjamins and C. Pierret-Golbreich. Assumptions of problem-solving methods. In N. Shadbolt, K. O'Hara, and G. Schreiber, editors, *Lecture Notes in Artificial Intelligence, 1076, 9th European Knowledge Acquisition Workshop, EKAW-96*, pages 1–16, Berlin, 1996. Springer-Verlag.

[Bre94a] J. Breuker. Components of problem solving and types of problems. In L. Steels, A. T. Schreiber, and W. van de Velde, editors, *Lecture Notes in Artificial Intelligence, 867, 8th European Knowledge Acquisition Workshop, EKAW-94*, pages 118–136, Berlin, Germany, 1994. Springer-Verlag.

[Bre94b] J. Breuker. A suite of problem types. In J. Breuker and W. van de Velde, editors, *CommonKADSLibrary for Expertise Modeling*, pages 57–87. IOS Press, Amsterdam, The Netherlands, 1994.

[BVB96] Barros, L. Nunes de, A. Valente, and V. R. Benjamins. Modeling planning tasks. In *Third International Conference on Artificial Intelligence Planning Systems, AIPS-96*, pages 11–18. American Association of Artificial Intelligence (AAAI), 1996.

[BvdV94] J. Breuker and W. van de Velde, editors. *CommonKADS Library for Expertise Modeling*. IOS Press, Amsterdam, The Netherlands, 1994.

[CFF+97] V. K. Chaudhri, A. Farquhar, R. Fikes, P. Karp, and J. Rice. The generic frame protocol 2.0. Technical report, Stanford, 1997.

[Cha90] B. Chandrasekaran. Design problem solving: A task analysis. *AI Magazine*, 11:59–71, 1990.

[CJB99] B. Chandrasekaran, J.R. Josephson, and V.R. Benjamins. Ontologies: What are they? why do we need them? *IEEE Intelligent Systems and Their Applications*, 14(1):20–26, 1999. Special Issue on Ontologies.

[CJS92] B. Chandrasekaran, T. R. Johnson, and J. W. Smith. Task-structure analysis for knowledge modeling. *Communications of the ACM*, 35(9):124–137, 1992.

[Cla85] W. J. Clancey. Heuristic classification. *Artificial Intelligence*, 27:289–350, 1985.

[DMW93] J. Domingue, E. Motta, and S. Watt. The emerging VITAL workbench. In Aussenac et al., editor, *EKAW'93 Knowledge Acquisition for Knowledge-Based Systems. Lecture Notes in Artificial Intelligence, LNCS 723*, Berlin, Germany, 1993. Springer-Verlag.

[Dom98] J. Domingue. Tadzebao and webonto: Discussing, browsing, and editing ontologies on the web. In *Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management, KAW'98*, Banff, Canada, 1998.

[FB98] D. Fensel and V. R. Benjamins. The role of assumptions in knowledge engineering. *International Journal on Intelligent Systems (IJIS)*, 13(7):715–748, 1998.

[FBMW99] D. Fensel, V. R. Benjamins, E. Motta, and B. Wielinga. UPML: A framework for knowledge system reuse. In *Proceedings of the 16th International Joint Conference on AI (IJCAI-99)*, page to appear, Sweden, 1999.

[FDES98] D. Fensel, S. Decker, M. Erdmann, and R. Studer. Ontobroker: The very high idea. In *Proceedings of the 11th International Flairs Conference (FLAIRS-98)*, Sanibal Island, Florida, 1998.

[FFR97] A. Farquhar, R. Fikes, and J. Rice. The ontolingua server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707–728, June 1997.

[FG97] 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*, pages 349–357, Madrid, 1997.

[FGPPP99] M. Fernandez, A. Gomez-Perez, J. Pazos, and Alejandro Pazos. Ontology of tasks and methods. *IEEE Intelligent Systems and Their Applications*, 14(1):37–46, January/February 1999.

[FS97] D. Fensel and A. Schönegge. Using KIV to specify and verify architecturs of knowlege-based systems. In *Proc. of 12th IEEE International Conference on Automated Software Engineering (ASEC-97)*. IEEE, 1997.

[FS98] D. Fensel and R. Straatman. The essence of problem-solving methods: making assumptions to gain efficiency. *IJHCS*, 48:181–215, 1998.

[FvdR98] M. Frohlich and R. P. van de Riet. Using multiple ontologies in a framework for natural language generation. In A. Gomez-Perez and V. R. Benjamins, editors, *Proceedings of the Workshop on Applications of Ontologies and Problem-Solving Methods, held in conjunction with ECAI-98*, pages 67–77, Brighton, UK, August 1998. ECAI.

[GF92] 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.

[GF95] M. Grüninger and M. Fox. Methodology for the design and evaluation of ontologies. In *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing held in conjunction with IJCAI-95*, Montreal, Canada, 1995.

[GG95] N. Guarino and P. Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In N. J. I. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing*, pages 25–32. IOS Press, Amsterdam, NL, 1995.

[GGM98] J. H. Gennari, W. Grosso, and M. Musen. A method-description language: An initial ontology with examples. In B. R. Gaines and M. A. Musen, editors, *Proceedings of the 11th Banff Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, pages SHARE.11.–SHARE.11.18, Alberta, Canada, 1998. SRDG Publications, University of Calgary.

[GHJV95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.

[GMV99] N. Guarino, C. Masolo, and G. Vetere. Ontoseek: Using large linguistic ontologies for accessing online yellow pages and product catalogs. *IEEE Intelligent Systems and Their Applications*, page to appear, 1999.

[GO94] T. Gruber and R. Olsen. An ontology for engineering mathematics. Technical report, Knowledge Systems Laboratory, Stanford University, CA, 1994.

[GP95] N. Guarino and R. Poli. The role of ontology in the information technology. *International Journal of Human-Computer Studies*, 43(5/6):623–965, 1995. Special issue on ontology.

[GP98] A. Gómez-Pérez. Knowledge sharing and reuse. In J. Liebowitz, editor, *The Handbook of Applied Expert Systems*. CRC, 1998.

[GPR99] A. Gomez-Perez and M. D. Rojas. Ontological reengineering for reuse. In D. Fensel and R. Studer, editors, *Proceedings of the EKAW-99*, page to appear. Springer-Verlag, 1999.

[Gru93] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.

[Gru95] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43:907–928, 1995.

[GTRM94] J. H. Gennari, S. W. Tu, T. E. Rotenfluh, and M. A. Musen. Mapping domains to methods in support of reuse. *International Journal of Human-Computer Studies*, 41:399–424, 1994.

[Gua98] N. Guarino. Some ontological principles for designing upper level lexical resources. In *Proceedings of the First International Conference on Language Resources and Evaluation*, pages –, Granada, 1998.

[Hoe98] E. Hoenkamp. Spotting ontological lacunae through spectrum analysis of retrieved documents. In A. Gomez-Perez and V. R. Benjamins, editors,

Proceedings of the Workshop on Applications of Ontologies and Problem-Solving Methods, held in conjunction with ECAI-98, pages 73–77, Brighton, UK, August 1998. ECAI.

[HY98] M. Hori and T. Yoshida. domain-oriented library of scheduling methods: design principal and real-life application. *International Journal of Human-Computer Studies*, 49(4):601–626, 1998. Special issue on Problem-Solving Methods.

[KBD+91] G. Klinker, C. Bhola, G. Dallemagne, D. Marques, and J. McDermott. Usable and reusable programming constructs. *Knowledge Acquisition*, 3:117–136, 1991.

[KL94] K. Knight and S. Luk. Building a large knowledge base for machine translation. In *AAAI–94*, 1994.

[KLW95] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 1995.

[LG90] D. B. Lenat and R. V. Guha. *Building large knowledge-based systems. Representation and inference in the Cyc project*. Addison-Wesley, Reading, Massachusetts, 1990.

[Mac91] R. MacGregor. Inside the LOOM classifier. *SIGART Bulletin*, 2(3):70–76, June 1991.

[Mah96] K. Mahesh. Ontology development for machine translation: Ideology and methodology. Technical report, Computer Research Laboratory, New Mexico State University, 1996.

[Mar88a] S. Marcus, editor. *Automating knowledge acquisition for expert systems*. Kluwer, Boston, 1988.

[Mar88b] S. Marcus. SALT: a knowledge-acquisition tool for propose-and-revise systems. In S. Marcus, editor, *Automating Knowledge Acquisition for Expert Systems*, pages 81–123. Kluwer, Boston, 1988.

[MHC98] M. Molina, J. Hernández, and J. Cuena. A structure of problem-solving methods for real-time decision support in traffic control. *International Journal of Human-Computer Studies*, 49(4):577–600, 1998. Special issue on Problem-Solving Methods.

[Mil90] G. A. Miller. WORDNET: an online lexical database. *International Journal of Lexicography*, 3(4):235–312, 1990.

[Mus93] M. Musen. An overview of knowledge acquisition. In J. M. David, J. P. Krivine, and R. Simmons, editors, *Second Generation Expert Systems*. Springer Verlag, 1993.

[MVI95] R. Mizoguchi, J. Vanwelkenhuysen, and M. Ikeda. Task ontology for reuse of problem solving knowledge. In N. J. I. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing.*, pages 46–57. IOS Press, Amsterdam, NL, 1995.

[MZ98] E. Motta and Z. Zdrahal. A library of problem-solving components based on the integration of the search paradigm with task and method ontologies. *International Journal of Human-Computer Studies*, 49(4):437–470, 1998. Special issue on Problem-Solving Methods.

[NFF+91] R. Neches, R. E. Fikes, T. Finin, T. R. Gruber, T. Senator, and W. R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, 1991.

[Ors96a] K. Orsvärn. *Knowledge Modelling with Libraries of Task Decomposition Methods*. PhD thesis, Swedish Institute of Computer Science, 1996.

[Ors96b] K. Orsvärn. Principles for libraries of task decomposition methods – conclusions from a case-study. In N. Shadbolt, K. O'Hara, and G. Schreiber, editors, *Lecture Notes in Artificial Intelligence, 1076, 9th European Knowledge Acquisition Workshop, EKAW-96*, pages 48–65. Springer-Verlag, 1996.

[PETM92] A.R. Puerta, J. Egar, S. Tu, and M. Musen. A multiple-method shell for the automatic generation of knowledge acquisition tools. *Knowledge Acquisition*, 4:171–196, 1992.

[PG98] C. Pierret-Golbreich. Supporting organization and use of problem-solving methods libraries by formal methods. *International Journal of Human-Computer Studies*, 49(4):471–495, 1998. Special issue on Problem-Solving Methods.

[Pup98] F. Puppe. Knowledge reuse among diagnostic problem-solving methods in the shell-kit D3. *International Journal of Human-Computer Studies*, 49(4):627–649, 1998. Special issue on Problem-Solving Methods.

[Ros94] D. Rosner. Generating multilingual documents from a knowledge base: The techdoc project. Technical report, FAW Ulm, Ulm (Germany), 1994.

[SA97] P-H. Speel and M. Aben. Applying a library of problem solving methods on a real-life task. *International Journal of Human-Computer Studies*, 46(May):627–652, 1997.

[SA98] P-H. Speel and M. Aben. Preserving conceptual structures in design and implementation of industrial KBSs. *International Journal of Human-Computer Studies*, 49(4):547–575, 1998. Special issue on Problem-Solving Methods.

[SAA+99] A. Th. Schreiber, J. M. Akkermans, A. A. Anjewierden, R. de Hoog, N. R. Shadbolt, W. Van de Velde, and B. J. Wielinga. *Engineering and Managing Knowledge, The CommonKADS methodology*. MIT Press, 1999.

[SBF98] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering, principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197, 1998.

[SCC+93] O. Stock, G. Carenini, F. Cecconi, E. Franconi, A. Lavelli, B. Magnini, F. Pianesi, M. Ponzi, V. Samek-Lodovici, and C. Strapparava. Alfresco:

Enjoying the combination of natural language processing and hypermedia for information exploration. In Mark T. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 9, pages 197–224. The MIT Press, 1993.

[SG95] W. Swartout and Y. Gil. Expect: Explicit representations for flexible acquisition. In *Proceedings of the Ninth Knowledge Acquisition for Knowledge-Based Systems Workshop*, 1995.

[Sow97] J. F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Book draft, 1997.

[SPKR97] W. Swartout, R. Patil, K. Knight, and T. Russ. Toward distributed use of large-scale ontologies. In *Spring Symposium Series on Ontological Engineering*, pages 33–40, Stanford, 1997. AAAI Press.

[ST99] B. Swartout and A. Tate. Coming to terms with ontologies. *IEEE Intelligent Systems and Their Applications*, 14(1):18–19, 1999. Special Issue on Ontologies.

[Ste90] L. Steels. Components of expertise. *AI Magazine*, 11(2):28–49, Summer 1990.

[Ste93] L. Steels. The componential framework and its role in reusability. In Jean-Marc David, Jean-Paul Krivine, and Reid Simmons, editors, *Second Generation Expert Systems*, pages 273–298. Springer-Verlag, Berlin Heidelberg, Germany, 1993.

[SWB93] A. Th. Schreiber, B. J. Wielinga, and J. A. Breuker, editors. *KADS: A Principled Approach to Knowledge-Based System Development*, volume 11 of *Knowledge-Based Systems Book Series*. Academic Press, London, 1993.

[SWdH+94] A. Th. Schreiber, B. J. Wielinga, R. de Hoog, J. M. Akkermans, and W. Van de Velde. CommonKADS: A comprehensive methodology for KBS development. *IEEE Expert*, 9(6):28–37, December 1994.

[SWJ95] A. Th. Schreiber, B. J. Wielinga, and W. H. J. Jansweijer. The KACTUS view on the 'O' word. In *IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[tT97] A. ten Teije. *Automated Configuration of Problem Solving Methods in Diagnosis*. PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, 1997.

[tTvHSW98] A. ten Teije, F. van Harmelen, A. Th. Schreiber, and B. Wielinga. Construction of problem-solving methods as parametric design. *International Journal of Human-Computer Studies*, 49(4):363–389, 1998. Special issue on Problem-Solving Methods.

[TvHWS93] P. Terpstra, G. van Heijst, B. Wielinga, and N. Shadtbolt. Knowledge acquisition support through generalised directive models. In Jean-Marc David, Jean-Paul Krivine, and Reid Simmons, editors, *Second Generation Expert Systems*, pages 428–455. Springer-Verlag, Berlin Heidelberg, Germany, 1993.

[UG96] M. Uschold and M. Gruninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[Usc96] M. Uschold. Building ontologies: Towards a unified methodology. In *Expert Systems 96*, 1996.

[UT98] M. Uschold and A. Tate. Special issue on ontologies. *The Knowledge Engineering Review*, 13(1), 1998.

[vdVSM94] P. E. van de Vet, P.-H. Speel, and N. J. I. Mars. The plinius ontology of ceramic materials. In N. J. I. Mars, editor, *Working papers European Conference on Artificial Intelligence ECAI'94 Workshop on Implemented Ontologies*, pages 187–206, Amsterdam, 1994. ECCAI.

[VL93] A. Valente and C. Löckenhoff. Organization as guidance: A library of assessment models. In *Proceedings of the Seventh European Knowledge Acquisition Workshop (EKAW'93), Lecture Notes in Artificial Intelligence, LNCS 723*, pages 243–262, 1993.

[vSW97] G. van Heijst, A. T. Schreiber, and B. J. Wielinga. Using explicit ontologies in KBS development. *International Journal of Human-Computer Studies*, 46(2/3):183–292, 1997.

[WAS98] B. J. Wielinga, J. M. Akkermans, and A. Th. Schreiber. A competence theory approach to problem solving method construction. *International Journal of Human-Computer Studies*, 49(4):315–338, 1998. Special issue on Problem-Solving Methods.

[WMK95] J. Dukes-Schlossberg W. Mark and R. Kerber. Ontological commitments and domain-specific architectures: Experience with comet and cosmos. In N. J. I. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing.*, pages 33–44. IOS Press, Amsterdam, NL, 1995.