Reusable and Shareable Knowledge Bases: A European Perspective*

B. J. Wielinga A. Th. Schreiber

University of Amsterdam, Social Science Informatics
Roetersstraat 15, NL-1018 WB Amsterdam, The Netherlands
E-mail: {wielinga,schreiber}@swi.psy.uva.nl

Abstract

Large amounts of knowledge are available in many knowledge bases for a variety of applications. This knowledge is however, not reusable since it is application specific. This paper discusses the problem of making knowledge shareable over applications and resuing it. Three principles are formulated that can for a basis for a methodology for designing sharable knowledge bases. The separation of domain and control knowledge, the explication of meta-models of the domain knowledge (ontologies), and the distinction between ontologies that submit to different classes of commitments are described as ways of achieving sharable and reusable knowledge bases.

*The research reported here was carried out in the course of the KADS-II project. This project is partially funded by the ESPRIT Programme of the Commission of the European Communities as project number 5248. The partners in this project are: Cap Gemini Innovation (F), Cap Programator (S), Netherlands Energy Research Foundation ECN (NL), Eritel SA (ESP), IBM France (F), Lloyd's Register (UK), Swedish Institute of Computer Science (S), Siemens AG (D), Touche Ross MC (UK), University of Amsterdam (NL) and Free University of Brussels (B). This article expresses the opinions of the authors and not necessarily those of the consortium.

1 Introduction

Knowledge technology is becoming of age: operational knowledge-based systems are used routinely in many companies and organisations, methods for developing knowledge-based systems are being consolidated [Schreiber *et al.*, 1993b] and many small and large knowledge bases have been built for a variety of applications. A major problem with current knowledge technology is however that these knowledge bases -however valuable their content may be- are usually idiosyncratic, i.e. they are designed to serve a specific application task and have little or no generality. It is a major challenge to investigate the question how this growing body of formalised knowledge can be reformulated in such a way that it could be (re)used for more general purposes.

In a similar vein there exist scientific disciples where an enormous amount of knowledge is being gathered, but where the accessibility of that knowledge is becoming more and more of a problem. For example, researchers in molecular biology are gathering large amounts of information about DNA sequences in the genetic material of many organisms. Although this information is currently stored in large databases that are world-wide accessible, the actual knowledge in these databases is often difficult to obtain in a form that is readily usable. In other scientific fields similar problems exist.

Thus, the challenge for the knowledge engineering community is to find principles and techniques for developing and modelling large bodies of knowledge in such a way that they are amenable to reuse, accessibility and sharing.

This problem has been subject of research for some time in the US. The DARPA Knowledge Sharing Effort [Neches *et al.*, 1991] is investigating how knowledge that is present in individual knowledge based applications, can be made available to other applications through the use of an intermediate language that allows knowledge exchange (KIF: Knowledge Interchange Format).

Where most efforts in the US have focused on issues of knowledge representation and translation, European research in the field of knowledge technology has concentrated on methods for knowledge acquisition, knowledge modelling and structured KBS development. Although a large European project (KACTUS) on reusable knowledge bases in technical domains is currently being negotiated, the problem of knowledge sharing has had only secondary attention in Europe. However the work on knowledge based systems within the Esprit programme has achieved results that are relevant to the problem at hand. In this paper we will investigate how the results of European research in the field of knowledge technology can contribute to the problem of knowledge sharing and how it relates to the American initiatives. The paper formulates a number of principles that are in our view essential in any knowledge sharing approach. In doing so we will adopt a certain bias, i.e. that of the KADS approach to knowledge modelling [Schreiber *et al.*, 1993b].

2 Structuring Knowledge Systems

The first principle that underlies most work on knowledge sharing and reuse, is that of distinguishing different types of knowledge according to their function. Where first generation expert systems represented factual knowledge about the application domain and knowledge about how to organise the reasoning process in a mixed representation, i.e. rules, current knowledge modelling approaches distinguish between two basic types of knowledge: knowledge about the application domain and knowledge about the task that needs to be performed. The *application domain knowledge* (or domain knowledge for short) is static in nature, it consists of the concepts, relations and facts that are needed to reason about a certain application domain. The *control knowledge* describes *how* reasoning with the application domain knowledge is done in terms of elementary reasoning operations on the domain knowledge and in terms of control structures and decompositions of the task as a whole. Fig. 1 schematically represents this view on knowledge systems.

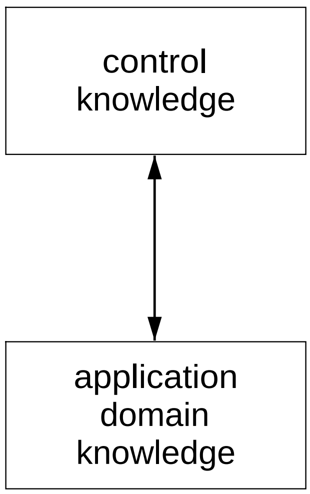

FIGURE 1: Two types of knowledge in a KBS

The KADS approach to knowledge-based system development [Wielinga *et al.*, 1992;

Schreiber *et al.*, 1993b] has refined the two-tier model that was among others proposed by Clancey [Clancey, 1983] and Steels [Steels, 1990]. KADS distinguishes between *inference knowledge*, knowledge about how to use the application domain knowledge in elementary reasoning steps, and *task knowledge* describing how to decompose the top-level reasoning task and how to impose control on the elementary operations.

Knowledge about how to control the reasoning process of a problem solver can come in one of two forms: a fully instantiated model of how to perform a reasoning process for a specific application or in the form of knowledge that describes how to decompose a problem into subproblems and how to achieve primitive subproblems [Benjamins, 1993]. We will call the former a *task model* and the latter a set of *problem solving methods*. Thus, a problem solving method can be viewed as a prescription of how to construct part of a task model, and a task model as a fully instantiated set of problem solving methods.

In the context of knowledge sharing, the separation of the application domain knowledge and the task related knowledge, gives rise to an important question: what are the dependencies between the two parts of the model. Chandrasekaran formulated the *interaction hypothesis* which states that task knowledge and application domain knowledge are highly dependent: one cannot define the application domain knowledge without knowing what the task model is going to be, and vice vera. Domain knowledge and task knowledge must be packaged [Chandrasekaran, 1989]. Early work on KADS [Wielinga & Breuker, 1986], on the other hand, worked under the assumption that application domain knowledge can be formulated independent of the task model.

There is a growing consensus that some interaction between the application domain knowledge and the task model exists, but that different types of such an interaction can be distinguished. Below we will discuss this idea in more detail.

Another, not unrelated problem arises when we want to couple different task models (or problem solvers) to the same application domain knowledge base. Different problem solvers often require different representations. This problem has been investigated in two large Esprit projects on knowledge acquisition and machine learning: MLT [Morik, 1989] and ACKnowledge [Anjewierden *et al.*, 1993]. Both projects were faced with the problem of combining and reusing knowledge generated by knowledge acquisition and machine learning programs that employ specific representational structures. For example, how can one combine a concept hierarchy acquired through a laddered grid tool [Major & Reichgelt, 1990] with rules that are generated by an AQ-type learning program. Several solutions were attempted in both projects. MLT defined a Core Knowledge Representation Language (CKRL) that should function as an intermediate representation between the various machine learning tools. Each tool should provide its output in the CKRL such that it could be used by other acquisition tools. In the Knowledge Engineering Workbench (KEW) built in ACKnowledge, a CKRL was developed as a hybrid language combining a frame language and predicate logic. Special transformation tools were developed to translate between the two languages. The solutions only worked to a small extent: it turned out that the problem was not so much in the representation language, but in the ontological commitments that individual tools and problem solvers imposed on the knowledge that they generate or require. These commitments were only partially explicated in the systems. In KEW the ontological commitments of the task of the problem solver were explicated in a top-down refinement process and these commitments were matched against those of the tools. Although this approach was successful for a number of scenario's, it lacked a structured theory of such ontological commitments to be generally applicable to the knowledge sharing problem.

In summary, while the separation of application domain knowledge is a first step towards building reusable and sharable knowledge bases, it poses several problems that stem from the ontological commitments embodied in the application domain knowledge and those required by the task model.

3 Meta-models of domain knowledge

When the KADS research hit the problem of interfacing the domain knowledge to the inference knowledge -a link that initially was left obscure- the solution was found [Schreiber, 1992] in building a meta-model, called domain view, of the structure of the domain knowledge and use this meta-model as a means of interfacing classes of domain knowledge statements to the inference operators. The *domain view* specifies how particular parts of the domain theory can be used as a "body of knowledge" by an inference.

Fig. 2 describes a primitive inference in a simple example domain (audio system diagnosis) with example references to domain knowledge. At the inference level a *decomposition* inference is specified. The action that is performed in this inference is the decomposition of a composite model of the audio system into sub-models. *System model* and *hypothesis* are examples of knowledge roles. They describe the role that domain objects like `audio-system` and `amplifier` can play in the problem-solving process. The *decompose* inference achieves its goal, the generation of a new hypothesis, through the application of decomposition knowledge. The domain view of this inference specifies that tuples of the `SUB-COMPONENT-OF` relation in the domain theory can be used as decomposition knowledge. Fig. 2 shows one applicable tuple of this relation.

In more recent work in the KADS-II project [Wielinga *et al.*, 1993] a full language was developed for meta-modelling the application domain knowledge: CML. Although parts of the CML definitions are specified in natural language, it has a very expressive syntax and has proved to be a powerful tool for meta-modelling.

Fig. 3 shows the basic modelling constructs of the CML. These constructs can be used both for typing the application domain knowledge as well as for meta modelling.

As example, Fig. 4 shows a graphic representation of part of the structure of a knowledge base for simple diagnosis of cars. The ovals *manifestation* and *state* model expressions in the knowledge base that involve the property *observable* and *state variable* of the concept class *car component*. Diamonds represent definitions of two relations (*cause* and *has manifestation*) that relate these different expressions. These relation define the structure of the statements that populate the application domain knowledge base.

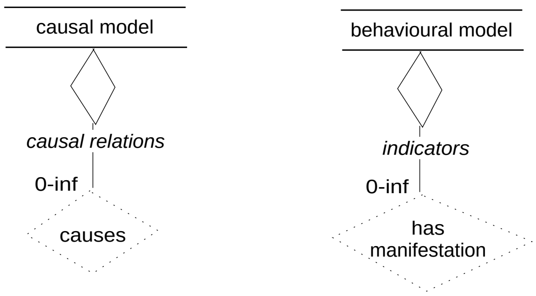

FIGURE 5: CML meta-model of two domain knowledge bases

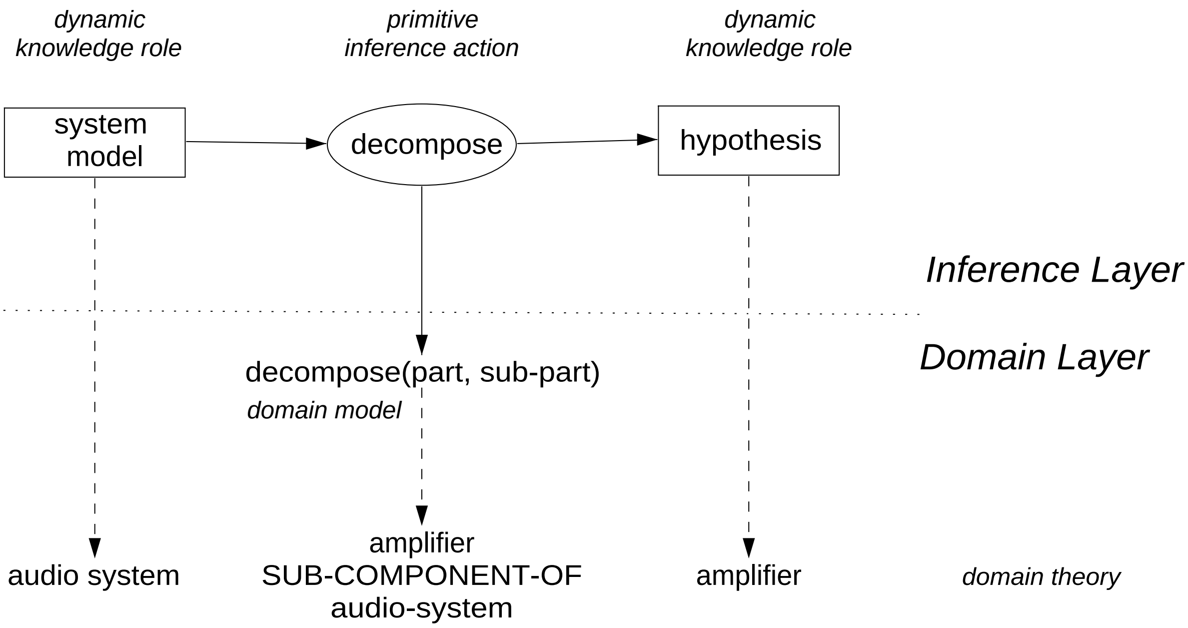

FIGURE 2: A primitive inference performing an decomposition action

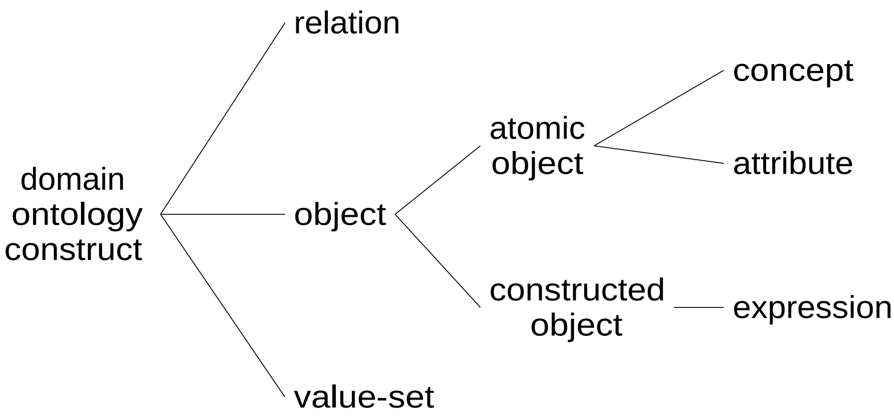

FIGURE 3: Hierarchy of object types in the CommonKADS conceptual modelling language

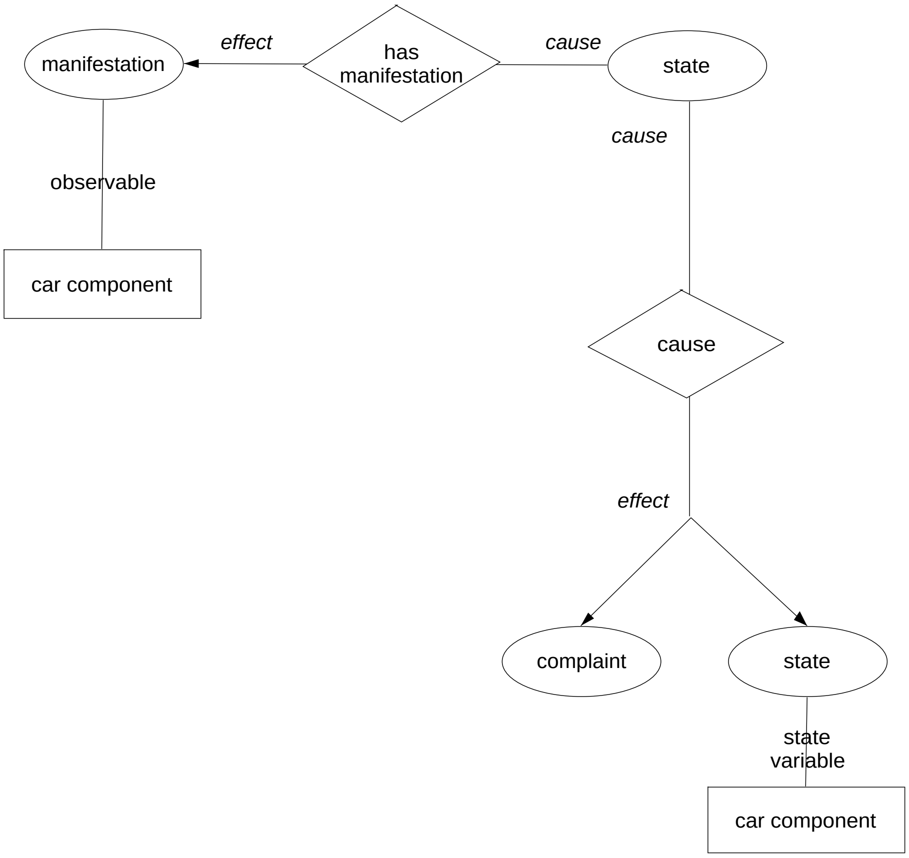

FIGURE 4: CML meta-model of relations

Fig. 5 shows the meta-model in the CML of two domain knowledge base partitions (domain models). The behaviour model is build from relational tuples that describe a causal relation between a state variable expression and an expression involving an observable property of a car component. An example of such a tuple is:

(power OF battery = low)
HAS-MANIFESTATION
(battery-dial OF control-panel = zero)

The second domain model is a causal model of instances of causal relations between states. The example shows that using the CML, partitions of the knowledge base can be described at a high level of abstraction.

Although the goals of the KADS meta-modelling language differed from those in the Knowledge Sharing Effort in the US, the similarities between Gruber's work on Ontolingua and KADS are striking. In both approaches an abstracted (meta-) model is build of the application domain knowledge, describing its structure and (meta-) properties. Ontolingua initially aimed at using the meta- model for translations between representations, while the KADS meta-model aims at specifying the constraints between inference and domain knowledge. As it turns out, [Schreiber *et al.*, 1994] Ontolingua models and KADS meta-models can be used for both purposes. In addition they appear to provide a powerful framework for specifying ontological commitments of knowledge acquisition techniques and tools.

If meta-models of the application domain knowledge are a way of attacking the knowledge sharing problem, what do these models look like and how do we build them?.

4 Ontologies as meta-models

The commonly used term for meta-models describing the structure of a knowledge base is *Ontology*. In some ways this term is unfortunate, since it has a well defined meaning in philosophy that differs from the sense in which the knowledge engineering community uses it. In the philosophical tradition an Ontology is a theory about that what can exist. In the context of knowledge-based system research not many people claim to say anything about what exists in the real world. We do however, make claims about what knowledge entities are manipulated by our programs when they perform a task. So, we could rephrase the traditional definition of the term "ontology" as follows:

> *An (AI-) Ontology is a theory of what entities can exist in the mind of a knowledgeable agent.*

What kind of theory is an ontology? A knowledge base can be viewed as a model of some part of the world, that allows us - given some inference mechanism- to reason about that world. The model is described in some language, that has a vocabulary and a syntax. An ontology defines constraints about what can be expressed in the model, in addition to the constraints imposed by the syntax. In KADS the ontology defines the vocabulary and the structure of the statements that express elements of the model. It does not define the full semantics of the domain model. Ontolingua takes similar decisions: in Ontolingua the structure of the terms that can occur in a knowledge base are defined, as well as some useful properties of those structures, but not the full range of possible uses that can be made of such a term. In KADS the use of the domain knowledge model is defined in the inference specification, while Ontolingua leaves the inference mechanisms to be specified in the target environment.

How does this help in knowledge sharing? The terminology that the ontology introduces are in essence representations of ontological commitments. So, by explicating the ontology, we also make the ontological commitments explicit and as such enable the definition of procedures that relate, compare or translate the knowledge from one set of commitments to another. Similarly a task model can be applied to an application domain knowledge base, as long as it is compatible with the ontological commitments of that knowledge base (see Fig. 6). Making the ontology of an application knowledge base explicit, is a second principle that supports knowledge sharing and reuse. This principle forms one of the cornerstones of the GAMES project [Schreiber *et al.*, 1993a] that aims at the development of reusable medical knowledge bases and tools that provide libraries of partial ontologies.

However, the full ontology of an application knowledge base will often contain task-specific elements. Thus, in order to make at least part of a knowledge base shareable, partitions have to be identified that are to some extent independent of the task. For that purpose we introduce several types of ontologies and corresponding knowledge bases.

5 Different types of ontologies

Ontological commitments that are the basis for ontology construction come from different sources and have different flavours, hence we can distinguish different types of ontologies. On the lowest level of complexity, we have the definition of the vocabulary of a knowledge

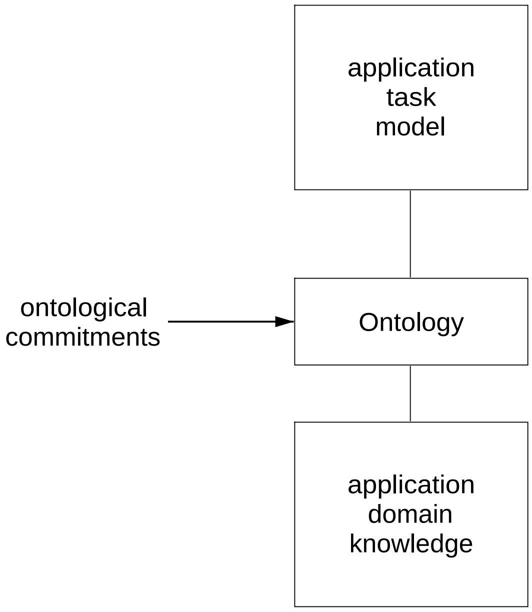

FIGURE 6: Ontology as intermediary between task model and domain knowledge

base. In KADS this part of the ontology is called the *application domain terminology*. If in addition we define a set of types and basic relations such is sub/super class relations, we can define a domain ontology: the extensional description of the domain vocabulary, typology and class hierarchy or lattice. Although the second order terms (such as concept, inheritance-relation, property etc) used in defining the domain ontology can be defined in the meta-ontology, these terms are usually a fixed set of types and relations with their standard interpretation. CML in KADS has a number of basic types (see Fig. 3) while Ontolingua has a similar set defined in the *Frame Ontology*. Gruber calls such an ontology a *representational ontology*: it makes certain commitments with respect to representational constructs, but makes no commitments about the content of the knowledge [Gruber, 1993b].

Although the sharing of domain ontologies is already an important step towards knowledge sharing, it only ensures a commitment to a common vocabulary and typology. It does not imply a consensus on the type of statements that a knowledge base will contain. Such a consensus ontology is what we would call a *domain-model oriented ontology*. This ontology describes general theories within a domain, that are (for all practical purposes) not biased towards particular tasks or methods. For example, in domain concerning devices, there are usually theories about the structure of the device in terms. In that case we can decide to use an ontology based on components with attributes. The goal of reasoning about the structure of a device does imply an ontological commitment, but still leaves the use of the knowledge base large open.

The lower half of Fig. 7 shows an example ontology for components, their properties and component models. This ontology is derived from the KADS-CML representation of the Ontolingua specification for the VT problem [Schreiber *et al.*, 1994]. This part of the ontology is reusable over all tasks that take a structural view on the device.

If we are, however, interested in a class of tasks such as configuration, we require additional knowledge elements to be able to perform the tasks. For example, configuration tasks generally require knowledge about constraints on components (see the grey area in Fig. 7). So, by committing ourselves to a certain class of tasks we introduce additional ontological commitments. The application domain knowledge is extended with a partition that is reusable over a class of tasks, but not usable by any task.

If we take one step further and require the task to be performed through the application of a particular problem solving method, additional requirements on the domain knowledge may come about. For example if we decide to perform the configuration task through the *propose and revise* method, we need knowledge

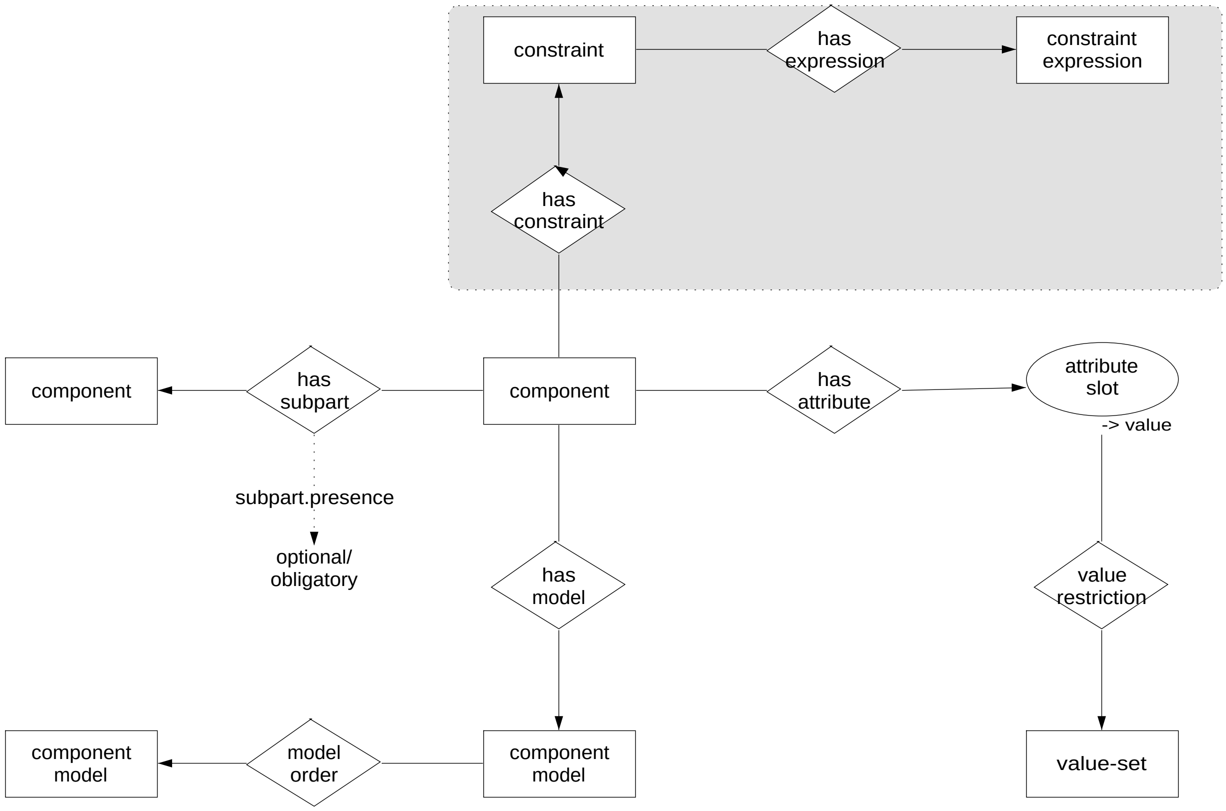

FIGURE 7: Graphical representation of part of the ontology described in [Schreiber *et al.*, 1994]. Lower half describes the ontology for modelling a device structure. The upper grey area is an expansion introduced by the task-type (in this case configuration)

about how to fix violated constraints. Adding this commitment to our ontology makes it not only specific for a class of tasks, but also for the method used.

Fig. 8 summarizes the different sources of ontological commitments and the resulting ontologies:

- Level 1: domain terminology
- Level 2: representational commitments (e.g. class, relation, etc.)
- Level 3: domain-model oriented commitments (e.g. structural models, anatomical models)
- Level 4: task-type oriented commitments (e.g. constraints)
- Level 5: method oriented commitments (e.g. fix knowledge)

The important consequence is that through the use of different ontologies with different generality and by partitioning the application knowledge base accordingly, we can identify different classes of knowledge bases, with different scope, generality and reusability. For example, if one neglects the distinction between task-related and method-related ontologies, this will hamper the reuse of parts of the knowledge base for a similar task but applying a different method.

6 Reusability and Standardisation

Only recently some ideas concerning standardisation in knowledge engineering have been launched. The IMKA initiative in the US

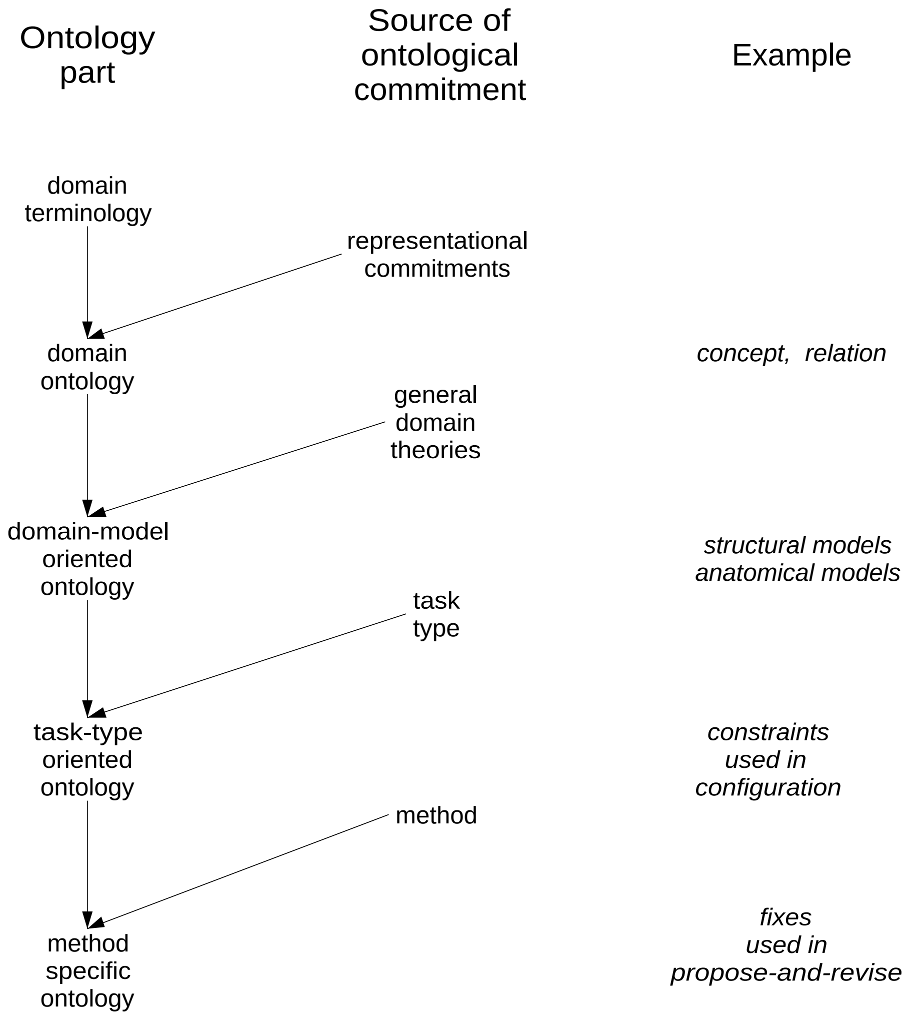

FIGURE 8: Different types of ontological commitments and their corresponding ontologies

aims at standardisation of the representational languages and formalisms used in knowledge-based systems. However useful this effort may be, it will not solve the problems that we have pointed out in this paper. A second -in our line of thinking more fruitful- approach is the development of *standard ontologies* at the various levels that we have discussed above. The basic idea here is that we can develop certain standard conceptualisations of certain areas of knowledge. Gruber [Gruber, 1993a] has developed several ontologies for general domains such as physical quantities, units of measure, bibliographic data etc. Although a lot of debate will be needed before such ontologies are generally accepted (possibly as ISO standards), they are a first step towards unification and standardisation of certain parts of the knowledge bases that are part of the knowledge-based systems that we build. This is essentially the approach that the KACTUS project has adopted and is in line with the EUROKNOWLEDGE initiative that plans to coordinate standardization in knowledge technology.

A third approach concerns the explication and standardisation of problem solving methods. In another paper [Akkermans *et al.*, 1993] we have tried to illustrate how the assumptions and applicability criteria of a problem solving method such as Cover-and Differentiate can be made explicit. Although only few attempts to standardisation of problem solving methods exist to date [Benjamins, 1993], we believe it to be an area of research that can bring significant benefits to the knowledge engineering community.

7 Discussion and Conclusions

In this paper we have discussed the problem of designing knowledge bases that can be shared by many applications, that can be reused or that can be queried by users with a variety of questions.

We have formulated a number of principles that can form the basis for a methodology for building reusable and shareable knowledge bases. The first principle concerns the separation of application domain knowledge and control knowledge. The second principle states that the ontology -viewed as a meta-model- of the domain knowledge should be explicated.

Ontologies have structure: certain parts are based on generally accepted theory, other parts are based on common practice, useful interpretations or on task oriented notions. The third principle for knowledge sharing methodology that we advocate is to distinguish different partial ontologies that are based on different types of ontological commitments. Shareability and Reusability depend critically on the distinctions between the view that underlies the different ontologies: the assumptions and the scope of the ontologies should be made explicit.

References

Akkermans, J. M., Wielinga, B. J., & Schreiber, A. T. (1993). Steps in constructing problem-solving methods. In Aussenac, N., Boy, G., Gaines, B., Linster, M., Ganascia, J.-G., & Kodratoff, Y., editors, *Knowledge Acquisition for Knowledge-Based Systems. Proceedings of the 7th European Workshop EKAW'93, Toulouse and Caylus, France*, number 723 in Lecture Notes in Computer Science, pages 45–65, Berlin Heidelberg, Germany. Springer-Verlag.

Anjewierden, A. A., Wielemaker, J., & Toussaint, C. (1993). Shelley – computer aided knowledge engineering. In Schreiber, A. T., Wielinga, B. J., & Breuker, J. A., editors, *KADS: A Principled Approach to Knowledge-Based System Development*, Knowledge Based Systems, pages 247–264. London, United Kingdom, Academic Press.

Benjamins, V. R. (1993). *Problem Solving Methods for Diagnosis*. PhD thesis, University of Amsterdam, Amsterdam, The Netherlands.

Chandrasekaran, B. (1989). Task–structures, knowledge acquisition and learning. *Machine Learning*, 4(3/4):337–338.

Clancey, W. J. (1983). The epistemology of a rule based system -a framework for explanation. *Artificial Intelligence*, 20:215–251.

Gruber, T. (1993a). Towards principles for the design of ontologies used for knowledge sharing. Technical Report KSL 93-04, Stanford University.

Gruber, T. (1993b). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220.

Major, N. & Reichgelt, H. (1990). ALTO: an automated laddering tool. In Wielinga, B. J., Boose, J., Gaines, B., Schreiber, G., & van Someren, M., editors, *Current trends in knowledge acquisition*, pages 222–236, Amsterdam, The Netherlands. IOS Press.

Morik, K., editor (1989). *Knowledge representation and (re-) organization in Machine Learning*. Lecture Notes, Springer-Verlag.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., & Swartout, W. R. (1991). Enabling technology for knowledge sharing. *AI Magazine*, pages 36–56. Fall.

Schreiber, A. T. (1992). *Pragmatics of the Knowledge Level*. PhD thesis, University of Amsterdam.

Schreiber, A. T., Terpstra, P., Magni, P., & van Velzen, M. (1994). Analysing and implementing VT using COMMON-KADS. In Schreiber, A. T. & Birmingham, W. P., editors, *Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop. Volume 3: Sisyphus II – VT Elevator Design Problem*, pages 44–1 – 44–29, Alberta, Canada. SRDG Publications, University of Calgary.

Schreiber, A. T., van Heijst, G., Lanzola, G., & Stefanelli, M. (1993a). Knowledge organisation in medical KBS construction. In Andreassen, S., Engelbrecht, R., & Wyatt, J., editors, *Proceedins of the 4th Conference on Artificial Intelligence in Medicine Europe, 3-6 October 1993, Munich*, volume 10 of *Studies in Health Technology and Informatics*, pages 394–405, Amsterdam. The Netherlands. IOS Press.

Schreiber, A. T., Wielinga, B. J., & Breuker, J. A., editors (1993b). *KADS: A Principled Approach to Knowledge-Based System Development*, volume 11 of *Knowledge-Based Systems Book Series*. London, Academic Press. ISBN 0-12-629040-7.

Steels, L. (1990). Components of expertise. *AI Magazine*.

Wielinga, B. J. & Breuker, J. A. (1986). Models of expertise. In *Proceedings ECAI–86*, pages 306–318.

Wielinga, B. J., Schreiber, A. T., & Breuker, J. A. (1992). KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition*, 4(1):5–53. Special issue 'The KADS approach to knowledge engineering'. Reprinted in: Buchanan, B. and Wilkins, D. editors (1992), *Readings in Knowledge Acquisition and Learning*, San Mateo, California, Morgan Kaufmann, pp. 92-116.

Wielinga, B. J., Van de Velde, W., Schreiber, A. T., & Akkermans, J. M. (1993). Towards a unification of knowledge modelling approaches. In David, J.-M., Krivine, J.-P., & Simmons, R., editors, *Second Generation Expert Systems*, pages 299–335. Berlin Heidelberg, Germany, Springer-Verlag.