Ontologies in the KRAFT Project

Pepijn R.S. Visser

KRAFT Position Paper KP3 - PRSV 1.2 / Aug96 [1]
Department of Computer Science
University of Liverpool
P.O. Box 147, Liverpool, L69 3BX
E-mail: pepijn@csc.liv.ac.uk

Abstract

KRAFT is a multi-site research project that aims at integrating heterogeneous knowledge, thus creating a distributed information base. This paper outlines the possible roles of ontologies in coping with some of the sharing and reuse problems in the KRAFT project. We start by illustrating problems in sharing and reuse and then discuss ontologies. Thereafter, we outline how ontologies could be applied in the KRAFT project. We then argue that ontologies can play a role in two main areas: (1) knowledge acquisition, and (2) knowledge exchange. Finally, we discuss some design issues with respect to the KRAFT architecture.

1. KRAFT

The expansion of computer networks creates opportunities as well as problems. Often, the possibilities raise problems. One of the possibilities that raises new problems is the sharing and reuse of heterogeneous knowledge. The KRAFT project is a research project that investigates the feasibility of sharing and reuse of heterogeneous knowledge over computer networks. The project is conducted by the Universities of Aberdeen, Cardiff, and Liverpool, and British Telecom. In this paper we discuss one issue of the KRAFT project, namely the role of ontologies. We argue that ontologies (to be explained below) can play a role in solving some of the key problems in the sharing and reuse of knowledge.

The paper is structured as follows. In section 2 we elaborate on the problems that impede the sharing and reuse of knowledge. In section 3 we elaborate on ontologies, addressing ontology languages, types of ontologies and their merits. In section 4 we sketch how ontologies could be applied in the KRAFT project. In section 5 we provide tentative answers to the questions that have been raised in an earlier version of this article (Visser, 1996). Then, we address four new issues that arose from a KRAFT meeting in Gregynog (Wales) in section 6. Thereafter, in section 7, we briefly discuss some related work. Finally, we draw conclusions in section 8 and discuss future research in section 9.

2. Problems in Sharing and Reuse

Gruber (1991, p.601) mentions two reasons why sharing and reuse of knowledge is impeded: "today's knowledge systems are isolated monoliths characterized by *high internal coupling* (...) and *a lack of external coupling interfaces* (...)". Here, we briefly address both reasons.

---

[1] This paper is a revised version of the KRAFT position paper PP1 (Visser, 1996) as presented at the Gregynog meeting in May 1996. Major changes - besides some textual modifications - are: (a) the addition of a section in which we give some tentative answers to the questions raised in the previous version of this article (section 5), (b) the addition of a section on new issues that arose on the Gregynog meeting (section 6), (c) an extension of the related work section (section 7), and (d) a future research section (section 9).

The first reason why knowledge sharing and reuse are impeded is *internal couplings*. That is, it is difficult to separate the reusable component from the non-reusable components because they are usually intertwined. The MYCIN experiments, for instance, have illustrated that domain knowledge and diagnostic strategies can be easily be compiled when specifying knowledge-base rules (Buchanan and Shortliffe, 1985). This form of internal couplings is referred to as the *interaction problem*; knowledge specifications are dependent on the purpose they are created for (Bylander and Chandrasekaran, 1987). It is a problem because knowledge specifications that are created for a certain purpose usually cannot be reused for other purposes. Apparently, the specifications have tacit use-specific assumptions which prevent the application of the knowledge specification for other purposes (hence, they are called *blocking assumptions*). To enhance the reusability of domain specifications for multiple purposes, the number of blocking assumptions have to be reduced (see also: Visser, 1995). This can be achieved by clearly separating the specifications of domain knowledge and the specifications of the control regimes. We propose the use of CommonKADS, and in particular its expertise model, to achieve such a separation. In appendix A we briefly discuss the main distinctions made in the expertise model.

The second reason why knowledge sharing and reuse are impeded is th*e lack of external coupling interfaces*. This concerns problems in the exchange of knowledge between various agents. Often there is a mismatch in the knowledge structures of the agents, or, in the way their knowledge is expressed. To illustrate these problems we discuss eight examples in which knowledge exchange is impeded by a communication mismatch between two inter-operating systems[2] (By systems we here denote databases, knowledge bases and problem solvers linked to the network. In this paper, we assume - for reasons of simplicity - that the intermediate agents in the network architecture, such as mediators and facilitators, are not regarded as systems):

1. ***Mismatch in variable typing*** occurs when two systems use the same terms (predicate relation) but differ in the type of the variables required. Consider the following two typed expressions: length(X:Road, L:Kilometres), and length(X:Road, L:Miles).
2. ***Mismatch in term definitions*** occurs when two systems use the same term (predicate) for different concepts. An example is the term mitre. One system may define a mitre with the expression: mitre(X) ← head_dress_of_bishop(X), whereas a second system may define a mitre with the expression: mitre(X) ← straight_angle_joint_of_wood(X). We note, that in this example, the term mitre denotes two rather distinct concepts. This does not necessarily have to be the case. We assume a mismatch in term definition also occurs if two knowledge bases have different definitions of rather similar concepts. One system may define a mitre with the expression: mitre(X) ← head_dress_of_bishop(X), whereas a second system may define a mitre with the expression: mitre(X) ← hat_of_bishop(X) ∧ deeply_cleft(X). Note that in the latter example, we assume the defined concepts to be distinct because they are defined differently (for this mismatch category we thus ignore the possible intention to define the same concept).
3. ***Mismatch in concept definition*** occurs when two systems use different terms (predicates) to denote the same concept. An example is found when using synonyms. Consider the use of the expression: ship(X) ← seagoing(X) ∧ large(X) by one system and vessel(X) ← seagoing(X) ∧ large(X) by another system.
4. ***Mismatch in aggregation level*** occurs if two systems both contain information on the same concepts, but at different levels of abstraction. For instance, one knowledge base that contains knowledge about the male population, say: males(X) ← obliged_to_join_the_civil_service(X) and a database that distinguishes between young and old males, say: young_males(3.200.000), and old_males(12.000.000).

---

[2] We do not think this list of mismatch examples to be complete (other mismatch examples could be found), nor do we consider the categories to be disjoint (some communication mismatch problems could be classified under multiple headings). However, the examples given are generic for a class of mismatch problems and drawing up a list of benchmark problems could facilitate assessing the adequacy of proposed solutions.

5. ***Mismatch of structural perspective*** occurs when two systems contain information about the same set of concepts but structure their knowledge from a different perspective. As an example we consider two knowledge bases that both contain knowledge about animals. A mismatch of structural perspective occurs when one knowledge base structures its knowledge around categories mammals and birds, whereas the second structures its knowledge around categories carnivores and ungulates.
6. ***Mismatch in attribute assignment*** occurs when two systems both contain knowledge about a concept but differ in the way they assign their attributes to the concept. An example is the concept of a (red) car. One knowledge base could represent a red car as red_car(X), whereas another knowledge base could represent it as car(X) ∧ red(X). We note that the two conceptualisations not necessarily have a different structural perspective.
7. ***Mismatch in formalisms*** occurs when two systems use different formalisms to express their knowledge. This is a very broad category of communication mismatch problems. For instance, one system uses a production-rule system (*e.g., if* rain *then* wet_street) and another system uses a horn-clause knowledge system (*e.g.,* wet_street ← rain).
8. ***Mismatch in formalism expressiveness*** occurs when two systems use different formalisms to express their knowledge while the expressiveness of the languages differs. Take as an example a system using predicate logic (*e.g.,* ∀ X: flies(X) ← bird(X)) and another system that uses propositional logic (*e.g.,* flies ← bird) (note that this type of mismatch is an instance of the previous type of mismatch).

Some authors distinguish between *syntactic problems* and *semantic problems* (*e.g.,* Wiederhold, 1994). An instance of a *syntactic problem* is when two knowledge bases use (slightly) different predicates to denote the same concept say: (product_number P N) and product_number(P,N). In this example, a simple mapping between the two expressions could solve the communication problem. For this reason, syntactic problems are considered to be relatively easy to solve (Wiederhold, 1994). However, not all communication problems are pure syntactic. Moreover, the distinction between semantic and syntactic problems is not always clear. For instance, when one attempts to map two formalisms with a different syntax, the mismatch problem could turn out a semantic problem rather than a pure syntactic problem in case the target formalism is less expressive than the source formalism. We do not elaborate on this distinction here. More important is that the communication problems described above cannot be solved with a simple mapping between predicate relations. Generally speaking, semantic problems are more severe and require more sophisticated solutions. To solve semantic problems between systems, knowledge of their respective domain conceptualisations is required. To achieve this, we need to make these conceptualisations explicit. The next section deals with explicit conceptualisations of domains, viz. ontologies.

3. Ontologies

Building knowledge systems involves the creation of a model of a particular domain (e.g., electronic circuits, legislation, plants). Such a model is necessarily an abstraction of the domain under consideration. This is what makes a model useful, it abstracts from irrelevant details and thereby allow us to focus on the aspects of the domain we are interested in. Building a model of a domain involves deciding upon what entities in the domain are to be distinguished, and what relations will be recognised between these entities. Moreover, it involves deciding upon what types of entities, and what types of relation exist. Often, the latter kind of decisions are straightforward and not always explicitly documented. For instance, in building a model of the blocks world we use predicates like *block(A)* and *on(A,B)* thereby implicitly assuming the domain to consist of blocks, tables, and hands, and that blocks, tables and hands have spatial relations. Making these assumptions requires the domain to be carved up in concepts. Alternatively stated, they reflect a *conceptualisation* of the domain under consideration. The conceptualisation tells us the types of entities and how they relate. In the blocks world the conceptualisation tells us that there are blocks, tables, hands and that blocks and tables have spatial relations. It does not, however, tell us what blocks, tables and hands there are, nor how they are spatially related.

It is important to note that there may be many 'correct' conceptualisations of one particular domain. Creating an ontology is not an unequivocally defined process, the same blocks world can be conceptualised in several ways. Some entities in the world may not need to be conceptualised at all (for instance, it is not always necessary to conceptualise the hand in the blocks world), other entities could have been specified more abstractly (e.g., recognising only entities and relations) or just less abstractly (e.g., recognising cubes, and cylinders). We may state that making a conceptualisation is accompanied by a considerable amount of freedom.

Ontologies

To be able to compare conceptualisations of a domain, it is necessary to make them explicit. Thus, we need to have an explicit conceptualisation. This is what the word ontology is used for in AI research:

> an *ontology* is defined as an explicit conceptualisation of a domain (Gruber, 1992).

Typically, an ontology is a set of definitions of (hierarchically ordered) classes, objects, attributes, relations, and constraints (henceforth shortly: entities and relations). It's main function is to provide a vocabulary for the expression of domain knowledge. An ontology is a knowledge-level description (Newell, 1982) in that it is independent of any representational formalism (Van Heijst, 1995). Also, an ontology is considered to be a meta-level description because it is a specification of (the entities and relations in) a specification (*viz.* the model under construction); it abstracts from the particulars of the model under construction and specifies only its building blocks. We could say that an ontology describes the domain knowledge that remains invariant over various knowledge bases in a certain domain (*cf.* Guarino and Giaretta, 1995). For instance, an ontology could specify that in all knowledge bases of the blocks world an empty block is a block with no block on top of it. It should be stressed here that different interpretations of the word ontology are used by different authors. This is nicely illustrated by Guarino and Giaretta (1995), who discuss seven interpretations of the word ontology.

Languages to specify Ontologies

To be able to specify a conceptualisation explicitly we need to have some specification language. In principle, this can be any language, but there are languages tailored to expressing ontologies. The most commonly used ontology languages are CML (Wielinga et al., 1994; Schreiber et al., 1995) and ONTOLINGUA (Gruber, 1992). Briefly stated, these languages allow us to define entities, relations between entities, attributes of entities (a special kind of a relation between two entities), and various constraints on entities, relations and attributes (e.g., on the values of the attributes). Currently, a small number of example ontologies is available in both ontology languages.

Towards a Classification of Ontologies

An ontology describes a domain from a particular viewpoint. The assumptions about the domain that define such a viewpoint are referred to as *ontological commitments* (Gruber, 1992). Because different ontologies express different viewpoints they have different ontological commitments. The type of ontological commitments can be used to classify ontologies. If we can distinguish different types of ontological commitments it will enable us to distinguish between different types of ontologies, and hence, it will enable us to classify ontologies. As a starting point to define commitments we adopt the commonly made distinction in AI literature between task, methods, and domains (*cf.* Wielinga et al., 1994; see also, appendix A). Hence, we arrive at three different types of ontological commitments (a) task commitments, (b) method commitments, and (c) domain commitments.

(a) ***Task commitments***. An ontology has task commitments if it defines entities and relations that express a task-specific perspective on the domain knowledge (by a task we mean a specification of a goal together with some input and required output, see also Visser, 1995). Typical task commitments are found in, for instance, an ontology for a diagnosis task, which contains entities, such as observations, causes, and hypotheses.

(b) ***Method commitments***. An ontology has method commitments if it defines entities and relations that express a method-specific perspective on the domain knowledge (by a method we mean a specification of how a task can be performed, see also Visser, 1995). Typical method commitments are found in, for instance, an ontology for the propose-and-revise method (within a design task), which contains entities, such as proposed solution, constraints, and value-assessment.

(c) ***Domain commitments***. An ontology has domain commitments if it defines entities and relations that relate to a particular domain (by domain we here refer to the commonly distinguished fragments of the real world which is being modelled, such as medical, legal, mathematical, financial, or social domains). Typical domain commitments are found in, for instance, an ontology for the legal domain, which contains entities, such as norms and acts.

Ontologies can be classified according to the type of commitments they make. Using the three types of commitments mentioned above, we could say that a *task ontology* is an ontology that makes (substantial) commitments towards a certain (group of) task(s), a *method ontology* is an ontology that makes (substantial) commitments towards a particular (group of) method(s), and a *domain ontology* is an ontology that makes (substantial) commitments towards a particular (group of) domain(s). We note that the amount of commitments (of a certain type) made in an ontology may vary, it can be a small amount or a large amount of commitments. Hence, we can define a *generic domain ontology*, an ontology that makes commitments towards a particular domain but is generic in that it can be refined in many subdomains. As an example, we mention a generic legal ontology that distinguishes between norms, acts and concept definitions but does not make commitments towards any legal subdomain (*e.g.,* Visser, 1995). Also, ontologies can have a combination of the commitments mentioned. It could be argued that the distinction of the hand in the blocks world is a commitment towards a certain method (namely: allowing methods that use 'intermediate' states).

The Merits of Ontologies

Recently the result of gathering vocabularies and structuring domains has been recognised as a valuable effort in its own right, deserving of attention (Wiederhold, 1994; p.7). In general, we can say that ontologies may contribute to the following five areas.

(1) ***Domain-theory development.*** Because an ontology explicitly states the building blocks of particular domains, it can be used for the analysis, comparison, and development of domain theories. Examples of this kind of ontology use can be found in Sim and Rennels (1995), and Visser and Bench-Capon (1996).

(2) ***Knowledge acquisition.*** Ontologies describe and structure the entities and relations that need to be acquired for the domain under consideration. Examples of this kind of ontology use are CUE (Van Heijst and Schreiber, 1994; Van Heijst, 1995), and MOBAL (Morik e*t al.*, 1993).

(3) ***Knowledge-system design.*** Ontologies are reusable constructs in the design of knowledge systems because they can be used to represent the invariant assumptions underlying different knowledge bases in the same domain. As such, they can be considered as initial building blocks of the knowledge base under construction. An example of this kind of ontology use can be found in the GAMES methodology (Van Heijst, 1995).

(4) ***System documentation***. Ontologies provide a meta-level view (vocabulary, structure) on their application domain which facilitates adequate system documentation for end-users. An example of this kind of ontology use is found in the Cyc project (Lenat and Guha, 1990).

(5) ***Knowledge exchange.*** Ontologies can be used to define assumptions that enable knowledge exchange between different systems. Examples of this kind of ontology use can be found in SHADE (Kuokka *et al.* 1993), and Wiederhold (1994).

In the next section we elaborate on the use of ontologies in relation to the sharing and reuse problems recognised in the KRAFT project.

4. Possible Roles of Ontologies in the KRAFT Project

In section 3 we discussed ontologies and their merits independent of the sharing and reuse problems discussed in section 2. In this section we attempt to bring these two issues together. In particular, we discuss the possible benefits from using ontologies to deal with the problems mentioned in section 2.

We envisage that the contribution of ontologies to the KRAFT project, that is, the contribution of ontologies to deal with the problems discussed in section 2, mainly concern the areas of knowledge acquisition (2), and knowledge exchange (5). Below, we elaborate on these issues.

Knowledge Acquisition

Ontologies may play a role in knowledge acquisition in that systems that want to join the KRAFT network have to comply with the standards defined in the (domain) ontology. Thus, the ontology prescribes the types of knowledge that needs to be acquired in the domain and the way it should be named and structured in order to be able to join the network. Therewith, the ontology provides a useful structure that may help in avoiding internal couplings. In section 2 we stressed that blocking assumptions in knowledge specifications may prevent the specification from being reused for other purposes. The specification of knowledge involves making selections, simplifications and knowledge compilations and also, the knowledge base is structured according to its intended use (Visser, 1995). Ontologies can be shown to prevent some of these undesired blocking assumptions (Van Heijst, 1995; Visser, 1995). In particular, ontologies can be shown to impose a structure on the knowledge base and may help to prevent certain selections, simplifications, and knowledge compilations.

Knowledge Exchange

As stated in section 3, ontologies can be used to define assumptions on a particular domain to enable information exchange between systems. There are three possible strategies to achieve the information exchange:

(S1) In the first strategy one ontology is created which defines a standard language (vocabulary) for all systems involved. All systems have to use this vocabulary when communicating knowledge with other systems.

(S2) In the second strategy ontologies are defined for each of the systems and mappings are defined between the individual ontologies of the systems involved.

(S3) The third, and probably most feasible, strategy is to define one common ontology but to allow each system to have its own ontology. Knowledge is exchanged by mapping individual ontologies onto the common ontology and vice versa.

Strategy two and three require the mapping of ontologies. This can be done by defining so-called 'ontology mapping rules' which relate knowledge expressions in the two ontologies. Examples of ontology mapping rules (dealing with the first and third type of communication mismatch example as described in section 2) are (in the first example KM-conversion is a declarative function that converts between miles and kilometres):

ONT-1: length(X,K) « » ONT-2: length(X,M) $\wedge$ KM-conversion(K,M)
ONT-1: ship(X) « » ONT-2: vessel(X)

One more issue needs to be addressed if we are to use ontologies for the exchange of knowledge between systems. We have not yet addressed what *type of knowledge* the systems are willing to exchange. An ontology conceptualises a particular domain, but we have not stated what possible domains could be conceptualised. In fact, there are different types of ontology domains, each with a different nature. We envisage four categories of domains for ontologies in the KRAFT project:

(OD1) *Declarative application knowledge.* Such an ontology captures the shared assumptions on the (declarative part of the) application domain. By application domain we here mean the domain that is modeled and reasoned about. Examples are bacteriological diseases, ceramic materials, social security regulations, or the cockpit of an airplane. We note that such ontologies would contain mainly domain commitments (see section 3).

(OD2) *Knowledge-manipulation operations.* Such an ontology captures the shared assumptions on the available knowledge manipulation operations that can be performed on the declarative application knowledge (such as: tasks, methods, database queries). This facilitates sharing the knowledge-manipulation operations between systems. We note that such ontologies would contain mainly task- and method commitments (see section 3).

(OD3) *Representation formalisms.* Such an ontology captures the shared assumptions concerning the representation formalisms of systems certain in the network. The ontology then can be used to define a standard representational formalism or to map the different representational formalisms onto each other. Depending on the type of representational formalism, the ontology will have domain, method, and / or task commitments.

(OD4) *System characteristics.* Such an ontology captures shared assumptions concerning system capabilities and structure. Thus, the ontology is a meta-level description of the characteristics of a system. For instance, the ontology could be used to express that knowledge systems in the network are built of task, inference and domain knowledge. That domain knowledge consists of a set of domain models, etc. We note that the ontology in this case would contain mainly task- and method commitments.

The discussion of the possible roles of ontologies as presented here should be seen as an inventory of alternatives rather than as a proposal to use ontologies in a particular way. We have discussed different types of ontologies, as well as different categories of domains in which ontologies can be applied. It should be noted that, within the KRAFT project, different ontologies can be created, each serving a different purpose. Which ontology is made (or: ontologies are made) depends on considerations that fall without the scope of this article (see below).

5. Tentative answers to previously raised questions

In this section we provide some tentative answers to questions that arise from the previous sections and that were mentioned in an earlier version of this article (Visser, 1996). The questions have been addressed (in more or less detail) at the Gregynog meeting in Wales (the answers given below do not necessarily reflect the opinion of the other researchers of the KRAFT project).

Question 1: *Do we want to exchange knowledge-manipulation operations (such as task- and method specifications) between systems?* This question has not been given an explicit answer during the Gregynog meeting. However, the current feeling is not to aim for the exchange of knowledge-manipulation operators. That is, it may be necessary to exchange problem-solving capabilities (*viz.* what problems can be solved), but it is not felt necessary to exchange the way the problems are solved (*viz.* how problems can be solved). The emphasis is put on the exchange of declarative domain knowledge rather than tasks and methods (*cf.* OD1, section 4).

Question 2: *Do we envisage the shared domain knowledge to be dynamic or static?* This question has been given a clear answer by formulating that dealing with the dynamics of the domain knowledge is a prerequisite for the KRAFT network. More debatable is the question of whether the domain ontology should be dynamic or static (see also section 6).

Question 3: *Do we aim to integrate a set of existing systems, or do we aim to define the conditions under which future systems can share and reuse their knowledge?* This question has not been addressed in full. That is, the desirability of a methodology to integrate existing systems is shared among all the researchers. The KRAFT network is meant to be an open architecture in that systems of different natures (*viz.* problem solvers, databases, knowledge bases) should be able to join (and leave) the network. As a consequence, we do not want to impose rigid standards on these systems, they should retain their 'individuality' as much as possible (*e.g.,* we do not want to impose one representation formalism onto each of the systems involved). However, it is not possible to integrate an arbitrary set of existing systems. Systems that are to join the KRAFT network will probably have to comply with several standardisation principles (such as the structure of their knowledge base). For these reasons, we cannot say whether we aim to integrate a set of existing systems, nor whether we just want to define the conditions under which future systems can share and reuse their knowledge. Probably, these alternatives have to be considered as two extreme positions along the same line.

Question 4: *Do we restrict the nature of the shared domain knowledge that is to be exchanged (e.g., time, space, normative modalities, probabilistic information)?* This question has not been given a clear answer. Again the desirability of not restricting the nature of the exchanged knowledge has been stressed in that for all the knowledge types mentioned in the question, researchers were found to stress the importance of supporting that type of knowledge. However, it is clear that not restricting the nature of the knowledge that is exchanged makes the project very ambitious, if not infeasible. Currently, it is thought to be necessary to restrict the nature of the knowledge. That is, in this phase of the project we do not want to facilitate the expression of modal logics, temporal logics, or probabilistic knowledge. We now confine ourselves to the constraint-logic expressions of the CoLan language (Bassiliades and Gray, 1994), and divert the expression of knowledge currently not explicitly supported by the CoLan language (such as modalities) to a later phase of the project (provided that it is found desirable). Therewith, we remark that we are aware that it may be difficult to scale-up the network to embed these types of knowledge.

Question 5: *What knowledge exchange strategy (see section 4) is to be applied?* What knowledge-exchange strategy (S1 through S3 in section 4) is to be applied needs to be determined for each (potential) ontology domain (OD1 through OD4 in section 4) individually. Not surprisingly most of the discussions in Gregynog have focused on using ontologies for the exchange of declarative application knowledge (OD1). It is felt that - in the long term - strategy S3 should be used for this ontology domain (however, we note that for the prototype strategy S1 has been chosen). Interestingly, most other projects that use ontologies for the integration of heterogeneous knowledge seem to have chosen for strategy S2 (*e.g.,* Tadeka *et al.*, 1995; Campbell *et al.*, 1995). Given the answer on question 1, it is not necessary to consider an exchange strategy for ontology domain OD2. With respect to ontology domain OD3 we opine that strategy S3 again is the most feasible solution (we remark that the *lingua franca* is assumed to be a very expressive language with a low computational tractability, and the resource languages are less expressive but more tractable). Ontology domain OD4 is probably best dealt with using strategy S1.

6. New issues arising from the Gregynog meeting

In this section we discuss four new issues that arose at the Gregynog meeting in Wales. All four issues concern the use of the ontology for ontology domain OD1 (here referred to as the domain ontology). In particular, we discuss: (a) two different scenarios for the domain ontology, (b) the location of the domain ontology, (c) the access to the domain ontology, and (c) a dynamic versus a static domain ontology.

(a) *the domain ontology; two scenarios*

The domain ontology defines a communication standard for the exchange of information between the agents in the network. We envisage two possible scenarios regarding the role of the domain ontology in this process. The first scenario is that the domain ontology is *solely used in the design of participating systems* (here a participating system is an agent that is to become part of the KRAFT network, such as knowledge bases, wrappers, and mediators). The ontology is then used by the knowledge engineers and system designers, and not by the wrappers, mediators, or facilitators themselves. Accordingly, the ontology does not necessarily have to be machine readable. After the construction of each participating system the domain ontology does not have a role any more (until a new system is to become part of the network). We note that this scenario does not imply that the ontology is not implicitly stored on different locations in the network, however, it won't be stored as 'an ontology' (see also the discussion on issue *b* below). The second scenario is that the ontology is used *both in the design of participating systems as well as during the actual operation of the network*. The ontology is used in the same way as it is used in the previous scenario but it is also used by some systems in the network to perform their tasks. Examples of such a use could be that a mediator supplies information on the subclasses of a given class, or, that a mediator checks whether an expression satisfies the constraints of an ontology (here, we do not address which systems would use the ontology, *how* they would use it, nor for *what purpose* they would use it in any more detail). This scenario requires the ontology to be stored in a system that is part of the KRAFT architecture (as a separate knowledge base - called an auxiliary, see Embury *et al.,* 1996), and it also requires the ontology to be machine readable.

From the discussions in Gregynog we draw the provisional conclusion that there is a growing belief that the second scenario is to be preferred over the first scenario. However, it is not yet clear for what purposes the ontology would be used in the operation of the network (nor how it should be used, and by what systems in the network). Without these questions having been answered it may yet seem pointless to make the ontology as such a part of the network (see also below).

(b) *location of the domain ontology*

Related to the previous issue is the question of where the domain ontology is stored in the network? Below, we illustrate that the answer to this question requires a brief discussion of the phrase 'stored in the network'. In particular, we argue that a distinction should be made between an *implicitly* and *explicitly* represented domain ontology.

Consider a KRAFT architecture in which each resource (*viz.* knowledge base, database, or problem solver) has its own domain ontology, and in which there exists a 'common domain ontology' that is used as an intermediate ontology (see: section 4, knowledge-exchange strategy S3 for ontology domain OD1). In the architecture, system communication is done by translating resource-ontology expressions into common-domain-ontology expressions (and *vice versa*). Consider furthermore that wrappers are responsible for these ontology translations, and that the wrappers use ontology mapping rules (see section 4) to achieve the translation. All mapping rules are based on two ontologies (*viz.* a recourse ontology and the intermediate ontology), and obviously they represent knowledge about these two ontologies. One could argue that the mapping rules 'represent' the respective ontologies, or similarly, that the ontologies are stored in the mapping rules. Accordingly, we could claim that the ontology is *stored* in the wrappers.

We disagree with this claim for the reason that the domain ontology as such is not stored *explicitly* in the mapping rules. Probably, the domain ontology can (partially) be derived from the mapping rules, but it is only *implicitly* present in these rules. We consider an explicitly stored ontology to be a knowledge base that contains a set of definitions of (typically hierarchically ordered) classes, objects, and relations. Hence, we consider the wrapper not to *store* the ontology.

The latter view allows us to re-address the previously raised question as to where the ontology is stored. If the ontology is used during the actual operation of the network (see issue a), then the ontology will have to be explicitly stored. We envisage two options, either it is stored in a separate knowledge base (as a dedicated resource), or, it is stored as part of a (dedicated) mediator program. The former options has the advantage that the ontology is readily available to all systems in the network, but has the disadvantage that the ontology knowledge base (which is a meta-level knowledge base) is embodied as an ordinary knowledge base despite the fact that the nature of its knowledge is different from the other knowledge bases. The latter option does not have this disadvantage but one may question whether the domain ontology, if it is part of a mediator program, is easily shared between the systems in the network (*cf.* internal couplings, see section 2). If the ontology is not used during the actual operation of the network, then it does not have to be stored, irrespective of the possibility that the domain ontology is implicitly embedded in several systems in the network.

(c) *access to the domain ontology*
The question as to what systems in the network should have access to the domain ontology cannot be answered before determining the purposes of the domain ontology in the network (see again issue a). We confine ourselves by remarking that if different agents need to have access during the operation of the network to the same ontology, it needs to be stored explicitly.

(d) *dynamics versus static domain ontology*
A last issue that arose from the Gregynog meeting concerns the question as to whether the domain ontology is to be static or dynamic. Maintenance has become a pivotal issue in the design of most software programs; problems regarding maintenance are increasingly addressed *during* software design rather than *after* software design. If we create a common domain ontology for the KRAFT network we obviously would like the ontology to be dynamic so as to be flexible with respect to changes in the domain knowledge.

Although this is certainly a desirable feature, there are severe problems concerned with making the common domain ontology dynamic. For instance, suppose we have defined a common domain ontology. If we change the ontology, we possibly have to modify all wrappers in the network. Possibly, the change in the common domain ontology is such that a translation between the two ontologies is not feasible any more[3]. In summary, we believe that, although making the ontology dynamic is certainly a desirable feature, only minor changes to the ontology must be allowed. We note, that because the domain ontology is a meta-level description with respect to the knowledge bases in the network, the ontology is less likely to change than the knowledge bases themselves.

7. Related work

The application of ontologies for knowledge sharing and reuse is being investigated by several research groups. Below, we briefly discuss the work of six research projects in this field: (1) Cyc, (2) KACTUS, (3) ARPA

---

[3] Wiederhold (1995) proposes an 'economic model' to deal with update problems of mediators. The idea is to maintain multiple versions of mediators but to discourage the use of old versions. The new version can be introduced without altering the old version. A similar idea could be applied to (mediators storing) common ontologies, keeping multiple versions of common ontologies but discouraging the use of outdated versions.

Knowledge Sharing Effort, (4) SHADE, (5) the Knowledgeable Community project, and (6) the Ontological Mediation project.

Cyc (MCC and others). The Cyc project is an ambitious attempt to create an enormous common-sense knowledge base. Lenat and Guha (1990) claim that for solving a diversity of real-world problems a large amount of common-sense knowledge is needed. Future knowledge systems should have access to such a knowledge base, even for solving rather trivial problems. The size of the Cyc knowledge base requires it to rely on firm ontological assumptions. These assumptions concern time, space, events, grain size, tangible and intangible objects, belief, agents, and the like. Although ontological assumptions play an important role in the project, they have not been defined as an explicit ontology. The project started in 1984 and ended in 1995 (more information can be obtained from: http://www.mcc.com/projects/cyc/cyc.html).

KACTUS (University of Amsterdam and others). The KACTUS project aims at the development of methods and tools for the reuse of knowledge about technical systems (Wielinga *et al*, 1994). The basic premise of the project is that knowledge sharing and reuse can be made possible through the definition of ontologies. The KACTUS architecture comprises libraries of ontologies at different levels, facilities to select, retrieve, merge and map ontologies (more information can be obtained from: http://www.swi.psy.uva.nl/projects/Kactus/home.html).

ARPA Knowledge Sharing Effort (Stanford University). The ARPA Knowledge Sharing Effort (KSE) is a consortium to develop conventions facilitating sharing and reuse of knowledge bases and knowledge-based systems (Gruber, 1993). The project has several working groups working on different issues and domains (see also SHADE). The most important working groups concentrate on a common language for the interchange of knowledge between heterogeneous systems (KIF = Knowledge Interchange Format), a Knowledge Query and Manipulation Language (KQML), and an ontology library (ONTOLINGUA) (more information can be obtained from: http://www.cs.umbc.edu/kse/).

SHADE (Stanford University, Lockeed). SHADE is part of the KSE project. The SHADE (SHAred Dependency Engineering) project is concerned with knowledge sharing and reuse for concurrent engineering (Gruber, 1992; Kuokka *et al.*, 1993). SHADE includes three main thrusts: (a) a shared knowledge representation and domain ontology, (b) protocols supporting dynamic information exchange, and (c) facilitation services for content-directed routing and intelligent matching of information (more information can be obtained from: http://hitchhiker.space.lockheed.com/aic/shade).

Knowledgeable Community Project (Nara Institute of Technology, Nara, Japan). The Knowledgeable Community Project (KCP) aims to provide a framework for knowledge sharing and reuse based on a multi-agent architecture (Takeda *et al.*, 1995). The KCP architecture recognises (a) agents, (b) mediators, (c) facilitators, (d) translation agents, and (e) dedicated ontology servers (more information can be obtained from: http://flute.aist-nara.ac.jp/KC/).

Ontological Mediation Project (Computer Science, State University of New York, Buffalo). The research project is exploring various issues in ontologies, knowledge sharing, vocabulary acquisition, and distributed databases. Also, the project recognises facilitators, mediators, amongst which an ontological mediators (more information can be obtained from: http://www.cs.buffalo.edu/~aec/OM/).

8. Conclusions

Knowledge sharing and reuse is impeded by (a) internal couplings and (b) a lack of external couplings. In the KRAFT project ontologies may contribute to both types of problems drawing on their adequacy for knowledge acquisition and knowledge exchange.

From a *knowledge acquisition* perspective they may contribute to reducing internal couplings by proving a structure on an application domain, thus preventing undesired selections, simplifications, knowledge compilations and use-dependent structuring. From a *knowledge exchange* perspective they may provide external coupling interfaces by either defining over-arching standards, or, by defining individual ontologies that can be mapped from one to the other.

This article does not propose to use ontologies in one particular way for the KRAFT project. It rather has to be seen as an inventory of possible application domains for ontologies. What ontologies are built, and for what purposes, depends on issues that fall beyond the scope of this article.

With respect to the design issues of the KRAFT network, as addressed in the five questions of section 5, we conclude the following. Emphasis will be put on using ontologies for the exchange of declarative domain knowledge, rather than for the exchange of tasks and methods - although we want to be able to exchange system capabilities (*question 1*). The network architecture should allow for changes in the domain knowledge that is exchanged, not necessarily for changes in the domain ontology (*question 2*). The KRAFT research is not solely focused on integrating a set of existing systems, nor solely on finding the conditions under which future systems can share and reuse their knowledge. Rather it has to be considered as an analysis of the boundaries of these two options (*question 3*). With respect to the nature of knowledge exchanged, we initially focus on the knowledge that is expressible with the CoLan formalism. Knowledge types not supported by this formalism are dealt with later on in the project (*question 4*). For the declarative application knowledge and the representation formalisms we apply exchange strategy S3, for the system characteristics we apply exchange strategy S1. It is not felt necessary to exchange problem-solving operators (*question 5*).

Additional design issues follow from the discussions during the Gregynog meeting (see section 6). The domain ontology will be used both during the construction of the systems participating in the KRAFT network, as well as during the operation of the network. We make the domain ontology explicitly available in the network architecture, either as a separate knowledge base, or in a mediator. Also, we initially assume the domain ontology to be static (meanwhile investigating the conditions under which the domain ontology could change without affecting the integrity of the network).

9. Future research

Future research in Liverpool concerning ontologies in the KRAFT project will focus on the following research topics:

- The use of so-called ontological mediators, which are dedicated translation mediators that have access to knowledge bases of ontology mapping rules.
- The creation of domain ontology mapping rules.
- The feasibility of mapping ontologies that impose substantially different views on the same body of domain knowledge.
- The feasibility of automatically retrieving ontologies from knowledge bases.

Acknowledgements

KRAFT is a multi-site project (of the Universities of Aberdeen, Cardiff, Liverpool, and of British Telecom) that aims to investigate the feasibility of heterogeneous information bases. The project started in May 1996.

This is a revised version of the position paper (PP1), presented at the Gregynog meeting in Wales. Contributions to the initial paper were made by Martin Beer, Trevor Bench-Capon, Bernard Diaz, Dean Jones, Ken Lunn, and Mike Shave. Revisions were made after comments of the ontology working groups at Gregynog, consisting of Nadir Azarmi, Andrew Jones, Dean Jones, Ken Lunn, Alun Preece, Jianhua Shao, and Mike Shave.

References

Bassiliades, N., and P.M.D. Gray (1994). CoLan: A Functional Constraint Language and its Implementation, *Data and Knowledge Engineering*, 14, pp.203-249.

Breuker, J.A., and W. van de Velde (1994). CommonKADS Library for Expertise Modelling, Reusable Problem Solving Components, J.A. Breuker, and W. van de Velde (eds.), IOS Press, Amsterdam, the Netherlands, ISBN 90-51991649.

Buchanan, B.G., and E.H. Shortliffe (1985). Rule-Based Expert Systems; The MYCIN experiments of the Stanford Heuristic Programming Project, Addison-Wesley Publishing Company, Massachusets, United States.

Bylander, T., and B. Chandrasekaran (1987). Generic tasks for knowledge-based reasoning: the "right" level of abstraction for knowledge acquisition, International Journal of Man-Machine Studies, Vol. 26, pp.231-243.

Campbell, A.E., H. Chalupsky, and S.C. Shapiro (1995). Ontologic Mediation: An Analysis, unpublished article, http://www.cs.buffalo.edu/~aec/OM/om.ps., pp.20.

Embury, S.M., K.Y. Hui, A. Preece, M. Ashwell, T.W. Lawson (1996). *The KRAFT Architecture for Distributed Information Systems, KRAFT Working Paper KP1*, Version 1.0, July 1996.

Gruber, T.R. (1991). The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases, Principles of Knowledge Representation and Reasoning, KR'91, Proceedings of the Second International Conference, J. Allen, R. Fikes, and E. Sandewall (eds.), pp.601-602, Cambridge, Massachusetts, United States.

Gruber, T.R. (1992). *Ontolingua: A Mechanism to Support Portable Ontologies*, technical report, Knowledge Systems Laboratory, Stanford University, Stanford, United States.

Gruber, T.R. (1993). *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*, technical report KSL-93-4, Knowledge Systems Laboratory, Stanford University, Stanford, United States.

Guarino, N., and P. Giaretta (1995). Ontologies and Knowledge Bases, Towards a Terminological Clarification, *Towards Very Large Knowledge Bases*, N. J. I. Mars (*ed.*), IOS Press 1995.

Heijst, G. van W.M. Post, and A.Th. Schreiber (1994) Knowledge-based integration of representation formalisms, A. Cohn (*ed.*) *Proceedings of the 11th European Conference on Artificial Intelligence,* pp.319-323, Wiley, London, United Kingdom.

Heijst, G. Van, and G. Schreiber (1994). CUE: Ontology-based knowledge acquisition, L. Steels, A. Th. Schreiber, and W. Van de Velde (*eds.*), A Future for Knowledge Acquisition, *Proceedings of the 8th European Knowledge Acquisition Workshop EKAW'94,* Vol. 867 of Lecture Notes in Artificial Intelligence, pp.178-199, Springer-Verlag, Berlin/Heidelberg, Germany.

Heijst, G. Van (1995). *The Role of Ontologies in Knowledge Engineering*, Doctoral Thesis, University of Amsterdam, Amsterdam, The Netherlands.

Kuokka, D.R., J. McGuire, J.C. Weber, J.M. Tenenbaum, T.R. Gruber, and G.R. Olsen (1993). SHADE: Knowledge-Based Technology for the Re-Engineering Problem, Annual Report 1993.

Lenat, D.B. and R.V. Guha (1990). *Building Large Knowledge-Based Systems; Representation and Reasoning in the Cyc Project*, Addison-Wesley, Reading, Massachusetts, United States.

Morik, K., S. Wrobel, J-U. Kietz, and W. Emde (1993). *Knowledge Acquisition and Machine Learning; Theory, Methods and Applications,* Knowledge-Based Systems, Academic Press Limited, London, United Kingdom.

Newell, A. (1982). The Knowledge Level, *Artificial Intelligence,* Vol. 18, pp.87-127.

Schreiber, G., B. Wielinga, and W. Jansweijer (1995). The KACTUS View on the 'O' Word, *IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995. Also in: J. C. Bioch and Y.-H. Tan (eds.). *Proceedings 7th Dutch National Conference on Artificial Intelligence NAIC'95*, pp. 159-168, EURIDIS, Erasmus University Rotterdam, The Netherlands.

Sim, I., and G. Rennels (1995). Developing A Clinical Trial Ontology: Comments on Domain Modeling and Ontologcal Reuse, Knowledge Systems Laboratory Medical Computer Science, KSL-95-60, June 1995.

Takeda, H., K. Iino, and T. Nishida (1995). Ontology-supported Agent Communication, *Working notes of 1995 AAAI Spring Symposium on Information Gathering in distributed environments*. pp.157-165.

Visser, P.R.S. (1995). Knowledge Specification for Multiple Legal Tasks; A Case Study of the Interaction Problem in the Legal Domain, Computer/Law Series, No 17, Kluwer Law International, The Hague, The Netherlands.

Visser, P.R.S., T.J.M. Bench-Capon, and H.J. van den Herik (1996). FRAMER: Assessment and Planning using a Single Domain Specification, *First European Conference on Law, Computers, and Artificial Intelligence,* Exeter, United Kingdom, pp.198-204

Visser, P.R.S., and T.J.M. Bench-Capon (1996). A Comparison of Legal Ontologies, University of Liverpool (*in preparation*).

Visser, P.R.S. (1996). *Ontologies in the KRAFT Project; KRAFT Position Paper PP1*, May 1996, pp.15, Gregynog, Wales, UK.

Wiederhold, G. (1994). Interoperation, Mediation, and Ontologies, *Proceedings International Symposium on Fifth Generation Computer Systems* (FGCS94), Workshop on Heterogeneous Cooperative Knowledge-Bases, Vol. W3, pp.33-48, ICOT, Tokyo, Japan.

Wiederhold, G. (1995). Modeling and System Maintenance, M.P. Papazoglou (*ed.*), *OOER'95: Object-Oriented and Entity Relationship Modelling, Springer Lecture Notes in Computer Science*, Vol. 1021, pp.1-20, *Proceedings, ER-OO conference*, Brisbane Australia, December 1995.

Wielinga, B., G. Schreiber, W. Jansweijer, A. Anjewierden, and F. van Harmelen (1994). *Framework for Expressing Ontologies*, DO1b.1-1-Framework-1.1-UvA-BW+GS+WJ+AA, University of Amsterdam, November 1994.

Appendix A: The CommonKADS Expertise Model

A.1 Introduction

One of the central attractions of knowledge systems when they first appeared in the early eighties was the potential they seemed to offer for reuse and maintenance. The idea of separating a declarative representation of the knowledge pertinent to a problem from the means of manipulating that knowledge should mean that the declarative representation can be used for other tasks which require that knowledge, and can be manipulated in different ways. In practice these hopes have never been realised. The ideal of a separate declarative representation of pertinent knowledge is rarely attained. We find:

- The separation is incomplete: typically the domain knowledge is intertwined with control knowledge, task knowledge and system related code (e.g. to perform input / output).
- The way the knowledge is represented is often dictated by the way the knowledge will be manipulated: concerns such as the problem solving strategies, conflict resolution and the like compromise the independence of the representation.

Recognition of this, coupled with a desire to realise the potential for reuse, has given rise to a wealth of research on how to engineer knowledge based systems so as to minimise these difficulties and so promote reuse. Such methodologies promote a clean separation of the various components for as long as possible, often making use of intermediate representations, and attempt to explicate implicit assumptions. This typically involves the construction of separate models for domain and task/control aspects. The best known of these is CommonKADS (Breuker and Van de Velde, 1994), and we should probably take their distinctions as our starting point.

A.2 The Expertise Model

A principal attraction of the CommonKADS expertise model is that it enforces a clear separation between control knowledge and domain knowledge, requiring independent descriptions of each. This is vital for our purposes since the compilation of domain and control knowledge is a major drawback of symbol level models with respect to reusability. The expertise model consists of three knowledge categories:

1. Domain Knowledge
2. Task Knowledge
3. Inference Knowledge.

Domain knowledge comprises three components (Wielinga et al 1992); (1) a domain ontology; (2) a case model; and (3) the domain models. The domain ontology is an explicit conceptualisation of the domain, similar to the work on Ontolingua of Gruber (1993). It defines the vocabulary, typology and taxonomy with which the case and domain models can be specified. Within CommonKADS there is a conceptual modelling language, CML, which provides three constructs: objects, (atomic and constructed), relations, and value sets. The case model is a description of the dynamic data related to a case, and includes the initial case data, available before the reasoning commences, intermediate data, derived during the reasoning process, and the case solution, which is the ultimate result of the reasoning process. All of these are specified by reference to the domain ontology. The domain models are coherent sets of expressions, again specified using the domain ontology which describes the domain from a particular perspective, such as its causal relations, or its components.

Task knowledge, which together with inference knowledge makes up the control knowledge of the domain, specifies how the goals of the case model can be achieved. Tasks may be *primitive tasks*, or *composite tasks*, built up from primitive tasks. Primitive tasks are linked onto inferences. Each task has a *task definition* which