Resolving Ontological Heterogeneity in the KRAFT Project

P.R.S. Visser, D.M. Jones, M.D. Beer, T.J.M. Bench-Capon, B.M. Diaz and M.J.R. Shave

CORAL - Conceptualisation and Ontology Research at Liverpool
Department of Computer Science, University of Liverpool
PO Box 147, Liverpool, L69 7ZF
United Kingdom
dean@csc.liv.ac.uk

Abstract. KRAFT is an agent architecture for the integration of heterogeneous information systems. The focus in KRAFT is on the integration of knowledge in the form of constraints. In this article we describe the architecture from an ontological perspective. We start by introducing the agent architecture and illustrate its application in the telecommunication-network design. We then describe how we assess the ontological heterogeneity in the domain, which problems the integration of constraint knowledge pose, and how we construct a shared ontology. Also, we describe the mapping, functions that are used to translate information between the shared and the local ontologies. Finally, we look at the direction our research is taking hereafter.

1 Introduction

KRAFT is a research project on the integration of heterogeneous information using an agent architecture [1]. The project differs from many other integration projects, such as OBSERVER [2], SIMS [3], Carnot [4], and COIN [5], in that knowledge (in the form of constraints) is integrated rather than just data or enriched data. The project is a collaboration between the Universities of Aberdeen, Cardiff and Liverpool in conjunction with BT and began in May 1996. At present the KRAFT architecture has been evaluated in the area of student-admission policies and the in the design of a router configuration in the telecommunications domain.

Information sources in the KRAFT architecture are heterogeneous with respect to their ontologies, or, domain conceptualisations, and before information can be integrated this heterogeneity has to be reconciled (here referred to as *ontological mediation*). In this article we report on the application of ontologies and ontological mediation as it is done in the KRAFT architecture. Also, we will briefly address our ideas on future generation of KRAFT architectures. It is intended as an overview article of the techniques applied. More details on each of the techniques can be found in other papers.

Section 2 discusses the KRAFT architecture and its application in the router-configuration domain. Section 3 contains an assessment of the communication needs and the heterogeneity between the various ontologies in terms of their ontology

T. Bench-Capon, G. Soda, and A M. Tjoa (Eds.): DEXA'99, LNCS 1677, pp. 668-677, 1999.

mismatches. This assessment is then used to develop a shared ontology on the basis of the local ontologies and WordNet. This is described in section 4. Section 5 addresses the issues involved in the mapping of expressions between local ontologies and the shared ontology. Finally, section 6 contains the conclusions.

2 The KRAFT architecture in the Router-Configuration domain

In essence, the KRAFT architecture allows the combination of information from a set of legacy systems, here referred to as resources. Examples of resources are databases, knowledge systems, constraint solvers, and web pages. The user interacts with the middleware architecture via the a user agent which allows him to formulate his queries and which also presents the results obtained. The agent-middleware architecture has three types of agents:

Wrapper (*W*) Links the external resources onto the KRAFT middleware. To do so it resolves language, protocol and ontology heterogeneity as these have to be translated into the KRAFT application internal 'standards'.

Facilitator (*F*) White and Yellow page facility. Consulted by the other agents to 'recommend' services that match their required functionality.

Mediator (*M*) KRAFT-internal problem solvers. Mediators analyse and decompose information request¸ and arrange for the required information to be gathered from the resources. Thereafter, information is combined if necessary and passed back to the agent that consulted the mediator.

Communication between the agents is done using our Constraint Command and Query Language (CCQL, based on KQML [6]). A typical session starts with the mediators and resources making themselves known to the facilitator (performative: *register*). After registration the mediator and resources advertise their capabilities (performative: *advertise*), also to the facilitator. The user agent, aiming to have a query answered, will contact the facilitator via his wrapper with the request to recommend an agent who can deal with his query (performative: *recommend-one* or *recommend-all*). The facilitator then consults his internal database of advertised capabilities and replies by forwarding (performative: *forward*) the appropriate advertisement(s) to the user agent. This advertisement allows the wrapper (or user agent) to communicate directly with the agent that made this advertisement. The latter agent will then be sent the query (performatives: *ask-one*, *ask-all*). By contacting the facilitator every time an agent needs the service of another agent the network can adopt to situations in which resources are (temporarily) out of service, or, select the most appropriate agent to deal with the query.

As mentioned earlier, KRAFT differs from other DB integration projects in that not only data is integrated but also constraints. To express constraints we have defined a Constraint Interchange Format (CIF), based on the constraint language CoLan [7].

The router-configuration prototype supports a network designer in his task of selecting a router that can be used to connect a customer site with a so-called point-of-presence (POP). The latter is the WAN entry point that the customer site will be connected to. An example of a typical user query is 'configure a network connection

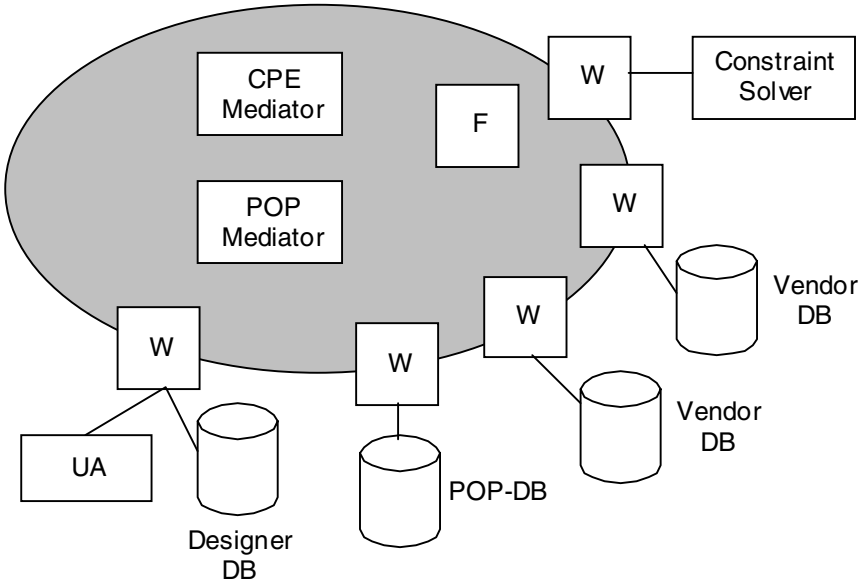

Fig. 1. The KRAFT Architecture in the Router-Configuration Prototype

from a given site to a Frame Relay network for under 2000 pounds'. The KRAFT architecture will decompose the query into sub queries such as "what is the best POP to connect the site to' and 'which manufacturers have router that supports Frame Relay and cost less than 2000 pounds'. The prototype has one user agent (UA) for the network designer and four resources (two databases of router vendors, one database for the point-of-presence information and one constraint solver). The middleware architecture has wrappers (W) for the user agent and all resources, one facilitator (F), a POP mediator (for selecting a POP), and a CPE mediator (for selecting the router). This is depicted in Fig. 1. Within the grey area there is one agent communication language (CCQL), one knowledge interchange format (CIF) and one shared ontology (more on the shared ontology in the following sections). Since in principle all agents can communicate with each other no inter-agent communication links are shown.

The constraints in our prototype domain express metadata about the kind of routers supplied by the vendors and are used to reduce the problem space of the network designer in his configuration task.. As an example, suppose a designer needs to configure a router that supports either the FrameRelay or ISDN protocol. Suppose that a vendor resource has the following constraint:

```
constrain each r in router

such that each p in protocol(r)

has name(p) in {"AppleTalk", "X.25", "ip", "FrameRelay",

"Switch56", "SMDS", "PPP", "SLIP"}
```

Retrieving this constraint from the vendor database obviously rules out all routers from this vendor since it does not supply routers that support the ISDN protocol. The CPE-mediator, on receiving this constraint, can eliminate this vendor's resource from this problem. Suppose instead this vendor had the following constraint:

```
constrain each r in router

such that some p in protocol(r)

to have name(p) = "ISDN"
```

This constraint effectively states that this vendor only sells routers that support ISDN. If this vendor also has the following constraint:

```
constrain each r in router

such that some p in protocol(r)

to have name(p) = "FrameRelay"
```

then all routers that are sold by this vendor support both FrameRelay or ISDN. Hence, the initial query can be relaxed as all routers from this vendor satisfy the designers constraints. This example illustrates how the fusion of a set of constraints can lead to the relaxation of the initial query or to the narrowing of the solution set. In both cases, however, the aim is to fuse a set of retrieved constraints in order to allow a more efficient way of solving the configuration problem.

3 Ontology Heterogeneity

The resources in the router-configuration prototype are heterogeneous with respect to their ontologies only. In practice this means that the databases have been developed independently of each other but using the same hardware platform, DBMS and interaction protocol. In this setting, the role of the wrappers is to perform ontology translations and to convert between the DB communication style and the agent communication style as used by the middleware.

In KRAFT we adopt a semi-structured approach to analyse the heterogeneity between two or more ontologies. This comparison will later be used for the design of the shared ontology (described in the next section). In the comparison we assume all ontologies to be defined as a set of (hierarchically related) entities with attributes and axioms. The idea behind the analysis is to identify the correspondence between the entities defined in the different ontologies. The existence of such a certain form of correspondence can be assumed simply since the need to integrate the resources means that there must be some correspondence in knowledge about the kind of information that is exchanged.

The first step in the comparison is finding semantically more or less similar entities across the ontologies. For all entities that have an approximate correspondence we then compare the attributes that are defined on these entities. We then refine the entity and attribute correspondences by classifying them in the ontology mismatch framework as described in [8]. This step allows us to identify mismatches, get an

indication for the hardness of resolving these mismatches and finally gives us a stepping stone for defining mapping functions (see section 5). It thus allows us to focus in an early stage on the integration difficulties that can be expected. The basic idea of the refinement is to classify the type of semantic correspondence, or rather the type of semantic mismatch. The mismatch framework distinguishes three components of a formal definition: the concept to be defined (C), the term to refer to it (T), and the *definiens* or body of definition (D). Two definitions can in principle fail to match in all combinations of these three components, which implies that the semantic correspondence between two entity definitions falls in one of seven different categories (if non of three matches then there is no correspondence). How the entity and attribute comparison is used in the construction of a shared ontology is described in the next section.

4 The Construction of the Shared Ontology[1]

In the KRAFT architecture the resource ontologies are not mapped directly onto each other. Instead, KRAFT uses shared ontologies which serve as the ontology counterpart of a lingua franca. Resource ontologies are mapped onto the shared ontology thus avoiding the potentially large number of individual resource mappings that may have to be defined. Although not used in the router-configuration domain the KRAFT architecture allows for multiple shared ontologies, organised in clusters. The idea behind ontology clusters is that resources do not have to commit to one over-arching standardised ontology but they can from groups (clusters) of resources that all want to commit to some standard.

In the router configuration domain there is one shared ontology. The local schemas in the resources are based on local ontologies. Ideally, the schemas are derived from their local ontology but this is - in the context of legacy system integration - usually not achievable. Rather, the ontology will be created after the schema, with the aim of clarifying the semantics of the entities (objects) used. Here, we will not address the relation between schemas and ontologies in any more detail. This issue is addressed in a separate - forthcoming - paper. The mapping functions bridge the gap between the concepts defined in the local ontologies and the concepts defined in the shared ontology (see section 5). This situation is depicted in Fig. 2 (the open arrows denote relation between schema and local ontology, the solid arrows denote mapping functions). There is not necessarily a one-to-one mapping between concepts in the local ontology and concepts in the shared ontology. Which concepts are mapped onto which shared concepts depends on whether a set of preconditions is satisfied. More on mapping functions in the next section. Before we can map the local ontologies onto the shared ontology we construct the shared ontology so as to support the definition of mapping functions.

[1] More details on this can be found in [9].

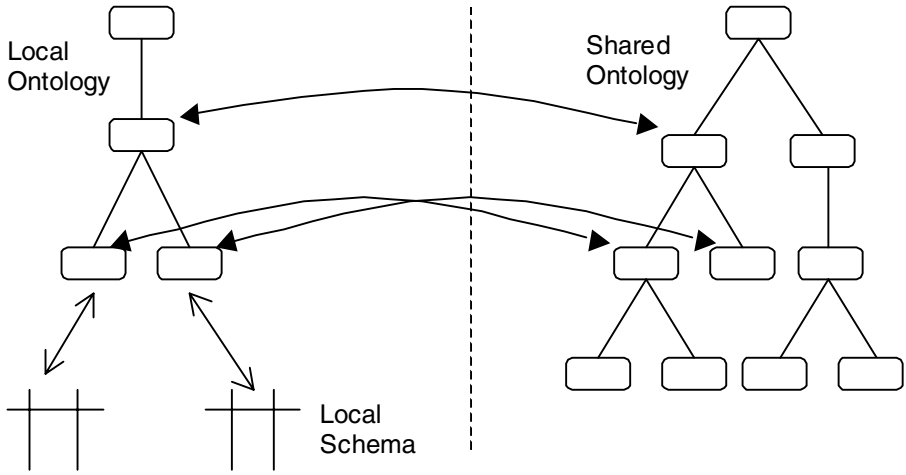

Fig. 2. Mappings between Local Schema and Local and Shared Ontologies

Besides supporting the mapping functions there are other design criteria for the shared ontology. How the shared ontology will be designed also depends on the communication needs. On the one hand, one would like the shared ontology to have the full expressiveness of the 'union' of the resource ontologies. The conclusion is of course that in the design of the shared ontology it is desirable to know what kind of queries and constraints have to be communicated. Also, the intensity of the communication plays a role in the design. Ignoring the optimal degree of expressiveness of the shared ontology we have chosen to make the shared ontology as expressive as the 'union' of the resource ontologies, thus anticipating future information requests.

The approach taken is the result of an ongoing research effort into conceptualisation and ontology design [10]. In short, we create the ontology by linking the required concepts onto the concept structure of an existing top-level ontology. The required concepts are determined by a text-based analysis of the router domain together with the results of the analysis that was described in the previous section. The ontology is thus more a *differential theory* derived from a corpus of natural language texts rather than a theory of concepts that are defined as extensions of a set of individuals in the domain [11]. The top-level ontology we use is WordNet [12]. In addition to this, we use text-analysis techniques to conceptualise domains [13] and techniques for the linking of our ontology into the top-level ontology [14]. The approach, which we here describe in condensed form, consists of three phases: (a) the domain analysis phase, (b) the ontology definition phase, and (c) the resource tuning phase.

A. Domain Analysis Phase

1. Determine a corpus of text material, such as brochures and technical manuals, that describe the domain. The advantage of using these documents as basis material is that we are sure to adopt the vocabulary that is used in the domain.

2. Determine the noun (phrases) used in the domain. This requires several processing steps, such as filtering out verbs, analysing the composite terms, and relating synonym¸ (here, we will refer to the remaining terms as *seed terms* after [14]. Tools can be used to assist in these steps (see, for instance [11]). We note that this way of filtering assumes the domain to be best described from a static perspective. Processes, actions, events and the like are - as yet - not considered.
3. Locate these terms in their correct interpretation in a top-level ontology (here: WordNet). This yields two groups: *supported seed terms* (those terms that do occur in their correct interpretation in the top-level ontology), and *unsupported seed terms* (those terms for which the top-level ontology does not have the correct interpretation).
4. Decide for the unsupported seed terms which ones will be included in the ontology. In particular, this may imply filtering out less relevant proper nouns such as names of brands etc. The decision to include or exclude a proper noun should be based on their relevance to the domain. For example, protocol names such as Token Ring and Ethernet are likely to be important in the networking domain and should be included.

All terms that are selected in step 4 of the analysis phase will be modelled in the domain ontology during the ontology definition phase. To do this, we need a specification language. Most ontology-specification languages support entities, attributes, and values. ONTOLINGUA, LOOM and Classic are examples of such languages.

For each term determined in the previous phase, it has to be decided whether it will be modelled as an entity, an attribute, or as a value (or indeed as a combination of these). For instance, given the terms ‘name’ and ‘customer’, it is common to model ‘name’ as an attribute of entity ‘customer’. This is done during the next phase.

B. *Ontology Definition Phase*

1. Define a concept for each of the supported seed terms using only terms that occur in the top-level ontology. The definition should have a unique concept identifier and the seed term itself should be kept as an attribute. In this stage, the concepts will not have any other attributes.
2. Define in a similar way concepts for all terms that are used to define the concepts in steps 1 and 2 and repeat this process until all terms are defined (or until a primitive term has been reached). If WordNet is used as a top-level ontology this means that all concepts on the path from the term to be defined until the root node(s) are modelled.
3. Define in a similar way as above the unsupported seed terms by linking them to the existing definitions. This may imply defining intermediate concepts. For instance, if the terms ‘X.21’ and ‘FDDI’ (denoting router interfaces) are to be defined and the term ‘router interface’ has been defined already, then it might be desirable to introduce intermediate classes representing LAN and WAN router interface.
4. Extend the ontology by assigning attributes to the defined concepts. This may involve the definition of additional concepts.
5. Complete the ontology by defining constraints over the defined concepts and their attributes. Define the constraints so that they exclude incorrect data, but do not use them to exclude unanticipated (unlikely) data.

The ontology as it stands covers the concepts that are relevant for application domain but it is not yet tailored to the specific resources. The last phase in the development of the shared ontology allows for the definition of additional concepts and attributes that enable a more convenient mapping onto the resources. This means that we extend the ontology with concepts required for the interoperation but not yet supported in the shared ontology.

C. *Resource Tuning Phase*
1. Extend the shared ontology with more specialised concepts that resemble the concepts defined each of the local ontologies.
2. Extend the shared ontology concepts with attributes and constraints so as to capture their intended meaning.

After this step, the shared ontology is completed and can be linked onto the resources via mapping functions. This is addressed in the next section.

5 Ontology Mappings

To overcome the mismatches between a resource and a shared ontology, an *ontology mapping* is defined. An ontology mapping is a partial function that specifies mappings between terms and expressions defined in a source ontology to terms and expressions defined in a target ontology. To enable bi-directional translation between a KRAFT network and a resource, two such ontology mappings must be defined. Here we describe the format that we use to specify ontology mappings.

In defining an ontology mapping, we begin by specifying a set of ordered pairs or *ontological correspondences*. An ontological correspondence specifies the term or expression in the target ontology that represents as closely as possible the meaning of the source ontology term or expression. For each term in the source ontology, we try to identify a corresponding term in the target ontology. It may not be possible to directly map all of the source ontology terms to a corresponding target ontology term. For some of the terms in the source ontology that cannot be mapped in this way, it may be possible to include them in the ontology mapping by defining correspondences between compound expressions. This leads us to the following classification of ontological correspondences:

class mapping: maps a source ontology class name to a target ontology class name;

attribute mapping: maps the set of values of a source ontology attribute to a set of values of a target ontology attribute;

attribute mapping: maps a source ontology attribute name to a target ontology attribute name;

relation mapping: maps a source ontology relation name to a target ontology relation name, and

compound mapping: maps compound source ontology expressions to compound target ontology expressions.

There are many subtypes for each of these types (more details on this can be found in [14]).

As the local and shared ontologies are not represented in the same format as that which is used for the CIF, the semantic transformation of CIF expressions by

wrappers is not done by directly interpreting the ontology mappings. Rather, the relevant ontology mappings are used as part of the specification of a wrapper. Consequently, developers have complete autonomy in the implementation of wrappers.

A pair of terms and/or expressions in an ontological correspondence are not necessarily semantically equivalent. However, when a wrapper translates a CIF expression, we need to ensure that the target CIF expression is semantically equivalent to the source CIF expression. If this were not the case, constraints passed to the CPE-mediator using terms defined in the shared ontology could express very different knowledge about a vendor's products than the original constraints expressed in terms defined in the local ontology. We ensure that the semantics of CIF expressions are maintained by defining *pre-* and *post-conditions* for each ontological correspondence. A wrapper that implements an ontology mapping must ensure that these conditions are satisfied when translating CIF expressions from the source to the target ontology.

6 Conclusions

KRAFT is an agent architecture that integrates knowledge rather than merely data. This is different from other resource integration projects such as OBSERVER [2], SIMS [3], and Carnot [4] in which merely data is interchanged. In the COIN project [5] the interchanged information consists of semantic values rather than pure data. Semantic values in COIN consist of a piece of data plus semantic information about the correct interpretation of the data. The expressiveness of this semantic information is limited. KRAFT uses the more expressive CIF constraint language to specify the exchanged knowledge. By translating local constraints into the shared ontology the design problem space can be reduced before the actual information retrieval. The aim is to provide the designer with a reduced problem space and to make the information retrieval itself more efficient.

Acknowledgements

The KRAFT project is funded by the Engineering and Physical Sciences Research Council (EPSRC) and BT. More information on the KRAFT project can be obtained from: http://www.csc.liv.ac.uk/~kraft/. The authors wish to express their gratitude to their colleagues of the KRAFT project and to Valentina Tamma.

References

1. Gray, P.D.M., A. Preece, N.J. Fiddian, W.A. Gray, T.J.M. Bench-Capon, M.J.R. Shave, N. Azarmi, M.Wiegand, M. Ashwell, M. Beer, Z. Cui, B. Diaz, S.M. Embury, K. Hui, A.C. Jones, D.M. Jones, G.J.L. Kemp, E.W. Lawson, K. Lunn, P. Marti, J. Shao, and P.R.S. Visser (1997) “KRAFT: Knowledge Fusion from Distributed Databases and Knowledge Bases”, *Database and Expert System Applications (DEXA' 97)*, Toulouse, France.

2. Mena, E., V. Kashyap, A. Sheth, A. Illarramendi (1996) "OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies", *Proceedings of 1st IFCIS International Conference on Cooperative Information Systems* (CoopIS'96), Brussels, Belgium.
3. Arens, Y., C.A. Knoblock, and W. Shen (1996) "Query Reformulation for Dynamic Information Integration", *Journal of Intelligent Information Systems*, **6**, 99-130.
4. Woelk, D., W. Shen, M. Huhns and P. Cannata (1992) "Model Driven Enterprise Information Management in Carnot", in C.J. Petrie Jr., (ed.), *Enterprise Integration Modelling, Proceedings of the First International Conference*, MIT Press, Cambridge, MA, USA.
5. Daruwala, A., C.H. Goh, S. Hofmeister, K. Hussein, S. Madnick and M. Siegel. (1995) "The Context Interchange Network", *IFIP WG2.6 Sixth Working Conference on Database Semantics (DS-6),* Atlanta, Georgia.
6. Finin, T., Y. Labrou, and J. Mayfield (1997) "KQML as an agent communication language", in Jeff Bradshaw (ed.) *Software Agents*, MIT Press, Cambridge, MA.
7. Bassiliades, N., and P.M.D. Gray (1994) "CoLan: A Functional Constraint Language and its Implementation", *Data & Knowledge Engineering*, **14**, 203-249.
8. Visser, P.R.S., D.M. Jones, T.J.M. Bench-Capon and M.J.R. Shave (1998) "Assessing Heterogeneity by Classifying Ontology Mismatches", in N. Guarino (*ed.*) *Formal Ontology in Information Systems*, (Proceedings FOIS'98, Trento, Italy), IOS Press, Amsterdam, p.148-162.
9. Jones, D.M. (1998) "Developing Shared Ontologies in Multi-agent Systems", *ECAI'98 Workshop on Intelligent Information Integration*, Brighton, U.K., August 25th.
10. Jones, D.M., T.J.M. Bench-Capon and P.R.S. Visser, (1998) "Methodologies for Ontology Development", *Proceedings IT&KNOWS Conference of the 15th IFIP World Computer Congress,* Budapest, Hungary.
11. Assadi, H. (1998) "Construction of a Regional Ontology from Text and its Use within a Documentary System", in N. Guarino (ed.) *Formal Ontology in Information Systems*, (Proceedings FOIS'98, Trento, Italy), IOS Press, Amsterdam, The Netherlands, p.236-249.
12. Miller, G.A., R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller (1990) "Introduction to WordNet; An On-line Lexical Database", *International Journal of Lexicography*, **3**(4), 235-244.
13. Bench-Capon, T.J.M., and F.P. Coenen (1992) "Isomorphism and Legal Knowledge Based Systems", *Artificial Intelligence and Law*, **1**(1), 65-86.
14. Swartout, B., R. Patil, K. Knight, and T. Russ (1997) "Toward Distributed Use of Large-Scale Ontologies", *Working notes AAAI-1997 Spring Symposium on Ontological Engineering*, Stanford University, Palo Alto, CA, USA.
15. Jones, D.M. (*forthcoming*) "Ontology Mappings", KRAFT Working Paper KPW55.