Bottom-up Integration of Ontologies in a Database Context

Jean-Christophe R. Pazzaglia
J-C.Pazzaglia@cs.cf.ac.uk

Suzanne M. Embury
S.M.Embury@cs.cf.ac.uk

Department of Computer Science, Cardiff University, Cardiff, Wales, CF2 3XF

Abstract

We describe a pragmatic approach to the creation of local ontologies for databases which must be integrated into multi-agent systems. Our approach avoids the introduction of an extra language, solely for modelling the required ontological information, by showing how to extend the expressiveness of a typical OODBMS data definition language. In this paper, we outline the principle behind our approach, and describe the implementation of a prototype system, which has demonstrated its feasibility.

Introduction

Ontologies have been proposed as a means of achieving consistent communication between agents in multi-agent systems. In order to do this, one ontology (or possibly a collection of overlapping ontologies) must be constructed which integrates the combined “world views”, or “local ontologies”, of the participating agents. In distributed information systems, however, many of these participants will be existing (legacy) databases, which bring with them some limited ontological information in the form of their schema. This database metadata can represent a significant contribution towards the description of the database agent’s local ontology, but it is too impoverished to play this role by itself. The schema information must be decorated with richer ontological information before it can be used in this way.

Proceedings of the 5th KRDB Workshop
Seattle, WA, 31-May-1998

(A. Borgida, V. Chaudhri, M. Staudt, eds.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-10/

The purpose of this paper is to describe a pragmatic proposal to extend a typical non-extensible object-oriented database language so that it can be used as an ontological language. The proposal has been validated by the implementation of a prototype system, which is being used in the context of the KRAFT Project [GPF+97]. Three ideas are behind this proposal :

- to eliminate the additional cost that would be caused by the introduction of a new language into the distributed information system,
- to reuse the conceptualisations already embedded in the different databases’ schemata, and
- to allow an incremental evolution of the ontological description of the domain covered by the different databases.

1 Background of the KRAFT project

The KRAFT project is a close collaboration between BT and the Universities of Aberdeen, Cardiff and Liverpool. It aims to develop a combination of database and artificial intelligence technology into a multi-agent system. The overall architecture is similar to those proposed in other projects, such as SIMS [ACHaK93] and InfoSleuth [BBB+97]. Where these projects have concentrated on distributed query processing, however, the focus of the KRAFT project is on the exchange and reuse of knowledge in the form of *constraints*. The KRAFT architecture, therefore, supports the extraction of *constraint* information (together with relevant stored facts) from various sites, and allows them to be combined with design goals and other user-supplied constraints in a common form which can be used to solve a specific problem.

Briefly, the KRAFT architecture distinguishes three different kinds of agent: wrappers, mediators and facilitators. As in other similar projects, the three different kinds of agent used in the KRAFT project all use a shared ontology to embody their commitment to a common semantics, thus making the network more easily extensible [HS97].

In addition to this conceptualisation role normally attached to an ontology, four further functions are associated with the ontology in a KRAFT network :

1. to provide an abstract data type to allow consistency inside the KRAFT network (*i.e.* global name space),
2. to attach a meaning to relationships appearing in the databases (such as *component of*) and to support reasoning over these,
3. to provide a way to store and to reuse standard mapping functions (*e.g.* between two known ontological concepts, such as standard unit translations), and
4. to provide a means to find "associated" or "similar" concepts (*e.g.* "pink" and "red").

When a resource joins the KRAFT network, for example, its owner must explicitly describe the way that resource commits to the shared ontology. The wrapper for this resource makes use of this information to perform its translation tasks, and also advertises its capabilities using the same terms. Our facilitator agents use ontologies to perform the knowledge-based reasoning required to achieve content-based routing of requests.

All this demands that the software agents within the KRAFT network must have access to the shared ontology in a machine-readable form. Furthermore, since the set of resources and agents participating in the system is likely to change over time (as new sources of data become available or as application demands alter, for example), the stored ontology must be extendible — both in terms of the conceptualisation itself and the modelling constructs available for describing that conceptualisation. Finally, since many of the resources wishing to join the KRAFT network will be databases, we wish to make use of the existing if impoverished ontological information already present in their schemas in extending the shared ontology. This paper describes a proposal for, and prototype implementation of, a software component which meets these requirements.

2 A *bottom-up* approach to extract ontology

We use our shared ontology, or more precisely an interconnected network of domain specific ontologies, as described in [GPF+97], to provide a *self-contained* conceptualisation of problems domain in a computer manipulable form. Commitment of a new resource to the network can be a complicated and time consuming business. One of the keys to speed this process up is the reuse of the ontological information already stored in the shared ontology and at the resource itself. Depending on the nature of the resource, we can directly reuse well defined ontology such as a Boolean algebra ontology for boolean constraint solvers or commonly accepted concepts already present in the shared ontology such as number, physical magnitude,etc.

When a resource joins the KRAFT network, we are faced with two alternatives :

1. an ontology already exists in the domain of interest of the resource, or
2. an ontology must be built or extended to take account of this *new* domain of interest.

In both cases, if we want to avoid misinterpretation and allow relevant advertisement, we need to extract the meaning of the information which will be exported from the resource to the KRAFT network. This step, part of the ontological commitment, involves the conceptualisation vocabulary and objects which are used by our resource. Since we have already developed ontologies to share information, we can, naturally, follow the same scheme to describe this conceptualisation of the *local* ontology of the resource.

In fact, several kinds of resources already embed an *ad-hoc* and explicit, *if not complete*, conceptualisation of their domain. Relational and Object Oriented Database resources belong to this category since their schema is, in essence, an abstraction which classifies data. From the perspective of someone who wishes to create a local ontology, for such a resource, the aim is to share the commonly accepted concepts already present in the shared ontology and/or the conceptualisation already embedded in the joining resource. We propose to support this task by reusing the information present in the schema thanks to the help of some *soft* extension to an object-oriented data definition language.

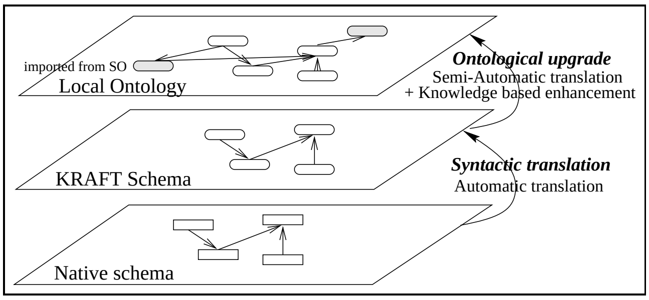

Figure 1: Bottom-up extraction of ontology

The different stages of this extraction process can be seen in figure 1 and are described below :

Native schema The database administrator must first produce a KRAFT exportable view of the resource in the native language.

Kraft Schema Technology developed at Cardiff within the ITSE project [BAFG96] achieves the syntactic translation from the resource's native language to the Kraft schema language (aka P/FDM).

Local Ontology The local ontology adds knowledge and further relationships between the entities in the translated schema.

This approach can be seen as a computer assisted extraction of the ontology since the two steps use partially automated translation tools. Obviously, the second step requires human intervention since the enhancement of a schema to an almost[1] *self-contained* conceptualisation cannot be fully automated.

This ontology can be now used to provide a domain specific node in our ontology network, or as a basis to provide a semantic translation to the rest of the Kraft network. In order to fulfill the commitment of the resouce to its ontology, and to be compliant with our translation scheme, we need to describe a way to translate expressions in terms of this local ontology to equivalent expressions in terms of the Kraft ontology. This task is achieved by describing mapping functions between the local and a target ontology. Another choice, commonly adopted, is to describe mapping functions directly from the database schema. However, we think that this first step (*i.e.* the ontology extraction) has to be done since knowledge about the resource must be decribed.

We will now describe how we upgrade an object-oriented data definition language to support this task. In the next section, we explain why we have been convinced by the study of Ontolingua related works that our planned extension is suitable as a language in wich to describe ontologies.

3 Ontological languages

Before describing our ontological tool suite, we will first introduce the Ontolingua language suite [Gru93, GF92, SK97]. We will then briefly highlight the differences between standard object oriented database language (OODL) such as O_2 [LRV88] or other ODMG [Cat97] compliant systems and OKBC, the API supported by Ontolingua to exchange ontologies. Finally, we will present a framework in which to describe ontologies.

3.1 Ontolingua

Ontolingua provides a set of tools to edit ontologies and, support different languages. The language used by Ontolingua is a syntactic superset of the Knowledge Interchange Format [GF92] that captures standard knowledge representation conventions in a frame-based language. Ontolingua also supports some simple inferences in order to check the consistency of ontologies. The OKBC API[2] allows the export of frame based ontology descriptions.

The success and wide acceptance of Ontolingua as a *de facto* standard for describing ontologies calls into question the value of projects designing yet more ontological description languages. However, the Ontolingua tool suite assists in *top-down* development ontologies, whereas many real applications demand the *bottom-up* construction of a shared ontology by reusing a number of pre-existing conceptualisations. A typical example would be the construction of a distributed information system within an organisation, from a collection of databases which have already been independently constructed.

It is this problem that our work aims to address. However we believe that classical database data models are not expressive enough to be able to capture the required richness of ontological knowledge, even when it is known and available within the organisation.

3.2 Object model comparison

The main difference between OODL's and OKBC is seen in the object oriented model underlying them. On the one hand, OODL's use a classic *class-based* model, while in the other OKBC uses a *frame-based* language.

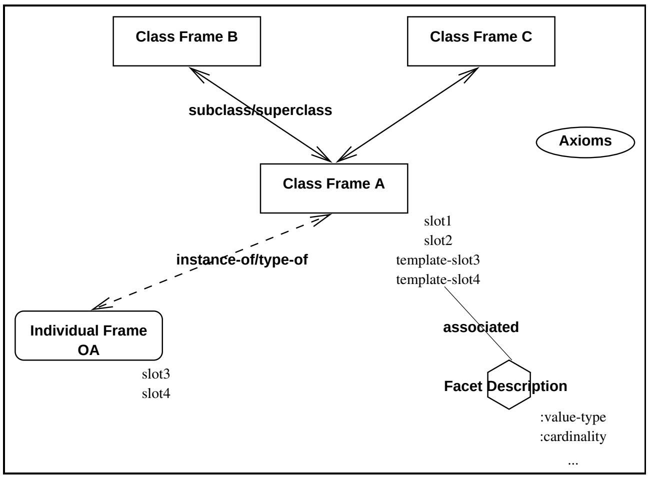

Figure 2: OKBC overview

However, this difference is not as important as it might seem since OKBC provides (and Ontolingua tools strongly promote) the use of a *Class frame* (see figure 2). Whereas an attribute description in OODL's

[1] Even if we follow this bottom-up extraction scheme, we allow user to import common concepts directly from the shared ontology.

[2] Former name was GFP

(also known as *template-slot* in the Ontolingua terminology) is only controlled by the class, OKBC can use a *facet* to describe this attribute. It captures the meaning and can limit not only the type but also particular values of an attribute or even characterise the relationships between several attributes (*e.g.* disjunction...). Furthermore, *KIF-axioms* express relationships between concepts and can be used to check the consistency of ontologies.

4 Ontolological language description

Since a class-based framework seems suitable for describing ontologies, we chose to extend the expressiveness of our existing OODL, rather than incorporate an additional language for specifying ontological information. Our idea is to provide a class based framework with extensible capabilities to describe ontologies. We describe a schema which captures our language definitions, and use standard metadata and database constructs to both populate this meta-description and also to extend the ontological meta-model.

4.1 Ontology description and the compilation process

The first step is to describe the ontological model as a standard schema. Basically, this description is the meta-description of our ontological language. Since an ontological language is focused on descriptive aspects, behavourial description (such as method calls) are outside our current scope. We therefore describe only the structural part of our language. The actual description can be seen as a re-engineered version of the metadata schema focused on ontological aspects. It provides support for documentation and hides some of the internal database aspects which are not useful in this context. Starting from this point and using the metadata schema facilities provided by the underlying database system, we provide a set of methods in order to reify[3] each standard OODL element. The main difference between our system and typical OODL's metadata is that we have now the capability to extend the metadata[4] and add useful information to our schema.

We also have the ability to extend the ontological model. Ontological experts can write these extensions in the new ontological language (OL). Such extensions can complete a hierarchy or add different kinds of relationships, and can, for example, capture data semantics, as facets do. This offers a useful link framework and a way to enable knowledge reasoning. The complexity of ontological descriptions can be adapted to the needs of the agent network and can therefore follow the evolution of application requirements and capabilities.

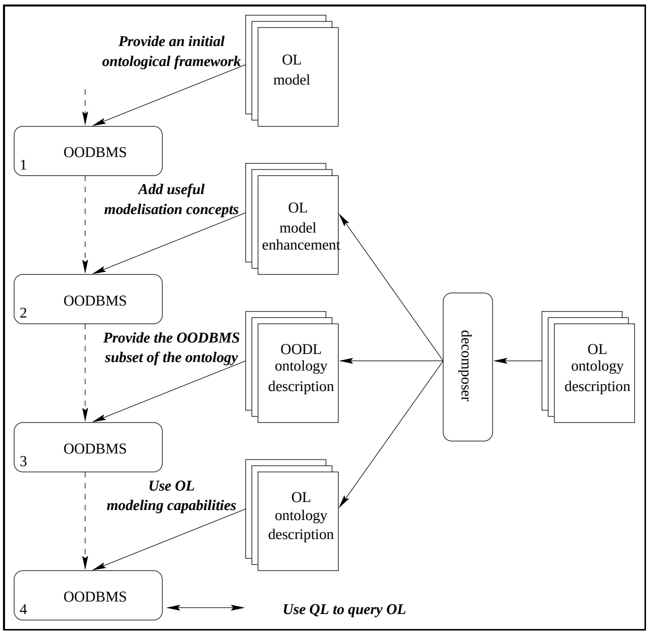

Figure 3: OL initialisation stages

An ontology description in OL consists of several standard OODL files. The different components are loaded in the OODBMS during an initialisation stage which converts the standard OODB into the extended OODB with additional ontological capabilities (shown in figure 3). Three main components can be distinguished :

Enhancement to the Ontological Model : an OODL schema, interpreted in our *ontological model* module which extends the standard Ontological Model, can add specific ontological extensions required by some application, and

OODL subset of the ontology : a standard OODL schema which describes the overall architecture of the ontology with the usual OODL semantics,

OL Ontology description : Useful information such that facets descriptions or relations typing can be added conformly to the Ontological Model.

Finally, the OODL database stores the ontological description of the application domain and can be queried with standard OODL's schema statements.

4.2 Prototype overview

Our prototype is implemented in P/FDM [Obj], an object-oriented database using the functional data

[3] To reify: to regard (something abstract) as a material or concrete thing

[4] As for example we can do with languages like CLOS [KdRB91].

model and an enhanced version of the Daplex language including metadata and constraints.

The basis of our prototype is a research DBMS called P/FDM that was developed at the University of Aberdeen [Obj]. P/FDM is based on a functional data model with object-oriented extensions. Despite this rather unusual history, the data model supports the principal features of an object-oriented data model, including object identity, (single) inheritance between classes and the ability to describe the behaviour of an object by defining methods on classes. P/FDM supports uniform querying of metadata using the standard query language, and allows a restricted set of updates to metadata at run-time. Unlike many object-oriented languages, however, and in common with the majority of object databases currently available, it does not allow modification of the meta-schema itself.

We have extended this language in order to allow further ontological information to be expressed within the schema, alongside the database schema information that is already there. In order to allow the existing P/FDM language processor to continue to parse the extended schemas, we chose to label the additional, ontological statements with two special prefixes: `%%e` and `%%o`. To P/FDM, these lines appear to be comment lines, and so are ignored. The ontological language processor, however, is able to detect these lines and to handle them accordingly.

```
%%e extend module ontological_model
%%e declare om_constrained_slots
%%e        ->> om_slots_method
%%e declare allowed_value(om_constrained_slots)
%%e       ->> string;

%%o om_constrained_slots(om_documentation)
%%o        = 'slots value in a set of string'
%%o om_documentation
%%o        (allowed_value(om_constrained_slots))
%%o               = 'set of allowed value'

declare date ->> entity
%%o om_documentation(date)
%%o        = 'this class allows to manipulate date'
...
declare month(date) -> string
%%o include month(date) in om_constrained_slots
%%o allowed_value(month(date))
%%o        = ['January','February',....]
```

Figure 4: A basic schema after ontological upgrade

We will illustrate this approach using a simple example, shown in figure 4. In this example, we distinguish the three components of an OL file:

1. The OODL schema file describes the class `date` (on the lines which do not begin with special character).

2. Lines starting with `%%e` describe enhancements to the ontological meta-description.

3. Lines starting with `%%o` describe ontological information.

The enhancement in this example consists of the introduction of a subclass `om_constrained_slots` of the class `om_slots_method` which allows the user to constrain values of certain attributes. The other information describes components of the schema. The slot `month(date)` is declared to be an instance of *constrained slots* and takes its value from a particular set of allowed values (*i.e.* [January, February, ...]). A query to the reified object instance of `om_constrained_slots` identified by `[month, date]` will give the values allowed for this attribute. This information can be useful both during the commitment phase when resources first link to the agent network, and also to search for agents talking about *January* during facilitation. The next section describes evolution of metadata during the compilation of this particular example.

4.3 Evolution of metadata

The evolution of both OODL metadata and OL during the compilation process are shown in the figures 5, 6, 7, and 8. These figures (bold font emphasize recently system's update) can be seen as snapshots of the memory in the numeroted OODBMS stages shown in the figure 3 during the compilation of the example described above.

The figure 5 shows the basic framework provided by our system as well as established links between them and the frozen metadata provided by our OODL. This framework is used to provide automatic translation to basic concepts.

The class `om_constrained_slots` is introduced, by the person providing a suitable ontological model, as subclass of `om_slots_method` (figure 6). By using the OL querying capability and in accordance with the semantic associated with this ontological model, a program will be able to reason about concepts and the instances installed in the following stages.

The classic OODL schema populates both classic metadata and the OL representation in the figure 7. The last stage unable to migrate the attribute `month` in order to express values allowed. The same manipulation could be done in order to link the number of days with the month. This would be done at the `class` level since this constraint will link several attributes.

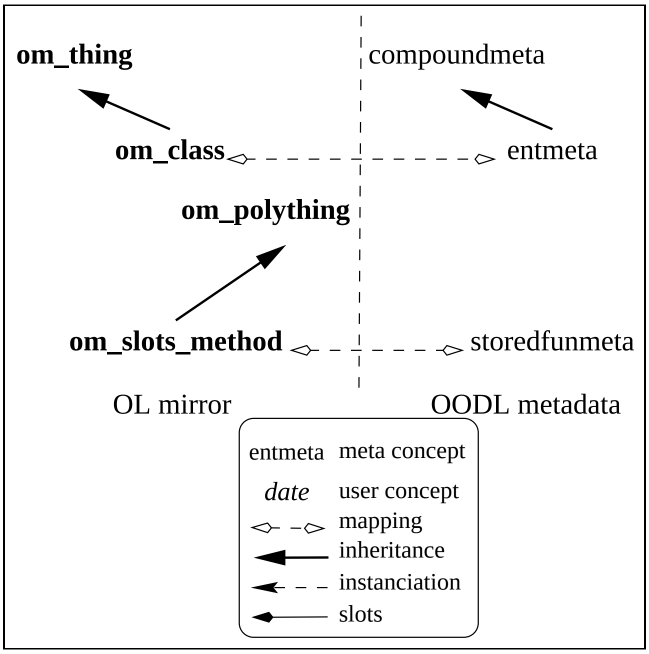

Figure 5: Evolution of the representation(1/4)

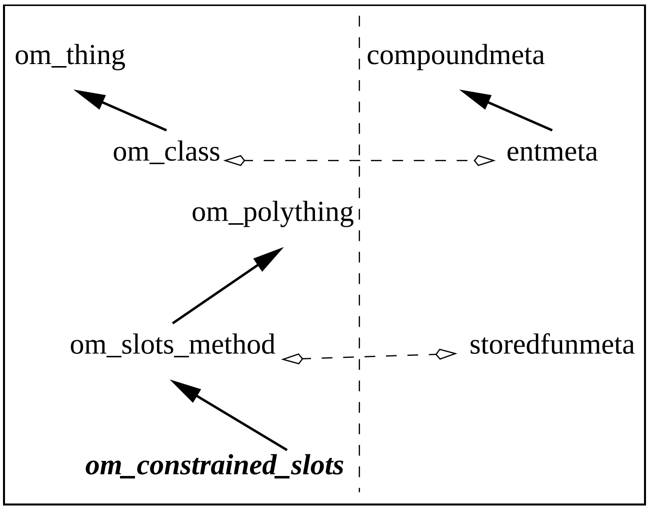

Figure 6: Evolution of the representation (2/4)

Conclusion

We have described an approach to the integration of ontological information with existing database metadata, which avoids some of the practical problems inherent in the creation of ontologoes from legacy database systems. In our prototype, we have shown how to bypass the extensibility limitations inherent in typical OODBMS metadata facilies. This is done by using introspection to create a mirror image of the metadata at the instance level, thus allowing it to be extended as if it were an ordinary database schema.

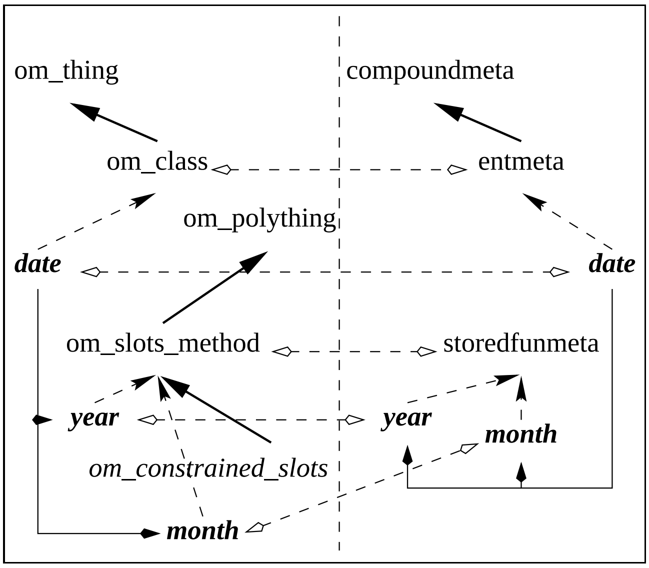

Figure 7: Evolution of the representation (3/4)

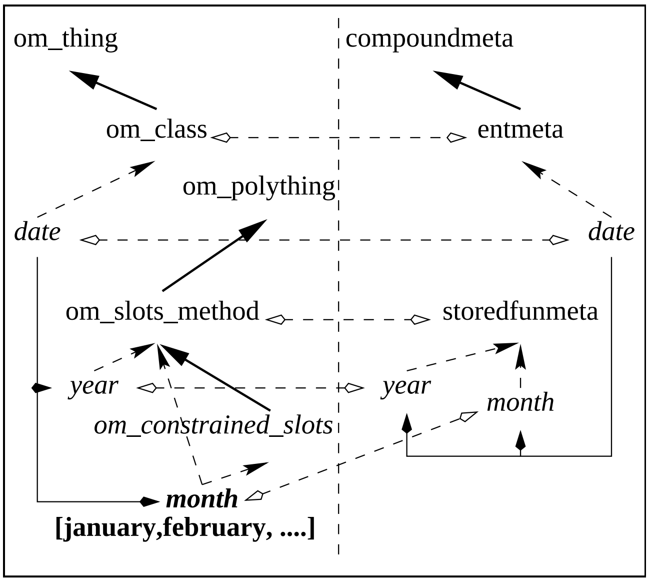

Figure 8: Evolution of the representation (4/4)

This simple approach provides us with the power to enhance the expressiveness of our modelling language as and when required. We have effectively created an *open framework* by which to add meaningful information to the *naked* schema used in database system. Furthermore, the framework enables better conceptualisation and automatic translation/control support since the underlying ontological model provides a clear semantics.

At this stage in our research, we have dealt only with the structural aspects of our system and few difficulties was encountered during the development. Obviously, an extensible meta-object system would have eased the implementation task, and would have freely offered the modelling capabilities we have gained. With this in mind, it would seem to be a relatively simple matter to provide, for example, an ODMG-compliant meta-program able to support this way of constructing an ontology from a schema. However, it should be remembered that many legacy systems will not have this kind of capability.

The next step in our work is to enhance our current ontological model to allow us to add meta-descriptions of KIF-like relationships and attributes to our schemas. When this is completed, we plan to allow batch interaction via OKBC.

After this first step, databases containing data about wide and local area network equipment, developed as part of the KRAFT project will be used as embryonic ontologies and enhanced in order to benefit from the Ontological Model. The next stage will be to study how *constraint* information can be represented in the ontological description, and extracted during commitment of several databases.

Acknowledgements

We would like to thank Philippe Marti and Wernher Behrendt, for fruitful conversations regarding the OL (O/PFDM) model, Graham Kemp for his great P/FDM support and the rest of KRAFT team. The KRAFT project is funded jointly by the EPSRC and BT.

References

[ACHaK93] Yigal Arens, Chin Y. Chee, Cgun-Nan Hsu, and Craig a. Knoblock. Retrieving and Integrating Data from Multiple Information Sources. *IJCIS*, 2(2):127–158, 1993.

[BAFG96] W. Behrendt, M.H. Ashwell, N.J. Fiddian, and W.A. Gray. Migration Tools for Heterogeneous Databases in Wide Area Networks. In *Workshop on Information Technologies and sytems*, 1996.

[BBB+97] R. Bayardo, W. Bohrer, R. Brice, A. Cichocki, G. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. Infosleuth: Agent Based Semantic Integration of Information in Open and Dynamic Environments . In *Sigmod*, 1997.

[Cat97] R. G. G. Cattel. *Object Database Standard: ODMG 2.0*. Morgan Kauffmann, 1997.

[GF92] Michael R. Genesereth and Richard E. Fikes. Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.

[GPF+97] P.M.D. Gray, 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. KRAFT: Knowledge Fusion from Distributed Databases and Knowledge Bases. In *DEXA*, 1997.

[Gru93] Thomas R. Gruber. Portable Ontology Specifications. Technical Report 92-71, KSL, 1993.

[HS97] Michael P. Huhns and Munindar P. Singh. Ontologies for Agents. *IEEE Internet computing*, November 1997.

[KdRB91] Gregor Kiczales, Jim des Rivières, and Daniel G. Bobrow. *The Art of the Metaobject Protocol*. MIT Press, 1991.

[LRV88] Christophe Lecluse, Philippe Richard, and Dernando Velez. *O_2, an Object Oriented Data Model*, chapter 3.6. Morgan Kaufmann, 1988.

[Obj] Object Database Group – University of Aberdeen. *P/FDM V10 – User's manual*.

[SK97] SRI and KSL. *Generic Frame Protocol – version 2*, 1997.