

An Experience with Ontology-based Agent Clustering

Pepijn R. S. Visser¹

Valentina A. M. Tamma

CORAL - Conceptualisation and Ontology Research at Liverpool
Department of Computer Science, University of Liverpool
PO Box 147, Liverpool, L69 7ZF
United Kingdom
(tel.) +44.151.794.3709
{pepijn,valli}@csc.liv.ac.uk

¹From January 99 at: SAP-AG HRMS Development D-69190 Walldorf
Pepijn.Visser@sap-ag.de

Abstract

This article presents a structure of multiple shared ontologies to integrate heterogeneous sources. This specific structure is intended to be easy to implement to maintain and to scale, and also close to the human model of conceptualisation. The structure has been investigated in a small scale experiment set in the domain of the international coffee preparation. The experiment has also addressed, an identification of the different types of heterogeneity that can affect the resources.

1 Introduction

In this article we discuss a small-scale experiment in finding architectures for the integration of heterogeneous resources. In this architecture resources are clustered on the basis of the resemblance between their conceptualisations of their domains. One of the motivating ideas is that - as with inter-person interaction - resources with a similar conceptualisation can have more 'in-depth' conversations than those who share less of their conceptualisation. The architecture investigated is intended to reconcile different types of heterogeneity, and for this reason one of the stages of the

The copyright of this paper belongs to the papers authors. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage.

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

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

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

experiment has been to identify the types of heterogeneity that can affect distributed systems. Furthermore, this architecture is also intended to be more convenient to implement and give better prospects for maintenance. The experiment is set in the international coffee-preparing domain and builds on ideas that have been launched in two previous papers [Sha97], [Vi98b]. The experiment is investigated as a spin-off of the KRAFT project [Gra97], [Gra98], a knowledge integration project that is currently conducted in the United Kingdom. In section 2 we introduce the context of our research, which is the KRAFT project. In section 3 the background of the ontology-clustering idea is discussed while section 4 presents the motivating scenario. In section 5 we define kinds of heterogeneity. Section 6 then presents a so-called 'ontology cluster' architecture and in section 7 we illustrate how communication between resources is performed in this architecture. Finally, in section 8 we look ahead and draw conclusions.

2 The KRAFT project

KRAFT is a research project on the integration of heterogeneous information by means of an agent architecture [Gra97, Gra98]. The project focuses on the integration of data plus relations between data items (in the form of constraints) rather than merely data or data enriched by context information. In this KRAFT differs from integration projects, such as OBSERVER [Men96], SIMS [Are96], Carnot [Woe92], and the COIN project [Dar95]. The project is conducted by the Universities of Aberdeen, Cardiff and Liverpool in conjunction with British Telecommunications plc. and started in May 1996.

The KRAFT architecture permits 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 semi-structured web

pages. Users interact with the middleware architecture via a user agent, which allows them to formulate their queries and which also presents the results to the users. The agent-middleware architecture has three types of agents: wrappers, facilitators and mediators. The *wrapper* links the external resources onto the KRAFT middleware. To do so the wrapper performs language, protocol and ontology translations from and into the KRAFT application internal 'standards'. The *facilitator* is the white and yellow page facility. It is consulted by the other agents to 'recommend' services that match the functionality they require. The *mediator* is the KRAFT internal problem solver. Mediators analyse and decompose information requests 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 a set of performatives based on KQML [Fin98]. 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 its wrapper with the request to recommend an agent who can deal with its query (performative: *recommend-one* or *recommend-all*). The facilitator then consults its internal database of advertised capabilities and replies by forwarding (performative: *forward*) the appropriate advertisement(s) to the user agent. This advertisement allows the user agent (via its wrapper) to communicate directly with the agent that submitted this advertisement. The latter agent will then be sent the query (performatives: *ask-one*, *ask-all*). By contacting the facilitator every time an agent that needs the service of another agent the network can adapt to situations in which resources are (temporarily) out of service, or, select the most appropriate agent to deal with the query.

Resources in KRAFT are legacy systems and are heterogeneous with respect to their ontologies (more on the resource heterogeneity in section 5). The strategy adopted in KRAFT is that of having *shared ontologies* that serve as an 'ontology lingua franca'. Information from resources is translated from the local ontology into one of the shared ontologies and vice versa. Many architectures for resource integration use a single shared ontology, even though having one shared ontology is not always optimal, especially for larger applications. The two prototypes that have been developed in the KRAFT project so far have used a single shared ontology as well, nevertheless KRAFT is exploring alternative solutions that exploit multiple but smaller ontology standards in order to overcome these disadvantages. In the next section the alternatives of a single shared ontologies vs. multiple ontologies are illustrated.

3 Multiple shared ontologies

Literature about ontologies is vast and addresses different problems such as the design of ontologies [Fri97] or the reuse of ontologies [Usc98].

In the remainder of this section the attention is focused on different solutions to the integration of heterogeneous resources with different ontologies, and both some approaches that are in the literature and a novel approach are presented. All the approaches, however, are based on the some functions performing the translations between the ontologies (shared or not). These functions are often called in the literature *mapping functions*: Concepts can be shared between different resources if an appropriate mapping function can be found that translates a concept understood by one resource into a concept that is understood by another resource. This is the minimal requirement for two resources to share knowledge.

The integration of heterogeneous sources can be accomplished without an intermediate ontology. This is the so-called 'one-to-one' approach, where for each ontology a set of translating functions is provided to allow the communication with the other ontologies. Such an approach would require in the worse case, that is if the mappings are not isomorphic, the definition of $(n^2 - n)$ mapping functions, if n ontologies are comprised in the structure. This is what happens in the system OBSERVER [Men96], where translations of concepts from a source ontology into a destination one are accomplished by defining appropriate mapping functions. This approach only seems feasible only if there are a few ontologies (resources). It also would not be very scalable because if a new resource is added to the structure this approach requires the definition of n new mapping functions.

Many architectures to integrate resources comprise a single shared ontology, an example is given by InfoSleuth, [Bay97] and by the KRAFT architecture. Whether such approach is conceptually realistic is a matter of debate [Sha97]. The drawbacks of dealing with a single shared ontology are similar to those of any standard (see also: [Vi98b]). Often, standards are not very convenient to use since they have to be suitable for all potential uses. Also, the task of defining such standards is often lengthy and complicated. Moreover, committing to a standard restricts the degree of heterogeneity that may exist between those using the standards, and, last but not least, standards - by their nature - resist changes, partly due to the aforementioned reasons.

In contrast to an approach in which all resources share one body of knowledge here we propose to locate shared knowledge in multiple but smaller shared ontologies. This approach, which is thought to be more flexible and scalable, is referred to as ontology-based resource clustering, or shortly, ontology clustering [Sha97]. Resources no longer commit to one comprehensive ontology but they are clus-

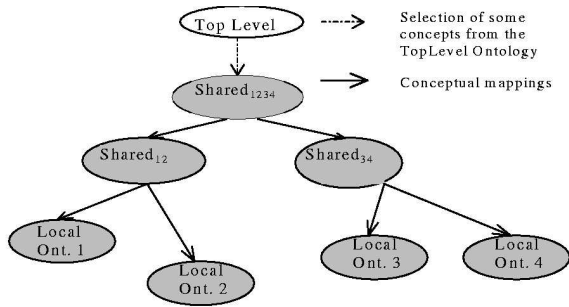


Figure 1: The hierarchy of multiple shared ontologies tered together on the basis of the similarities they show in the way they conceptualise the common domain.

Ontology clusters are then organised in a hierarchical fashion. In the top level cluster there are the general concepts that are shared by all the resources, where in the lower level clusters the concepts defined in the higher level ontology clusters are extended and characterised in order to provide the local conceptualisation of one or more resources. Therefore, each sibling cluster specialises the concepts that are in its parent cluster. The lower level clusters have more precise concept definitions than the higher levels, making the latter more abstract. Since different siblings can extend their parent cluster concepts in different ways the cluster hierarchy permit the co-existence of heterogeneous (sibling) ontologies. Figure 1 illustrates this particular structure, where *Local Ont. 1*, *Local Ont. 2*, *Local Ont. 3* and *Local Ont. 4* are the local ontologies, *Shared₁₂* is the ontology shared by the local ontologies 1 and 2 and analogously *Shared₃₄* is the ontology shared by the local ontologies 3 and 4. *Shared₁₂₃₄* indicates the ontology shared by the two below that is *Shared₁₂* and *Shared₃₄*.

This idea has been used in a small experiment conducted within a (telecommunications) business process [Vi98b] and in the remainder of this paper an another experiment with the principles of multiple-ontology architectures is illustrated. Both experiments are conducted as independent spin-off projects of the KRAFT project.

4 Motivating Scenario

Our experiment is set in the domain of preparing coffee. Four agents from four different countries are hypothesised to tell each other what coffee means in their country. In the remainder the word agent refers to either a human or a software one. Software agents were not implemented in the coffee domain experiment. The experiment focuses on investigating the structure of multiple shared ontologies for the knowledge sharing, therefore the implementation of software agents was deemed beyond the scope of the experiment. The agents are *François* from France, *Nicola* from Italy, *Charles* from the United Kingdom and *Klaas* from the Netherlands. The agents share a basic understanding of the domain in that they know what the basic ingredients

Concept	Description
Coffee ingredient	The substance derived from the coffee plant that is used to prepare coffee.
Kitchen appliance	A physical object that is used as kitchen tool.
Coffee drink	A drink produced by using some coffee ingredient, water and a kitchen appliance.
Coffee maker	A kitchen appliance that produces coffee drink and is composed by a filter component, a liquid container and a coffee holder that holds either some solid substance or some liquid substance.
Heating device	A kitchen appliance that is used to warm something.
Hot water	Is water with a specific temperature greater than 90 Celsius degrees.
Solid container	Some sort of substance container.
Liquid container	Some sort of substance container.

Table 1: Concepts shared by all agents are and that the coffee powder (where powder refers either to the ground coffee or to the instant coffee grains) and hot water somehow need to be combined, but there are regional differences. Agent know how coffee is made in their nation, what the ingredients are and the tools necessary to prepare it, and what their name is. Stereotyping these nations a bit further we here assume that François only knows about *cafetière* coffee¹, Nicola only knows espresso coffee (prepared with an espresso coffee maker²), Charles only knows about instant coffee (prepared with a kettle), and Klaas can only make coffee with an electric coffee maker. The shared concepts however, should guarantee that dialogues about the meaning of unfamiliar concepts are possible and it will be illustrated that agents who share more concepts can have more 'meaningful' conversations. Table 1 shows the most important shared concepts:

Besides the universally shared concepts, the agents also have a set of local concepts about coffee drinks, such as the concept of coffee maker in their countries or the type of coffee used in their nations to prepare a coffee drink. These concepts are related to more general knowledge such as that the coffee drink has water and coffee as ingredients, that a coffee maker is a kitchen tool etc. The local concepts for each agent are illustrated in Table 2.

Although all the agents share the basic concepts above those are not the only shared concepts. In fact some more knowledge is shared by a restricted number of agents. For

¹A *cafetière* is a jug in which ground coffee is placed. After pouring hot water on the coffee a filter is pressed through the jug.

²An espresso coffee maker has a water tank, a filter (that also holds the coffee), and a coffee reservoir. Coffee is made by forcing boiling water under pressure (in the water tank) through the ground coffee that is held in the filter.

François	Nicola	Charles	Klaas
Ground coffee powder	Ground coffee ingredient	Instant coffee grains	Ground coffee
French coffee	Espresso	Instant coffee	Dutch coffee
Cafetière (composed by a jug, a filter device and producing French coffee)	Espresso coffee maker (composed by a water reservoir, a coffee reservoir, and a filter and coffee holder, and producing Espresso coffee)	Mug	Electric coffee maker (composed by a jug, a water reservoir and a filter and coffee holder constituent and producing Dutch coffee)
Kettle		Kettle	

Table 2: Concepts not shared by all agents

example, from the description of the local concepts in table 1 it is possible to notice that the concept of Electric coffee maker, known by Klaas, is quite similar to the one of Espresso coffee maker, known by Nicola. Indeed both are coffee makers and both have a component where the water is put, a component to hold the coffee ingredient that also acts as filter during the coffee preparation and a component where the coffee drink is saved.

Communication between the agents centres on finding the similarities in the conceptualisations. That is, an agent tries to explain to another agent how coffee in his country is made, using his own concepts as starting point. He will try to understand what the other agent knows and explains unknown concepts in terms familiar to the other agent. To illustrate the process, we here give an example of the type of interactions between the agents in the experiment. This specific dialogue refers to a conversation between Nicola and François (disregarding their native languages to preserve clarity).

Nicola: What do you use to make coffee?
 François: I use hot water, ground coffee powder, a kettle and a cafetière
 Nicola: How hot is the water?
 François: Hot water is a kind of water that has temperature higher than 90 degree Celsius
 Nicola: You use ground coffee powder, what is that?
 François: Ground coffee powder is the same as ground coffee
 Nicola: What is a kettle?
 François: A kettle is a heating device
 Nicola: What is a cafetière?
 François: A cafetière is a coffee maker that consists of a jug and a filter device
 Nicola: Does the cafetière have a water reservoir?
 François: A cafetière has a jug that is a substance container, where substance can be either solid or liquid. The jug can contain liquid that is hot water or the actual coffee drink
 Nicola: What is the ground coffee kept in?

François: Ground coffee is kept in the jug, as it can also contain solids

Nicola: Does the jug have a filter, then?

François: No, the jug does not have a filter, but the cafetière has a filter device that is a filter component.

Nicola and François are struggling to understand each other because they share only very general concepts about coffee. Moreover, the dialogue does not completely explain the relationships between the meaning of terms. For example, Nicola will be able to understand that a jug can contain both liquids and solids, but he will not be able to fully infer that the jug in the cafetière corresponds to both the water reservoir and the coffee reservoir and the filter (in that it contains ground coffee) in the Espresso coffee maker. A conversation between Nicola and the Dutch agent Klaas will be less troublesome since these agents share more concepts.

5 Heterogeneity Definition and Assessment

The agents in the scenario are heterogeneous both with respect to their languages and their ontologies. The first question addressed in this experiment is what forms of heterogeneity exist between our agents. The vast amount of literature on the integration of heterogeneous information sources is sometimes confusing regarding the kind of heterogeneity that is dealt with, especially where the knowledge engineering and data modelling fields meet. This makes it less easy to compare the different approaches [Tam98]. Although many efforts have been made to classify heterogeneity between databases [Mar90], [Kim91], [Cer93], few efforts are known classifying heterogeneity between ontologies [Vi98a]. In this section we attempt to define heterogeneity in distributed agent architectures and position the four agents in our experiment with respect to the distinguished heterogeneity kinds.

To define heterogeneity between resources of a distributed agent system we take a *system design process perspective* that is based on the design of such a system. The idea being that "difference in character" [All90] between resources results from differences in the choices that are made during the various stages of the design of a system. We commence by describing an imaginative bird's-eye perspective of steps of a software engineer starting from the perception

of the world up to the implementation of the resources. The steps are selected on the basis of their relevance for our present purposes, and are by no means exhaustive. It is not based on an in-depth analysis of human software engineers, and any control structure is omitted. In practice this means that some series of steps may be performed at once, some series of steps may be iterated, and other steps may be omitted. We do claim however, that most steps (if not all) have been identified in the knowledge engineering literature. The steps identify the choices that are made, the results they produce, as well as some examples of these results. We start with the steps that are taken to build the individual resources and divide these into three phases:

- (A) *Ontology definition phase* (steps A1-A7),
- (B) *Resource application design phase* (steps B1-B4),
- (C) *Resource application implementation phase* (steps C1-C3).

The complete set of steps is given in Table 3. We assume that the resources in the architecture are knowledge systems and / or databases, and that the designer forms an ontology (that may not have been made explicit) of his domain as a precursor for the design of each of the resources.

In each step of the process, different people may make different decisions and in the design of a distributed architecture this may lead to the introduction of a (new) kind of heterogeneity between the resources. Already during the first listed step - perception (A1) - the first heterogeneity may be introduced. Although it is possible, it is not necessary that all steps introduce heterogeneity underlying the applications. Different choices for the conceptual modelling language, for instance, can but are not very likely to make the resulting applications more or less heterogeneous. However, if these steps do introduce heterogeneity then, using the table, we can identify and refer to the different kinds of heterogeneity more accurately. Ontology heterogeneity can, for instance, arise from any (possibly all) of the seven steps identified in phase A, thus identifying seven primitive types of "ontology heterogeneity" (cf. [Vi98a]).

Besides the overloading of the term "ontology heterogeneity" in the literature, we also find the term "language heterogeneity" to be used for different things. In the table we have already identified five different languages that are involved in the process of making a resource. The first language is the (natural) language from which the vocabulary (used to refer to the concepts) is extracted (step A3). The second language is the ontology language, which is (usually) a formal language, which is at the least machine-readable (step A6). The third and fourth languages that may be used are the conceptual and formal modelling languages (step B1 and B3). The fifth and last language is the implementation language in which the application is realised (step C2). This illustrates that the phrase "the systems have a language heterogeneity" is highly ambiguous

(and should hence be used with care). Perhaps it is interesting to note that the heterogeneity that can be introduced during steps A3, A4, and A6 can all be said to be both "language heterogeneity" and "ontology heterogeneity". The indexes in the table can be used to point out more accurately which kind of heterogeneity is meant.

So far, the steps that are taken in the design of the individual resources have been listed. There are other kinds of considerations that have to be taken into account if the resources are to be embedded in an agent architecture. These considerations are the minimal requirements to permit resources to communicate. Again, we stress that the requirements serve only to identify different types of heterogeneity rather than completeness from a software-engineering perspective. In designing an agent architecture some of the actions taken can introduce further kinds of heterogeneity. Of these, we mention the (D1) choice of the Agent Communication Language (such as KQML), (D2) the choice of the Content Language within this communication language (such as KIF or PROLOG), (D3) the ontology that is used to express the information exchanged, and (D4) the design and (D5) implementation of the agent interface of the resources. The reason for focusing on these steps is that they identify another kind of "ontology heterogeneity" (bringing the total up to eight) and two more kinds of "language heterogeneity" (bringing the total up to seven). Both the agent communication language and the content language can, in principle, differ between agents. Notice, that although ontology heterogeneity is already covered in the ontology definition phase, the agent communication ontology may be different from the agent's ontology and hence introduce another type of ontology heterogeneity.

Table 3 and the further considerations allow us to characterise our architecture with respect to its heterogeneity status. We assume that two agents perceive the objects in the world in an identical fashion (A1), but they differ in the way they conceptualise them and in the way they create their ontology (A2-A5), although they use the same ontology language (A6). From the agent architectural viewpoint, we assume that the agents use the same Agent Communication Language (D1) and the same Content Language (D2), but possibly different communication ontologies and design and implementation of the agent interface of the resources (D4 and D5). For the remainder, they are identical, except for their (ontology, conceptual, formal, and implementation) models. In short, the heterogeneity between our agents is described in Table 4 (steps marked with a Ξ are assumed to cause heterogeneity):

6 Ontology Clusters

Ontology clustering is based on the similarities between the concepts known to the different agents. Since in this application all agents are assumed to be familiar with concepts such as coffee beans, water, and kitchen appliances, we

Step	Action	Result	Examples
A1	Perceive	Perception	An image
A2	Concept Formation	Concepts	An old lady, A young lady
A3	Choose Natural Language to represent the Conceptualisation	Natural Language	English, Dutch, French, Italian
A4	Select a subset of the Natural Language to Refer to the Concepts	Named concepts	{“young lady”, “old woman”}, {“oude mevrouw”, “jonge mevrouw”}, {“jeune fille”, “vieille dame”}
A5	Describe Meaning of Concepts	Informally defined concepts	“A young lady is a female person with an age < 20”, “A young lady is a female person with an age < 30”
A6	Choose Ontology Language	Ontology Language	CLASSIC, LOOM, ONTOLINGUA, Daplex
A7	Formalise Ontology	Ontology	“define ontology my_ontology”
B1	Choose Conceptual Language	Conceptual Language	CML, ER-diagrams, Flow-charts
B2	Conceptual Modelling	Conceptual Model	...
B3	Choose Formal Language	Formal Language	FOPL, KIF, ML ²
B4	Formal Modelling	Formal Model	...
C1	Identify OS and Platform	OS / Platform	Unix, Windows, Linux, ...
C2	Choose Implementation Language	Implementation Language	C++, Java, PROLOG, Access, P/FDM
C3	Implement Application	Application	Database, Knowledge System
D1	Choose Agent Communication Language	Agent Communication Language	KQML
D2	Choose Content Language	Content Language	KIF, PROLOG
D3	Choose the Ontology to perform the communication	Communication Ontology	“define ontology communication_ontology...”
D4	Choose the Design of the Agent Interface of Resource	Design of the agent version of Resource	...
D5	Choose the Implementation of the Agent Interface of Resource	Agent version of Resource	...

Table 3: Design process steps to identify heterogeneity kinds

A1	A2	A3	A4	A5	A6	A7	B1	B2	B3	B4	C1	C2	C3	D1	D2	D3	D4	D5
	⊕	⊕	⊕	⊕		⊕		⊕		⊕			⊕			⊕	⊕	⊕

Table 4: Design process steps to identify heterogeneity kinds

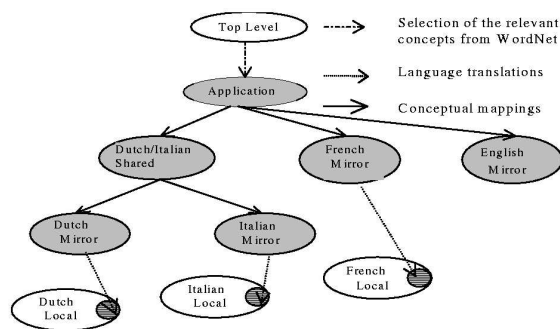


Figure 2: The hierarchy of multiple shared ontologies

group these concepts in a so-called application (specific) ontology, rooted at the top of the hierarchy of ontologies. The ontology on top of the hierarchy describes the specific domain and so it is not reusable. For this reason, and following Van Heijst approach [Van97] it was named application ontology.

The concept definitions in this application ontology are derived from an existing top-level ontology, which is here chosen to be WordNet [Mil90].

The application ontology contains a relevant subset of WordNet concepts. For each concept a sense is selected, depending on the domain, from those provided by WordNet. If some agents share concepts that are not shared by other agents then there is a reason to create a new ontology cluster. A new ontology cluster here is a child ontology that defines certain new concepts using the concepts already contained in its parent ontology. The Italian and Dutch agent, for instance, share the concept of a "coffee-maker device" that has a water container a filter that also holds coffee and a coffee container. This concept is unknown to the French and English agents. Ultimately, the agents are likely to have concepts that are not shared with any other agent. In our ontology structure, we then create a separate, agent-specific ontology as sub ontology of the cluster in which the agent resides. We refer to these ontologies as mirror ontologies since they mirror the local agent ontologies. The mirror ontologies are the leaf nodes of our ontology hierarchy. Since the local ontologies are expressed in the agent's mother tongue the language heterogeneity A3 and A4 occurs between the local and the mirror ontologies. To overcome this kind of heterogeneity the local ontologies are translated in one common language, here English. In figure 2 the ontology hierarchy together with mappings between local and mirror ontologies are presented.

In each of the ontologies in the structure concepts are described in terms of attributes and inheritance relations holding in the ontology's structure. Concepts are hierarchically organised and the inheritance allows the passing down of information through the hierarchy. Concepts are expressed in terms of *inherited* and *distinguishing* attributes. Inherited attributes are those express-

ing the similarities between a parent concept and its siblings (the parent concept can be defined in the ontology itself or in a parent ontology). They describe the main characteristics of a concept that are also present in its sub-concepts. A concept that specialises a more general one inherits all the attributes from its parent concept. In these translations we exclude multiple inheritance, chiefly since the language that has been chosen to define our ontology (P/FDM - [Emb92]) does not support it.

To the set of inherited attributes other attributes are added to distinguish the specific concept from the more general one. These attributes describe the characteristic differences between a concept and its siblings. The distinguishing attributes are used to map concepts from a source ontology into a destination ontology preserving the meaning of the concept.

The following example shows a concept expressed in terms of its attributes, both inherited and distinguishing. For purpose of explanation in the remainder all the examples of ontologies will be expressed in ONTOLINGUA ([Gru92, Ontol]) rather than P/FDM although we still disregard multiple inheritance.

In the coffee domain introduced in section 4 the Dutch local ontology refers to a parent shared ontology. The Dutch local ontology contains the specialised concept of Filter and coffee holder constituent which is a component of an electric coffee maker:

```
(Define-Class Filter-And-Coffee-Holder-
Constituent (?X)
"A filter and coffee holder con-
stituent is a kind of con-
stituent of an electric coffee maker that
holds coffee and uses a paper filter."
:Def (And (Constituent ?X)))
```

with attributes described by the following slots:

```
(Define-Relation Holds-In (?Frame ?Value)
"A filter and coffee holder con-
stituent holds in it ground coffee "
:Def (And (Filter-And-Coffee-Holder-
Constituent ?Frame) (Ground-
Coffee ?Value)))
```

```
(Define-Relation Uses-Paper-
Filter (?Frame ?Value)
"A filter and coffee holder con-
stituent uses a paper filter"
:Def (And (Filter-And-Coffee-Holder-
Constituent ?Frame)(Paper-Filter ?Value)))
```

This concept is declared as subclass of constituent that in the Dutch local ontology is defined as:

```
(Define-Class Constituent (?X)
"A constituent is a kind of physical ob-
ject that is part of a kitchen appliance."
:Def (And (Physical-Object ?X)))
```

with the attribute:

```
(Define-Relation Is-Part-Of (?Frame ?Value)
"A con-
stituent is part of a kitchen appliance"
:Def (And (Constituent ?Frame) (Kitchen-
Appliance ?Value)))
```

Thus, the concept filter and coffee holder constituent inherits from the parent concept constituent the features of being part of a kitchen appliance and of being a physical object. Moreover it adds to these the characteristic attributes that allow to distinguish between a generic constituent and a filter and coffee holder constituent.

7 Communication between resources

In the ontology structure presented in section 6 communication between resources is performed via mapping functions (section 4). In this experiment mappings can be either partial or total and are not necessarily isomorphic; that is if a mapping function exists from a resource A to a resource B this does not imply that the opposite mapping from the resource B to the resource A exists.

The remainder of this section outlines how we envisage that communication between the resources in the ontology structure is performed. Two kinds of translations between ontologies are distinguished:

1. Translations of the first type are those mapping concepts from the agent's local ontology onto a corresponding concept within the 'mirror' ontology. This is a language translation and will largely imply a direct word-by-word mapping although common language-translation problems occur here (e.g. [Mah95]). This first step resolves the heterogeneity kind A3 (see table 2 above).
2. Translations of the second type are those that will be encoded in functions mapping concepts between the ontologies composing the structure, thus translating concepts from one ontology, possibly repeatedly, into its child or into its parent ontology. The aim of this step is to resolve the other types of heterogeneity. Concepts belonging to one of the 'mirror' ontologies are mapped into concepts of another 'mirror' ontology via one or more shared ontologies. The remainder of this section will focus on this type of translations.

In the reminder we will use the term source ontology to denote the ontology containing the concept that is to be translated, whereas we use the term destination ontology to denote the ontology the concept has to be translated to.

The ontologies in the structure are hierarchically organised, and for this reason translating from the source ontology into the destination ontology may generally consist of two types of translation steps. The first type of is generalisation (from the concept to its hypernym in the same or in

a parent shared ontology). The second type is specialisation (from the concept in the parent shared ontology to its hyponym in the same or in another ontology). However, the mere translation of a concept through a generalisation and a subsequent specialisation is not enough; indeed such a translation is guaranteed to preserve the meaning only if the concept to translate has a synonym in the local destination ontology. If this is not the case the concept will be mapped into a more general one, and thus it will be an approximation. This is what happens in the SIMS project [Are96] where a query is reformulated as the union of its more general concepts using the relationship holding between a class of concepts and its super-class. To preserve the meaning, however, some constraints can be added.

The translation between local ontologies can be summarised by the following steps:

- a) The concept needing to be translated is identified.
- b) Once identified, the concept is translated into the terms of the shared ontology immediately above the source ontology. If a direct translation does not exist the first hypernym of the concept is found such that a translation exists between the hypernym and a concept in the shared ontology immediately above. The same translation process is applied to all the concepts in the destination ontology.
- c) The hypernym of the concept is then located in the shared ontology.
- d) The attributes of the concept in the source ontology are compared with the attributes of the hypernym just found to select the distinguishing features;
- e) Then the concept expressed in terms of the shared ontology, (that is the relationships holding between concepts in the structure are identified) together with its distinguishing attributes is passed to the parent shared ontology;
- f) If in the destination local ontology there is a concept that is a specialisation of the one passed to the shared ontology, then for this local concept a mapping can be defined between the original local concept and the one just selected. If not the procedure is recursively applied, climbing up a level to the more general shared ontology.

This kind of translation obtained by subsequent generalisation and specialisation steps is effective only if the source and the destination concepts have a common ancestor that is not too high in the hierarchy, otherwise the generalisation steps can lead to a too general ancestor. In this latter case, the information loss due to the generalisation is too high, and the translation obtained might be a trivial one. To avoid the loss of information that is intrinsic of a generalisation, attributes and relations linking concepts play a

crucial role. In fact they not only allow the identification of the hypernym of a concept (either in the same or in a shared ontology) but they also allow to "attach" some characterising information to each concept thus giving a distinction between the concept itself and its parent.

The following example illustrates what kind of information is passed when translating a concept from one ontology to another. The example refers to the natural-language dialogue that is presented in section 4 where the French agent, François, tries to explain to the Italian agent, Nicola, what a cafetière is. The concept of Cafetière is represented in the French mirror ontology. François wants to explain it to Nicola, who knows the concept of Espresso coffee maker (defined in the Italian ontology). As stated before, we focus on ontology conversions and disregard any language translations. Hence, we do not address the translation from the local ontologies into the mirror ontologies. Here we assume that the source concept and those in the destination ontology are already expressed in terms of their respective shared ontologies, as in b) above.

The French (mirror) ontology describes a cafetière as:

```
(Define-Class Cafetiere (?X)
"A cafetire is a kind of kitchen appli-
ance that produces French cof-
fee and has a filter device and a
jug."
:Def (And (Coffee-Maker ?X)))
```

so as a subclass of Coffee Maker (subclass of Kitchen Appliance in the Application ontology) with attributes *Has filter device*, *Has jug*, and *Produces*, which is inherited by Kitchen Appliance:

```
(Define-Relation Has-Filter-
Device (?Frame ?Value)
"A cafetiere has a filter device"
:Def (And (Cafetiere ?Frame) (Filter-
Device ?Value)))

(Define-Relation Has-Jug (?Frame ?Value)
"A cafetiere has a jug"
:Def (And (Cafetiere ?Frame) (Jug ?Value)))

(Define-Relation Produces (?Frame ?Value)
"A kitchen appliance is used to pro-
duce some substance"
:Def (And (Kitchen-Appliance ?Frame) (Sub-
stance ?Value)))
```

where a jug is defined as a substance container, because it can contain both a liquid and a solid substance. The espresso coffee maker is defined by the following expression:

```
(Define-Class Espresso-Coffee-Maker (?X)
"An espresso cof-
fee maker is a kind of tool that pro-
duces espresso coffee and has a wa-
ter reservoir,
a filter that also holds coffee and a cof-
fee reservoir."
```

```
:Def (And (Coffee-Maker-Device ?X)))
```

and slots:

```
(Define-Relation Produces (?Frame ?Value)
"An espresso coffee maker pro-
duces espresso coffee"
:Def (And (Espresso-Coffee-
Maker ?Frame) (Espresso ?Value)))
```

```
(Define-Relation Has-Coffee-
Reservoir (?Frame ?Value)
"An espresso coffee maker has a cof-
fee reservoir"
:Def (And (Espresso-Coffee-
Maker ?Frame)(Coffee-Reservoir ?Value)))
```

```
Define-Relation Has-Filter-And-Coffee-
Holder (?Frame ?Value)
"An espresso coffee maker has a fil-
ter and coffee holder"
:Def (And (Espresso-Coffee-
Maker ?Frame) (Filter-And-Coffee-
Holder ?Value)))
```

```
Define-Relation Has-Water-
Reservoir (?Frame ?Value)
"An espresso coffee maker has a wa-
ter reservoir"
:Def (And (Espresso-Coffee-
Maker ?Frame) (Water-Reservoir ?Value)))
```

As there is no direct resemblance between the definition of the two concepts it is necessary to generalise the concept of a cafetire (as is done in the motivating scenario). The French (mirror) ontology does not belong to the same cluster as the Italian ontology. This latter ontology has as parent ontology the Dutch/Italian shared. In this shared ontology a concept Coffee maker device is defined as a kind of Coffee maker, concept (the latter which is defined in the application ontology). The coffee maker device is defined as:

```
(Define-Class Coffee-Maker-Device (?X)
"A coffee maker device is a kind of cof-
fee maker. It produces cof-
fee drinks and has a water container,
a filter that also holds coffee and a cof-
fee container."
:Def (And (Coffee-Maker ?X)))
```

```
(Define-Relation Has-Filter-Holding-
Coffee (?Frame ?Value)
"A coffee maker device has a fil-
ter also holding coffee."
:Def (And (Coffee-Maker-
Device ?Frame) (Filters-And-Holds-
Coffee ?Value)))
```

```
(Define-Relation Has-Water-
Container (?Frame ?Value)
```

```
"A coffee maker device has a wa-
ter container"
:Def (And (Coffee-Maker-
Device ?Frame) (Water-Container ?Value)))

(Define-Relation Has-Coffee-
Container (?Frame ?Value)
"A coffee maker device has a cof-
fee container"
:Def (And (Coffee-Maker-
Device ?Frame) (Coffee-Container ?Value)))

(Define-Relation Produces-
Drink (?Frame ?Value)
"A coffee maker device pro-
duces drink coffee"
:Def (And (Coffee-Maker-
Device ?Frame) (Coffee ?Value)))
```

At this stage the relationships between concepts and their attributes in the ontology clusters are taken into account. As mentioned in section 3 attributes are considered hierarchically organised, therefore the following relations hold in the example above. The ontology where a concept is defined is indicated in parentheses:

- Espresso coffee maker (Italian mirror ontology) specialises the concept of Coffee maker device (Dutch/Italian shared ontology);
- The attribute *Has water reservoir* in Espresso coffee maker specialises the attribute *Has water container* in Coffee maker device;
- The attribute *Has filter and coffee holder* in Espresso coffee maker specialises the attribute *Has filter holding coffee* in Coffee maker device;
- The attribute *Has coffee reservoir* in Espresso coffee maker specialises the attribute *Has coffee container* in Coffee maker device.

In this case all these attributes are inherited and there is no distinguishing attribute.

In order to access an ontology that is shared both by the Italian and by the French agent, that is the application ontology itself it is necessary to climb a step further in the ontology hierarchy.

In the application ontology the concept coffee maker is defined as a direct hypernym for the concept cafetire and is also hypernym of the concept espresso coffee maker through the intermediate concept coffee maker device:

```
(Define-Class Coffee-Maker (?X)
"A coffee maker is a kind of kitchen appli-
ance. It produces coffee
and has a liquid container, a filter com-
ponent and a coffee holder."
:Def (And (Kitchen-Appliance ?X)))
```

with attributes:

```
(Define-Relation Has-Coffee-
Holder (?Frame ?Value)
"A coffee maker has a component hold-
ing coffee"
:Def (And (Coffee-Maker ?Frame) (Substance-
Container ?Value)))

(Define-Relation Has-Filter-
Component (?Frame ?Value)
"A coffee maker has a component filtering"
:Def (And (Coffee-Maker ?Frame) (Filter-
Component ?Value)))

(Define-Relation Has-Liquid-
Container (?Frame ?Value)
"A coffee maker has a component contain-
ing liquids"
:Def (And (Coffee-Maker ?Frame) (Substance-
Container ?Value)))

(Define-Relation Is-Used-To-
Produce (?Frame ?Value)
"A coffee maker has a component contain-
ing liquids"
:Def (And (Kitchen-Appliance ?Frame) (Sub-
stance ?Value)))
```

where the *Filter component* is defined as a *Component* and the attribute *Is used to produce* is inherited by Kitchen appliance.

To explain the cafetière concept to Nicola, the Italian agent, it will be necessary to map this concept into a concept Nicola is familiar with, that is the concept coffee maker in the application ontology. This is obtained by relating some of the attributes of cafetière as inherited from the attributes of the concept coffee maker in the application ontology. The remaining attributes of cafetière are the distinguishing attributes that will also be added to the mapping function. In this way the relationships relating the concept of cafetière to the concept of coffee maker can be described as:

- Cafetière (French mirror ontology) specialises the concept of Coffee maker (Application ontology)
- The attribute *Has filter device* in Cafetière specialises the attribute *Has filter component* in Coffee maker
- The attribute *Has jug component* in Cafetière specialises the attribute *Substance container* and is related to the attributes *Has liquid container* and *Has coffee holder*.

This latter point deserves further attention. The attribute *Has Jug* is defined of type Jug in the French mirror ontology. The ONTOLINGUA definition for Jug is:

```
(Define-Class Jug (?X)
```

```

"A jug is a kind of substance con-
tainer. It is a component of a
cafetire that can both contain a liq-
uid and hold a ground coffee
powder"
:Def (And (Substance-Container ?X))

(Define-Relation Composes (?Frame ?Value)
"A jug is a component of a cafetiere"
:Def (And (Jug ?Frame) (Cafetiere ?Value)))

(Define-Relation Contains (?Frame ?Value)
"A jug contains some liquid"
:Def (And (Jug ?Frame) (Liquid ?Value)))

(Define-Relation Holds-
Inside (?Frame ?Value)
"A jug holds inside ground coffee"
:Def (And (Jug ?Frame) (Ground-Coffee-
Powder ?Value)))

```

In this case the Jug is a kind of Substance container (French mirror ontology) that specialises the homonymous concept of Substance container defined in Application ontology. Both the attributes *Has liquid container* and *Has coffee holder* for the concept Coffee maker (Application ontology) are defined as kind of Substance container. Therefore Jug, in turns, specialises Substance container (Application ontology) adding the distinguishing features that a jug composes a Cafetière and that both contains a liquid substance and holds the coffee powder. Jug is related to *Has liquid container* and *Has coffee holder* as those concepts specialise the hypernym for Jug (Substance container) in the Application ontology.

Once related the cafetière to the coffee maker the information that a cafetière is a kind of coffee maker and is characterised by a set of distinguishing attribute is passed down to the Italian ontology. In this way, Nicola will be able to approximate the concept of a cafetière by relating it to a concept, Coffee maker, that he knows.

8 Conclusion

In this article we reported on a small experiment in the integration of heterogeneous information sources. To position our experiment in the vast amount of heterogeneous-resource-integration literature we commenced by defining the term heterogeneity more precisely. A total of seventeen kinds of resource heterogeneity was distinguished based on the steps taken in an imaginary system-engineering process. In these, seven different kinds of "language heterogeneity", and eight different kinds of "ontology heterogeneity" were distinguished. It is interesting to note is that three heterogeneity kinds fall arguably in both of these categories.

The aim of our experiment is to investigate the feasibility of using a set of related ontologies rather than one over-

arching ontology or several independent ontologies. We discussed a proposal for an agent architecture with a hierarchical ordering of ontologies. Ontologies lower in the structure contain more refined concepts than the ontologies higher in the structure and since different branches of ontologies may extend on their concepts in different ways, the structure allows heterogeneous ontologies. The coffee-preparing domain is attractive as it serves to illustrate that different communities may share knowledge at different abstraction levels. Since all communities share the 'coffee basics', there will always be a way to explain unknown concepts in known terms, albeit that this may cause loss some of information.

Although the idea of using abstract and more refined ontologies is not a novelty, the idea to use a structured set of heterogeneous ontologies simultaneously in a distributed architecture has not received much attention. In such architectures we hope to combine the advantages of having abstract ontologies (general applicability) and refined ontologies (more meaningful communication). Unfortunately, we also inherit some disadvantages. One important disadvantage of ontology structures such as the one proposed is that translations are required between the ontologies in the structure. In the article we have shown the role of inherited and distinguishing attributes in such translations. We think the disadvantage can be outweighed by the benefits of having a more flexible and maintainable way of dealing with communication standards. Ongoing experiments will focus on the evaluation of the translations obtained with such approach, and on extending the approach in the case of real life applications with several definitions. These experiments aim at giving us more insights regarding the circumstances under which advantages and disadvantages take manifest.

This research is partly conducted as a PhD project of the second author who will continue to explore the possibilities of ontology structures and their implications on agent communication.

8.0.1 Acknowledgements

This research is in part funded by BT Research Laboratories in the United Kingdom. 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/~pepijn/kraft.html> and <http://www.csd.abdn.ac.uk/research/kraft.html>.

The authors wish to thank Floriana Grasso for her invaluable contribution and would like to express their gratitude to Mike Shave, Trevor Bench-Capon, Ian Finch and the remaining colleagues of the KRAFT project.

References

- [All90] R. E. Allen *The Concise Oxford Dictionary of Current English*. Clarendon Press, Oxford, 1990.
- [Are96] Y. Arens, C. Hsu, and C. A. Knoblock Query processing in the SIMS Information Mediator. *Advanced Planning Technology*, Austin Tate (Ed.), AAAI Press, Menlo Park, CA, 1996.
- [Bay97] R. J. 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. *ACM SIGMOD Record Vol. 26, No. 2 (June 1997), SIGMOD '97. Proceedings ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, USA, pp. 195-206, 1997.
- [Cer93] S. Ceri, and J. Widom Managing Semantic Heterogeneity with Production Rules and Persistent Queues. *Proceedings of the 19th VLDB Conference*, Dublin, Ireland, pp.108-119, 1993.
- [Dar95] A. Daruwala, C. H. Goh, S. Hofmeister, K. Hussein, S.E. Madnick, M. Siegel The Context Interchange Network Prototype, Database Applications Semantics. *Proceedings of the Sixth IFIP TC-2 Working Conference on Data Semantics (DS-6)*, R. Meersman, and L. Mark, Eds., Stone Mountain, Atlanta, Georgia, USA, Chapman & Hall, London, pp. 65-92, 1995.
- [Emb92] S.M. Embury, Z. Jiao and P. M. D. Gray Using Prolog to Provide Access to Metadata in an Object-Oriented Database. *Proceedings of the International Conference on the Practical Application of Prolog*, 1992.
- [Fin98] T. Finin, Y. Labrou, and Y. Peng Mobile Agents can benefit from Standards Efforts in Inter-agent Communication. *IEEE Communications Magazine, special issue on Mobile Software Agents for Telecommunications*, Vol. 36, Nr. 7, 1998.
- [Fri97] N. Fridman Noy, and C. D. Hafner The State of the Art in Ontology Design: A Survey and Comparative Review. *AI magazine*, Vol. 18, Nr. 3, pp. 53-74, 1997.
- [Gra97] P. D. M. 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. *Proceedings of Database and Expert System Applications Conference (DEXA' 97)*, Toulouse, France, 1997.
- [Gra98] P. D. M. Gray, Z. Cui, S. M. Embury, W. A. Gray, K. Hui, A. Preece An Agent-Based System for Handling Distributed Design Constraints. *Workshop on Agent-Based Manufacturing at Agents'98 International Conference*, Minneapolis, USA, 1998.
- [Gru92] T. R. Gruber A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, Vol. 2 Nr. 5, pp. 199-220, 1992.
- [Kim91] W. Kim, and J. Seo, J. Classifying Schematic and Data Heterogeneity in Multidatabase Systems. *IEEE Computer*, Vol. 24, pp. 12-18, December 1991.
- [Lab96] Y. Labrou, Y. *Semantics for an Agent Communication Language*. PhD Dissertation Thesis. University of Maryland, Baltimore County, USA, 1996.
- [Mah95] K. Mahesh, and S. Nirenburg Semantic Classification for Practical Natural Language Processing. *Proceedings of the Sixth ASIS SIG/CR Classification Research Workshop: An Interdisciplinary Meeting*, Chicago IL, USA, 1995.
- [Mar90] S. T. March *Special Issue on Heterogeneous Databases*. ACM Computing Surveys, ACM Press, Vol. 22, No. 3, 1990.
- [Mil90] G. A. Miller Nouns in WordNet: a lexical inheritance system. *International Journal of Lexicography*, Vol. 3, Nr. 4, pp. 245-264, 1990.
- [Men96] E. Mena, V. Kashyap, A. Shet, A., and A. Illarramendi OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. *Proceedings First IFCIS International Conference on Cooperative Information Systems (CoopIS '96)*, Brussels, Belgium, IEEE Computer Society Press, pp. 14-25, 1996.
- [Ontol] Ontolingua Reference Manual. <http://www-ksl.stanford.edu/people/brauch/demo/reference-manual/index.html>.
- [Sha97] M. J. R. Shave Ontological Structures for Knowledge Sharing. *The New Review of Information Networking*, Vol 3, pp. 125-134, 1997.

- [Tam98] V. A. M. Tamma, and P. R. S. Visser Integration of Heterogeneous Sources: Towards a Framework for Comparing Technique. *Proceedings of the AI*IA '98 Workshop on Techniques for Organisation and Intelligent Access of Heterogeneous Information*, Padova, Italy, pp. 89-93, 1998.
- [Usc98] M. Uschold, P. Clark, M. Healy, K. Williamson, and S. Woods An Experiment in Ontology Reuse. *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling, and Management*, (KAW'98), 1998.
- [Van97] G. Van Heijst, T. Schreiber, and B. Wielinga Using Explicit Ontologies in KBS. *International Journal of Human-Computer Studies*, Vol. 46, Nr. 2/3, pp. 183-292, 1997.
- [Vi98a] P. R. S. Visser, D. M. Jones, T. J. M. Bench-Capon, and M. J. R. Shave. Assessing Heterogeneity by Classifying Ontology Mismatches. *Proceedings Conference on Formal Ontology (FOIS'98)*, N. Guarino , Ed., Trento, Italy, pp. 148-162. An earlier version of this article appeared at the *AAAI 1997 Spring Symposium on Ontological Engineering*, Stanford, CA, USA, 1996.
- [Vi98b] P. R. S. Visser, and Z. Cui On Accepting Heterogeneous Ontologies in Distributed Architectures. *Proceedings of the ECAI'98 workshop on Applications of Ontologies and Problem-solving methods*, Brighton, UK, 1998.
- [Woe92] D. Woelk, W. Shen, M. Huhns, and P. Canata Model Driven Enterprise Information Management in Carnot *Enterprise Integration Modeling: Proceedings of the First International Conference*, C. J. Petrie Jr., Ed., MIT Press, Cambridge, MA, 1992.