An Ontological Approach to Federated Data Integration

Agustina Buccella and Alejandra Cechich
Departamento de Ciencias de la Computación, Universidad Nacional del Comahue,
Buenos Aires 1400, Neuquén, Argentina
Email: abuccel, acechich@uncoma.edu.ar

Nieves R. Brisaboa
Departamento de Computación, Universidade de A. Coruña,
Campus de Elviña s/n, 15071 – A. Coruña, España
Email:brisaboa@udc.es

Abstract. During the last years a lot of projects and lines of research have emerged from different proposals trying to find the best way to reach data integration. Two powerful techniques have appeared separately – ontology and contextual information – in order to help solve semantic heterogeneity problems. In our proposal we combine both techniques exploiting the advantages of each of them. We propose a new approach, in which three main components work together in order to achieve a consistent integration. Each component contains some type of semantic information modeled by ontologies and contexts. Our approach helps the building of each of the components and address other types of heterogeneity such as ontological heterogeneity.

Keywords: Data Integration, Ontology, Semantic Heterogeneity, Context, Federated Databases, Federated Systems.

1 Introduction

Terminologically, *Federated Systems* and *Federated Databases* are used to provide an integrated data access, which has a number of physically distributed, heterogeneous and autonomous information sources (databases) [8]. Here *autonomy* means a direct access to the data through the federated system or users' local interfaces. In this sense, it is important that the federation does not damage the performance of the local system. On the other hand, *distribution* is intimately connected to the concept of Internet, and thus to information located in different geographic places. Lastly, in our case, *heterogeneity* will be on ontology terms: there is an *ontology heterogeneity* problem if two systems make different ontological assumptions about their domain knowledge. Besides, within a Federated System the information sources integrate any type of information systems, such as HTML pages, databases, filing, etc. either static or dynamic. On the latter case, some mechanisms should be created so as to know which information is available at a given moment.

Data integration is the main problem we must face within a Federated System. To get a consistent integration, a series of decisions should be made in a correct way. When users query a Federated System, they should get a suitable semantic answer. Herein, the semantic heterogeneity makes this task difficult because of its bearing problems on synonymous, generalization/specialization, etc.

Our proposal is based on the use of ontologies and contexts, which provide a higher degree of semantics to read and combine information sources. On the hand of the ontologies, they provide the nouns and descriptions of the domain specific entities by using predicates. These predicates represent the relationships between the entities. An ontology provides a vocabulary to represent and

communicate knowledge about the domain, and also a number of relationships containing the vocabulary terms at a conceptual level. On the other hand, contexts are useful tools to model concepts which are in conflict with one another. Concepts are true or false according to the context it is in.

In our work, we define one ontology for each information source. These ontologies, called *source ontologies,* converge in a *shared vocabulary* (or global ontology). Hence, our proposal is based on a hybrid ontology approach [21], where a specific context or domain is defined for each source ontology. Also, within these ontologies, each concept can involve one or more contexts. Therefore, a component must be included to deal with the information flow between the source ontologies and the shared vocabulary. We called this component *Ontology and Context Mapping* (OCM) [2].

An *ontological heterogeneity* problem [20] appears when the mapping between the source ontologies and the shared vocabularies must be done. The ontological heterogeneity has a series of inherent problems because each ontology corresponds only to one information source created independently. There are two basic types of ontology mismatches: *conceptualization mismatches* and *explication mismatches*. The former, *conceptualization mismatches*, may appear in two or more conceptualizations of a domain. These conceptualizations differ in the ontological concepts or in the way these concepts are related The latter, *explication mismatches*, are not defined on the conceptualization of the domain but on the way the conceptualization is specified. The definitions are considered as 3-tuples *Def* = <*T,D,*C> in which *T* is the definiendum, *D* is the definiens, and *C* is the ontology-concept description to be defined (distinguished during the conceptualization process). An example of a definition is the concept description "A vessel is taken to be something large and seagoing" that is explicated as vessel(X) _ seagoing(X) _ large(X), in which vessel(X) is the definiendum *(*T), and seagoing(X) _ large(X) is the definiens *(*D). Another example is, one ontology contains the definition *car(X) black(X) large(X)* to define the concept of a *car* and the other ontology contains the definition *shark(X) black(X) large(X)* to define the concept of a *shark.*

We find several research works using a hybrid ontology approach and contexts in literature. One example is the framework of the COIN project [6,18] that uses a logic and formal specification of its components. This framework is made up of three components: the *domain model*, the *elevation axioms* and the *context axioms*. Each information source contains one set of elevation axioms and one set of context axioms. Both converge in a unique domain model. Another example is presented in [16], in which the context is defined based on operations needed by the user to describe the application domain. A last example is the OBSERVER system [10,11], which although without contexts, uses local ontologies (built up independently) for each information source.

In this paper we define an approach containing some aspects of the systems mentioned above with an extra combination of ontologies and contexts. In section 2, essential components of our proposal and their specific functions are described. Then, section 3 explains our approach especially created to help build these components. Conclusions and future work are addressed in the last section.

2 A new approach for data integration

The main task of a Federated System is retrieving optimal and consistent information, and returning this information to the users. In order to do that, data within the available information source must be selected generating an answer that should be useful for the users.

Figure 1 shows the main components of our proposal. For brevity reasons, in this paper we will only focus on describing the federation layer. We assume the *wrapper layer* involves a number of modules belonging to a specific data organization. These modules know how to retrieve data from the underlying sources hiding those data organizations. There is a communication between the

source ontologies and the modules since both interact to retrieve the required information. As the federated system is autonomous, local users access their local databases through the *access layer* independently from users of other systems. Otherwise, they need to use the *user interface layer* to access the federated system.

The model in Figure 1 is based on the work presented in [3], in which our extension adds the federation layer.

2.1 The Federation Layer

The *federation layer* (FL) is the main component of our proposal. This layer should solve the problems related to the ontological heterogeneity. In order to do that, the federation layer is made up of three components: the source ontologies, the shared vocabulary and the OCM.

As we have previously mentioned, there is one source ontology for each information source. Also a specific context (or domain) is defined for each source ontology. Thereby, we use the tuple (ontology, context) to define each ontology:

$$FL \leftarrow (O_1, C_1) \wedge (O_2, C_2) \wedge \ldots \wedge (O_n, C_n)$$ for n equal to the number of source ontologies (and information sources) within the system.

Besides, each ontology might be related to several contexts indicating the different roles of one database. For example, the *use cases* of a UML specification [5] might be the source to obtain some of the contexts. Each context contains a series of concepts included in the ontology, and the specific context (or domain) is made up of all of these concepts. Then, the set of contexts associated to the ontologies can be written as:

$$(O_1, C_1) = \{ c_{11}, c_{12}, c_{13}, \ldots, c_{1m}\} \text{ for } m \geq 0.$$
.....
$$(O_n, C_n) = \{ c_{21}, c_{22}, c_{23}, \ldots, c_{nm}\} \text{ for } m \geq 0.$$

We represent each ontology as follows:

$$(O_1, C_1) = \{ concept_1, concept_2, \ldots, concept_n\}$$ for n equal to the number of concepts in the ontology (1)
$$(O_1, C_1) = \{ c_{11}, c_{12}, c_{13}, \ldots, c_{1m}\} \text{ for } m \geq 0.$$

$$c_{11} = \{ concept_1, concept_2, concept_5, concept_{10}\}$$
.....
$$c1_m = \{ concept_1, concept_3, concept_5, concept_{12}\}$$ for m equal to the whole number of contexts.

The concepts in (1) might be relationships, classes, subclass/subrelationship relationships, class/instance relationships and hierarchy relationships [7].

The OCM component deals with the relationships among the contexts of the different source ontologies. These relationships are equality, inclusion, intersection, etc. For instance, the equality relationship means that contexts in one ontology are the same as contexts in another ontology. As another example, the union (join) of two contexts may be included in another context and vice versa. Some generic examples are:

$$< O_n, C_n(C_{nm}) > = < O_k, C_k(C_{kl}) >$$

$< O_n, C_n(C_{nm}) > \cap < O_n, C_n(C_{np}) > = < O_k, C_k(C_{ko}) >$
$< O_n, C_n(C_{nm}) > \subset < O_k, C_k(C_{kl}) > \cup < O_k, C_k(C_{ko}) >$

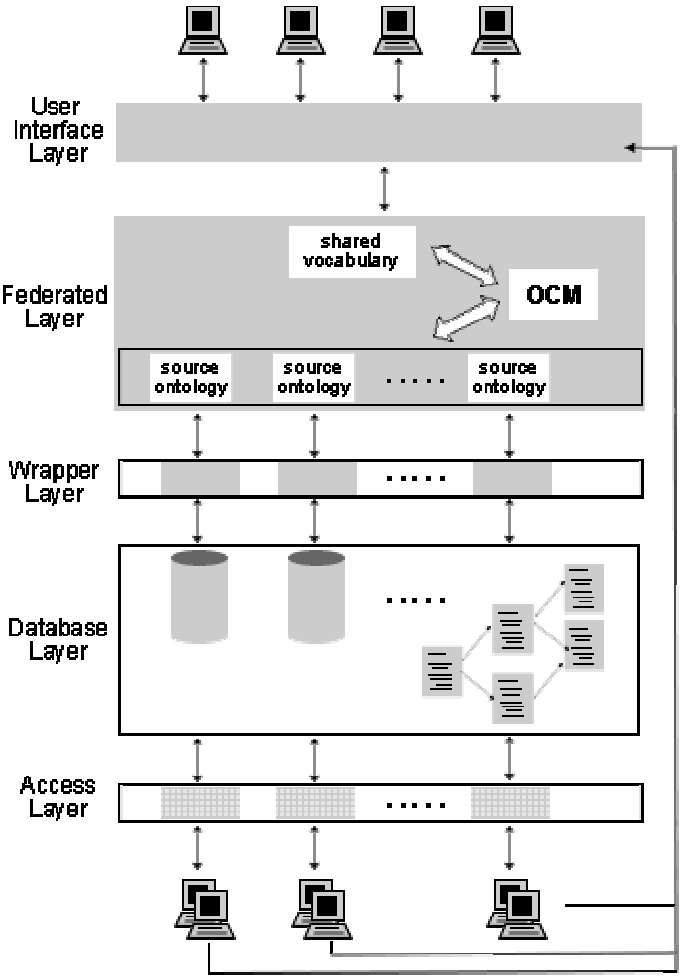

Figure 1. Federated System

These types of relationships help to create a component called *shared vocabulary*. It includes the generic concepts involving the source ontologies together with the resulting contexts. The OCM component is responsible of providing these elements. Also, the OCM component will have the *equality axioms* describing the similarity relationships between concepts of two or more contexts, which are connected by some of the relationship described above. We use the functions defined in [15,16] in order to find the similarity values within these related contexts. The (2) and (3) functions show the formulas where *a* and *b* are concepts of two ontologies (O_1 y O_2 respectively).

$$S(a^{O_1}, b^{O_2}) = w_p.S_p(a^{O_1}, b^{O_2}) + w_f.S_f(a^{O_1}, b^{O_2}) + w_a S_a(a^{O_1}, b^{O_2}) \quad for \quad w_p, w_f, w_a \geq 0 \quad and \quad w_p + w_f + w_a = 1 \tag{2}$$

$$S(a,b) = \frac{|A \cap B|}{|A \cap B| + \alpha(a,b)|A/B| + (1-\alpha(a,b))|B/A|} \quad for \quad 0 \leq \alpha \leq 1 \tag{3}$$

The function (2) is a sum of products (value times weight (*w*)) where w represents the parts, the functions and the attributes (w_p, w_f, y w_a respectively). This model is called *feature matching*, where *parts* are structural elements of a concept (or class), such as "roof" and "floor" of a building; *functions* represent the purpose of the concept; and *attributes* correspond to additional characteristics of a concept. Consequently, the relationships between the contexts define values to these weights.

The function (3) is based on the Tversky's model [19] where *A* and *B* correspond to description sets of *a* and *b* (i.e., synonym sets, sets of distinguishing features, etc). The equality axioms are built with the result of these functions for the significant cases, that is for high similarity values.

Finally, the shared vocabulary involves a series of generic concepts and contexts built using the equality axioms. Users use the vocabulary to query and get answers through the user interface layer. Once the user chooses the context and makes the query, the system will use the OCM component to know which concepts are related with. Thereby, the system gets access to the information sources to produce the data.

3 An Ontology Construction Approach

In this section we will describe our approach for building the federation layer's components. Figure 2 shows the algorithm designed to do so. The method has three main stages: *build source ontologies*, *build the mappings among the source ontologies (the OCM component)* and *build the shared vocabulary*. We will briefly explain each stage by using an example. Several languages may be used to represent an ontology – CLASSIC [1], LOOM [9], Ontolingua [7], etc. Here, we have used Ontolingua because of its expressiveness to describe the elements of the ontology.

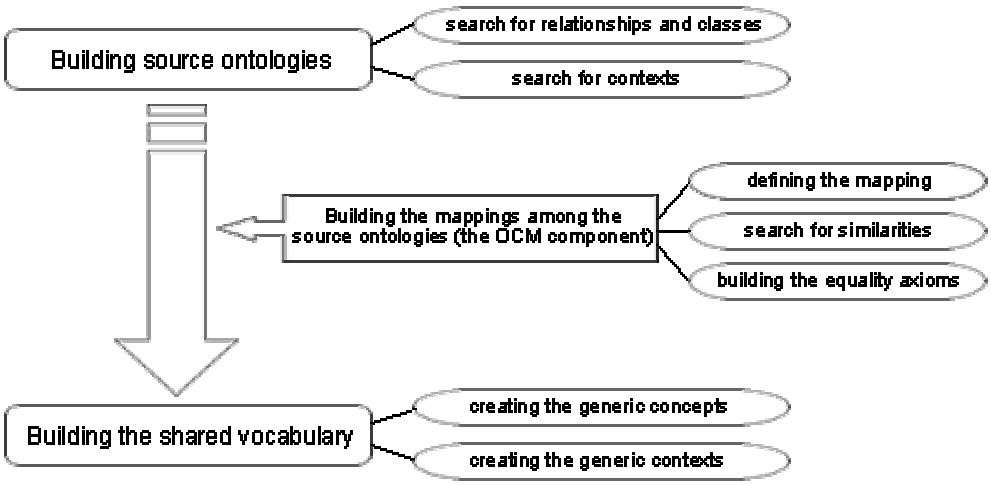

Figure 2. The Ontology Construction Method

3.1 Building source ontologies

As Figure 2 shows, this stage contains two main steps: *search for relationships and classes* and *search for contexts*. The first step implies a complete analysis of the information sources, e.g., what information is stored, how it is stored, the meaning of this information (the semantic), etc. Each ontology will be built independently by using any available tool, such as Protégé [13], DODDLE [17], Ontolingua [4] etc.

Figure 3 shows an example of two similar systems containing information about selling vehicles. The first system sells three types of vehicles: cars, trucks and pick-ups. The *buying_customer* class stores information about the vehicles sold to a customer. The second system also stores information about vehicles sold to a client (*buying_client*), but it divides the *vehicle* class into different subclasses. To build each ontology we use the Ontolingua Editor [12]. Each resulting ontology will contain the classes (and its attributes), the relationships and the axioms needed to define it. Figure 4 shows a part of the ontologies of each system represented using Ontolingua.

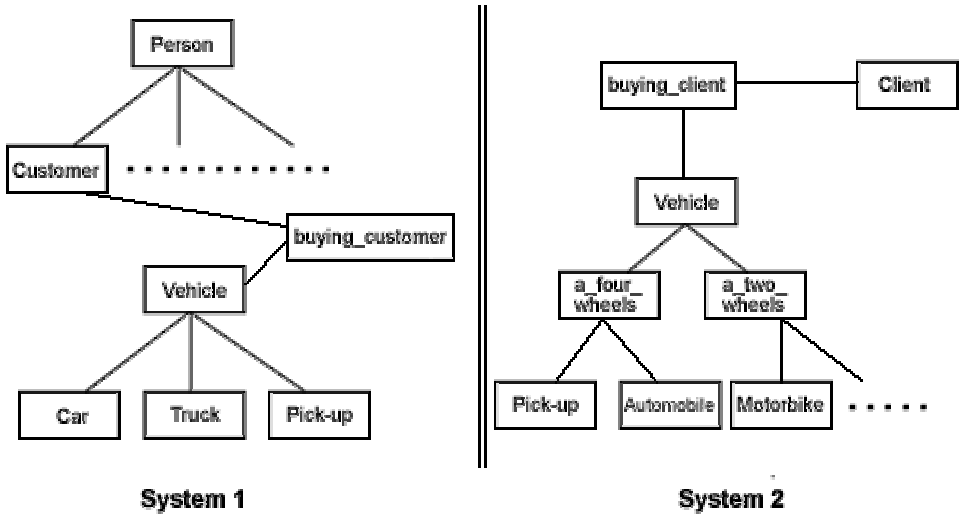

Figure 3. Two systems about selling cars

The second and last step, *search for contexts*, implies the definition of the specific context (or domain) for each ontology, and the definition of the several contexts within an ontology.

Here, we define the classes involved in each context. As we have explained in the last section, these contexts indicate the different roles or functions of a system. Figure 4 also describes some defined contexts for each ontology. In order to ease the finding of the mappings between ontologies, it is important that the contexts in the ontologies are not defined independently.

3.2 Building the mappings among the source ontologies (the OCM component)

As Figure 2 shows, this stage contains three main steps: *defining the mapping*, *search for similarities* and *building the equality axioms*.

The first step implies defining the relationships among the contexts of the source ontologies built in the last stage. This is a straightforward step because the contexts are defined globally. For example, some relationships can be:

$(O_1,Context_1)\ (c_{11}) = (O_2,Context_2)\ (c_{21})$
$(O_1,Context_1)\ (c_{11}) \cup (O_1,Context_1)\ (c_{13}) = (O_2,Context_2)\ (c_{23})$
$(O_1,Context_1)\ (c_{13}) = (O_2,Context_2)\ (c_{22})$
$(O_1,Context_1)\ (c_{14}) \subset (O_2,Context_2)\ (c_{26})$

The second step, *search for similarities*, is the most important step. Here, we should find the similarity values among the concepts related by the contexts. To do that, we use the similarity functions described in section 2.1.

Part of theOntology 1	**Part of theOntology 2**
;;; ---------- Classes ---------- ;;;Vehicle (Define-Class Vehicle (?X) "a conveyance that transports people or objects" :Def (And (Thing ?X))) ;;;Car (Define-Class Car (?X) "cars for selling" :Def (And (Vehicle ?X))) ;;;Person (Define-Class Person (?X) "the set of people" :Def (And (Thing ?X))) ;;;Customer (Define-Class Customer (?X) "the customer who buys vehicles" :Def (And (Person ?X))) ;;; ---------- Relations ---------- ;;;Last_Name (Define-Relation Last_Name (?Frame ?Value) "The last name of the person" :Def (And (Person ?Frame) (String ?Value))) ;;;Make (Define-Relation Make (?Frame ?Value) "The make of the vehicle" :Def (And (Vehicle ?Frame) (String ?Value)))	;;; ---------- Classes ---------- ;;;Vehicle (Define-Class Vehicle (?X) "a conveyance that transports people or objects" :Def (And (Thing ?X))) ;;;Automobile (Define-Class Automobile (?X) "automobiles for selling" :Def (And (A_Four_Wheels ?X))) ;;;Client (Define-Class Client (?X) "the client who buys vehicles" :Def (And (Thing ?X))) ;;; ---------- Relations ---------- ;;;Last_Name (Define-Relation Last_Name (?Frame ?Value) "The last name of the client" :Def (And (Client ?Frame) (String ?Value))) ;;;Make (Define-Relation Make (?Frame ?Value) "The make of the vehicle" :Def (And (Vehicle ?Frame) (String ?Value))) ;;;Door (Define-Relation Door (?Frame ?Value) "The amount of doors in an automobile" :Def (And (Automobile ?Frame) (Number ?Value)))
<u>Some contexts for these systems are:</u>	
$Ontology_1 = O_1$ $Context_1$ = selling_vehicles1 c_{11} = buying_car_customer c_{12} = buying_truck_customer c_{13} = buying_pickup_customer c_{14} = buying_vehicles_customer …….. $(O_1,Context_1) = \{ c_{11}, c_{12}, c_{13}, c_{14}\}$ $(O_1,Context_1)$ (c_{11}) = { customer, car, buying_customer, ...} $(O_1,Context_1)$ (c_{12}) = { customer, truck, buying_customer, ….} $(O_1,Context_1)$ (c_{13}) = { customer, pickup, buying_customer, ….} $(O_1,Context_1)$ (c_{14}) = { customer, vehicles, truck, car, pickup, buying_customer, make, ….} **……**	$Ontology_2 = O_2$ $Context_2$ = selling _vehicles2 c_{21} = buying_automobile_client c_{22} = buying_pickup_client c_{23} = buying_a_four_wheels_client c_{24} = buying_motorbike_client c_{25} = buying_a_two_wheels_client c_{26} = buying_vehicles_client …….. $(O_2,Context_2) = \{ c_{21}, c_{22}, c_{23}, c_{24}, c_{25}\}$ $(O_2,Context_2)$ (c_{21}) = { client, automobile, buying_client, door, ...} $(O_2,Context_2)$ (c_{22}) = { client, pickup, buying_client ….} $(O_2,Context_2)$ (c_{23}) = { client, a_four_wheels, pickup, automobile, buying_client, ….} **……**

Figure 4. Part of the two ontologies with some of their contexts

We must begin with the contexts containing only subclasses such as (O_1,Context$_1$) (c_{11}) and (O_2,Context$_2$) (c_{21}). In this way, when other contexts (involving superclasses and subclasses) such as (O_1,Context$_1$) (c_{14}) and (O_2,Context$_2$) (c_{23}) are compared, the included subclasses will not be compared again.

For instance, the equality relationship between (O_1,Context$_1$) (c_{11}) and (O_2,Context$_2$) (c_{21}) only contains subclasses and we will begin comparing the concepts included in them. Figure 5 shows the parts, the functions and attributes for the *car* class and *automobile* class of each ontology in our example. We have used WordNet [14] in order to look for the parts and functions of each concept.

(O_1,Context$_1$) (c_{11}) = buying_car_customer **Class Car** Parts = {accelerator pedal, door, window, car mirror, car seat,…, n_parts } Function = {a conveyance that transports people or objects} Attributes = {engine_number, make, model, color}	(O_2,Context$_2$) (c_{21}) = buying_automobile_client **Class Automobile** Parts = {accelerator pedal, door, window, car mirror, car seat,…, n_parts } Function = {a conveyance that transports people or objects} Attributes = {engine_number, make, model, color, doors, boot_capacity}

Figure 5. Parts, functions and attributes of two classes

The similarity function (3) applied to the parts is:

$$S_p(car, automobile) = \frac{n}{n + (\alpha(car, automobile)x0) + (1 - \alpha(car, automobile)x0)} = 1$$

In this case, the result is exactly 1 because the *car* class and the *automobile* class are synonyms. The same happens with the similarity function applied to the function because the *car* and *automobile* function are the same. Therefore $S_f(car, automobile) = 1$.

The similarity function applied to the attributes generates another result because the classes share only some attributes. First of all, the $\alpha(car,automobile)$ function must be calculated. To do so, we use the vehicle hierarchy of the two systems showed in Figure 3. The *depth(car)* function in the Ontology$_1$ is 2 whereas *depth(automobile)* is 3 for the Ontology$_2$. See [15] for more details on the calculations.

$$\alpha(car^{O_1}, automobile^{O_2}) = \frac{depth(car^{O_1})}{depth(car^{O_1}) + depth(automobile^{O_2})} = \frac{2}{2+3} = 0.4$$

Then the similarity function applied to the attributes is:

$$S_a(car, automobile) = \frac{4}{4 + ((0.4)x0) + ((0.6)x2)} = \frac{4}{4 + 1.2} = 0.64$$

Now, in order to conclude the calculation, the function (2) must be applied. In this case we assume equal values for the weights (w) because the contexts are related by an equality relationship. Therefore w_p=w_f=w_a=0.33. Then, the function (2) is :

$$S(car^{O_1}, automobile^{O_2}) = ((0.33)x1) + ((0.33)x1) + ((0.33)x(0.64)) = 0.87$$

As we can see, the final similarity value between *car* and *automobile* is very high, and surely, these classes will be in the equality axioms. Other similarity values for other classes included in the analyzed related contexts are:

$$S(customer^{O_1}, client^{O_2}) = 0.93$$
$$S(buying\_customer^{O_1}, buying\_client^{O_2}) = 0.76$$

These functions applied to other combination of classes generate null or low similarity values. For instance, if we compare the *car* class with the *client* class the similarity values are:

$$S_p(car, client) = 0, \;\; S_f(car, client) = 0 \;\; and \;\; S_a(car, client) = 0 \Rightarrow$$
$$S(car^{O_1}, client^{O_2}) = 0$$

These similarity values are not taken into account when we create the equality axioms.

The last step, *building the equality axioms*, also is a straightforward step because high similarity values must be looked for in the related context. In our example, and extending the language representation, some axioms are:

(<=> (Car ?a_car) (Automobile ?a_car))
(<=> (Customer ?a_client) (Client ?a_client))
(<=> (Buying_Customer ?a_purchase) (Buying_Client ?a_purchase))

3.3 Building the shared vocabulary

As Figure 2 shows, this stage only has two main steps: *creating the generic concepts* and *creating the generic contexts*. The former step can be achieved by using the equality axioms created in the last step of the previous stage. For each equality axiom, one concept must be chosen to be the generic concept within this vocabulary. For example, for the first axiom between *car* and *automobile* we can choose the word *car* to identify both. Once the generic concepts are chosen, the ontologies must be used to generate a unique ontology or shared vocabulary. This vocabulary will be used by the users to query the federated system.

In Figure 6 a part of the resulting ontology is represented using the chosen concepts for each case. Again, here we use Ontolingua as specification language, but Figure 6 shows the concepts also graphically in order to make the example clearer.

The latter step, *creating the generic context*, consists of determining the contexts that will be used by the users. The contexts must include (like in the first stage) its generic concepts. Figure 7 shows some examples of them.

4 Conclusions and Future Work

In our proposal, we have combined two powerful tools – ontologies and context information – to help solve many semantic heterogeneity problems. We create a new approach using three main components within a federation layer: *source ontologies, ontology and context mapping (OCM)* and *shared vocabulary*. Each of them contains semantic information using ontologies and contexts in order to achieve a consistent integration. We have also presented an approach to build the

components based on three main stages: *building source ontologies*, *building the mappings among the source ontologies (the OCM component)* and *building the shared vocabulary*.

Our ongoing approach should still analyze a number of aspects. For example, we are working on defining another relationships among contexts, and the exact values of the weights (*w*) in the similarity functions. These values might be based on context relationships, taking into account another problems about ontological heterogeneity not considered in this work, such as attribute-type mismatches, structure mismatches, concept mismatches, etc. Also, some automated processes are being developed to improve and make more efficient some tasks such as the relationships among contexts, the comparison between two concepts, etc.

Finally, the approach and their extensions need to be validated by using more complex examples and real cases for study.

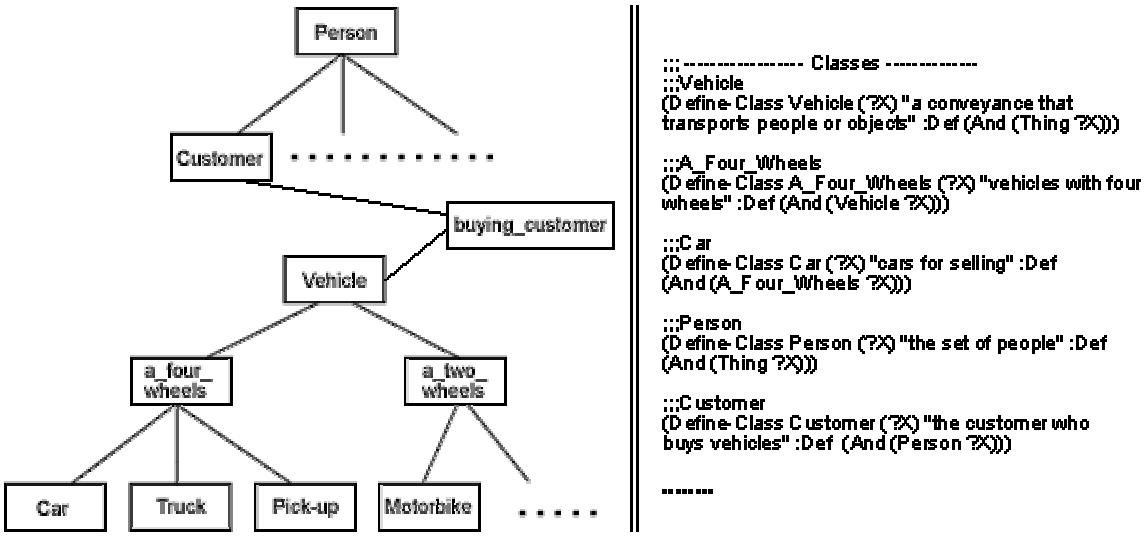

Figure 6. The resultant shared vocabulary (or ontology)

buying_car_customer= { customer, car, buying_customer, ...}

buying_truck_customer= { customer, truck, buying_customer, ...}

buying_pickup_customer= { customer, pickup, buying_customer, ...}

buying_vehicles_customer = { customer, vehicles, a_four_wheels, truck, car, pickup, buying_customer, make, ...}

buying_four_wheels_buyer={ customer, a_four_wheels, pickup, car, truck, buying_customer, ...}

.....

Figure 7. Some generic contexts

References

1. Borgida, A., Brachman, R.J, McGuinness, D.L. and Resnick, L.A. CLASSIC: A structural data model for objects. In Proceedings ACM SIGMOD-89, Portland, Oregon, 1989.
2. Buccella A., Cechich A. and Brisaboa N.R. "Applying an Ontology on Data Integration" , WICC'03, 5th Workshop of Investigations on Computer Science. Universidad Nacional del Centro de Buenos Aires, Tandil –Argentina, (Pages 99-102). May 2003.
3. Busse, S., Kutsche, R.-D., Leser, U., Weber H. Federated Information Systems: Concepts, Terminology and Architectures. Technical Report. Nr. 99-9, TU Berlin. April 1999.
4. Farquhar, A., Fikes, R., Rice, J. The Ontolingua Server: A Tool for Collaborative Ontology Construction. Proceedings of KAW96. Banff, Canada, 1996.
5. Fowler, M. and Scott, K. *UML distilled*, Addison-Wesley 1997.
6. Goh, C.H., Bressan, S., Siegel, M. and Madnick, S. E. Context Interchange: New Features and Formalisms for the Intelligent Integration of Information. ACM Transactions on Information Systems, Vol. 17, No. 3, (Pages 270–293). July 1999.
7. Gruber T. Ontolingua: A Mechanism to Support Portable Ontologies. Knwoledge Systems Laboratory, Stanford University, Stanford, CA, Technical Report KSL 91-66. 1992.
8. Hasselbring, W. Information System Integration. Communications of the ACM. June 2000.
9. MacGregor, R. Inside the LOOM clasifier. SIGART bulletin. Nº 2(3): (Pages 70-76). June, 1991.
10. Mena, E., Kashyap, V., Sheth, A. and A. Illarramendi, A. Managing Multiple Information Sources through Ontologies: Relationship between Vocabulary Heterogeneity and Loss of Information. In Proceedings of Knowledge Representation Meets Databases (KRDB'96), ECAI'96 conference, Budapest, Hungary, August 1996, pp. (Pages 50-52). 1996.
11. Mena, E., Kashyap, V., Sheth, A. and A. Illarramendi, A. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. Kluwer Academic Publishers, Boston. (Pages 1-49). 2000.
12. Ontolingua Editor. http://ontolingua.stanford.edu/index.html.
13. Protégé 2000. http://protege.stanford.edu/doc/users_guide/index.html.
14. Richardson, R. and Smeaton, A. Using WordNet in a Knowledge-Based Approach to Information Retrieval. Technical Report CA-0395, Dublin City Univ., School of Computer Applications, Dublin, Ireland, 1995.
15. Rodriguez, A., Egenhofer, M. Determining Semantic Similarity among Entity Classes from Different Ontologies. IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 2, March/April 2003.
16. Rodriguez, A., Egenhofer, M. Putting Similarity Assessments into Context: Matching Functions with the User's Intended Operations. Context 99, Lecture Notes in Computer Science, Springer-Verlag, September 1999.
17. Sekiuchi, R., Aoki, C., Kurematsu, M., and Yamaguchi, T. DODDLE : A Domain Ontology Rapid Development Environment. In Proc of PRICAI 98,1998.
18. Siegel, M. and Madnick, S. E. A metadata approach to resolving semantic conflicts. In Proceedings of the 17th Conference on Very Large Data Bases (Barcelona, Spain, Sept.). VLDB Endowment, Berkeley, CA, (Pages 133–145). 1991
19. Tversky, A. Features of Similarity. Psychological Rev., vol. 84, (Pages 327-352). 1977.
20. Visser, P., Jones, D., Bench-Capon, T., Shave, M. An Analysis of Ontology Mismatches; Heterogeneity versus Interoperability. AAAI 1997 Spring Symposium on Ontological Engineering.
21. Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H.. Schuster, G., Neumann, H. and Hübner, S. "Ontology-based Integration of Information - A Survey of Existing Approaches," In:

Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, WA, Vol. (Pages 108-117). 2001.