Towards an Ontology-based Integration of Federated Information Sources

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. Integrating data from a Federated System is a very complex process that involves a series of tasks. Characteristics such as autonomy of the information sources, their geographical distribution and heterogeneity are some of the main problems we face to perform the integration. In this paper we focus on the problem of heterogeneity, more specifically on semantic heterogeneity. The semantic heterogeneity makes the integration difficult because of its bearing problems on synonymous, generalization/specialization, etc. Here, we briefly explain our three level approach to solve these problems. Then we show the structure of software components used to implement our supporting tool.

Keywords: Federated Databases, Ontology, Semantic Heterogeneity.

1. Introduction

The federation of different data sources is a long-standing and thoroughly studied problem. Since the appearance of the ontologies and the proliferation of the *Semantic Web* [1], this problem has regained much attention. Nevertheless, integration of data from different sources is still an open issue. The autonomy of the information sources, their geographical distribution and the heterogeneity among them, are the main problems we must face to perform the integration [12]. The semantic heterogeneity has been one of the most researched aspects in the last years. Works like [7,13] are aimed to fill the semantic gap among the information sources, using the semantic information provided by the ontologies.

In recent works [2,3,4,5,6] we have proposed an architecture and a three level approach to solve semantic heterogeneity problems [13]. Our architecture [3,4] is based on three main components: a *global ontology* or *shared vocabulary*, an *ontology mapping* and *source ontologies*. The global ontology component contains the generic concepts that will be used to query the system. The *Ontology Mapping (OM)* component deals with the information flow between the source ontologies and the shared vocabulary. This component contains a set of mappings relating concepts in the *sources ontologies* with concepts in the shared vocabulary. Once a user chooses the concepts from the shared vocabulary and makes a query, the system uses the OM component to know which concepts are related with. Thereby, through the source ontologies, the system gets access to the information sources to produce the data. There is only one source ontology for each information source.

To develop the architectural components, our method [6] defines three main stages: *building the source ontologies*, *building the mappings among source ontologies* and *building the shared vocabulary*. Each of these stages serves as a guideline to create the aforementioned components.

This paper is organized as follows: Section 2 presents a summary of our three level approach to find similarities between concepts. Then, in Section 3 we will briefly describe the software

components used to develop our automated tool, which implements our approach to calculate similarity of concepts. Future work and conclusions will be discussed afterwards.

2. A Three Level Approach for Searching Similarities

Finding similarities between concepts of different ontologies is a very complex activity. In general, it is not possible to determine fully automatically all mappings between them. Therefore, our approach only determines mapping candidates that users should accept, reject or change. Furthermore, a user might specify mappings for concepts for which the system was unable to find satisfactory match candidates.

Figure 1 shows our three level approach [3,4] for searching similarities among concepts. In this paper, we only show the structure of our method; for a detailed description including the similarity functions applied in it, we refer the reader to [3,4].

In the figure, the double bounded boxes represent external modules used to retrieve information.

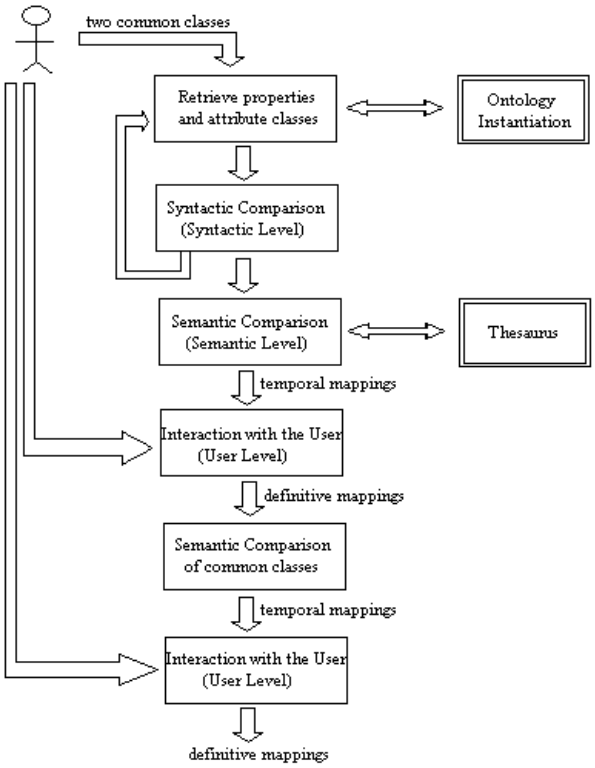

Figure 1. Method for searching similarities

First of all, the *Ontology Instantiation* module obtains the object structure [4] from ontologies described for example in OWL [15]. The ontology is divided into classes and properties. The classes are subdivided into *common classes* and *attribute classes*. The *common classes* have the role of representing things about the domain, and the *attribute classes* have the role of representing information about a common class (attribute). For example, an ontology can have the *Animal* class as a common class and the *Organ* class as an attribute class because *Organ* exists to describe a characteristic about a common class. The *Organ* class has no properties. On the other hand, the properties are also subdivided into *datatype properties* and *special properties*. The properties have restrictions to denote functions, cardinality, domain, range, etc. The *datatype properties* are properties relating a class or a set of classes (intersection of classes) with a data type; for example, "animal name" is a datatype property between the *Animal* class and the *String* data type. The *special properties* are properties relating classes to each other. For example, relationships relating

the *Animal* class to the *Organ* class in order to describe the organs of an animal. Thus, a common class has both datatype properties and special properties, and attribute classes do not have properties.

The *Thesaurus* module is an important source of semantic information. Thesauruses are used to search for synonyms, which are detected by the module through the use of a similarity function. The function returns 1 if a synonym relationship is found, and 0 otherwise.

Accordingly to our approach, a user indicates the first mapping, for example between the *Animal* class of one ontology and the *Creature* class of another ontology. These two mapped classes are inputs of the *Retrieve Properties and Attribute Classes* module. This module retrieves the attribute classes and special and datatype properties of each concept by using the object structure of each ontology. This information enters as input of the *Syntactic Comparison* module, which syntactically analyses classes and properties relating with the concepts.

Then, special properties included in the common classes must be compared. The comparison is similar to the previous case, but the data type of the properties is not compared because the ranges of the properties are also classes. Therefore we take into account the range (classes) of the special properties together with their properties. Therefore, this is a recursive method that will stop when the ranges are attribute classes (because they do not have properties).

Finally, the classes are compared syntactically. Results of all calculations are stored to be used by the following module. It is the *Semantic Comparison* module, which compares the classes and properties semantically. To do so, we extract semantic information of the Thesaurus module. Again, the algorithm is divided into comparisons between datatype and special properties and classes. Using the results of the syntactic level functions, we construct functions that combine these values together with the thesaurus information.

All temporal mappings found for the properties and attribute classes will be showed to the user and they will decide if the mappings are correct. This is the main task of the *Interaction with the User* module within the User Level. The mappings accepted by the user will be classified as definitive mappings.

Then, if the classes are common classes, these definitive mappings enter to the *Semantic Comparison for common classes* module. This module works at the semantic level, and uses the mappings added by the comparison of properties in order to denote the set of similar attributes of both classes. A temporal mapping is added if the final function exceeds the threshold.

Once all similarity values are obtained for two classes, the temporal mappings are displayed to the user (in the Interaction with the User module) and again they must decide if these mappings must be added permanently.

3. A Supporting Tool

In order to implement our approach for searching similarities, we have developed an automated software tool. The tool was developed using the Java Platform [10], and Eclipse [8] as the working environment. The interfaces were created in the Web browser. The connection between the users and the server is made by using the technology of Java Servlet [11]. The server uses the Linux operating system and the PostgreSQL [14] database. Currently, the system is off-line and only works locally because it is still under testing.

Figure 2 shows the structure of software components used to implement the supporting tool. The diagram shows the components and their dependencies where the structure is represented using the UML notation [9]. We refer the reader to [5] for a detailed description of the components, their interfaces and dependency relationships in the diagram.

Following, the most important components are briefly described in terms of their interfaces and sub-components, and others are only mentioned for brevity reasons.

- The *Coordinator Component*: The intent of this component is to coordinate all the processes accordingly by using each component at a time. Once the ontology is loaded by

the user (in OWL Language [15]), the Coordinator calls the Parser and Instantiation Component to obtain an object structure (representing an instantiation of the Ontology Model Component) as a result. In this way the whole ontology, its common and attribute classes and its special and datatype properties, will be objects of the Ontology Model. In order to calculate the similarity values among the concepts included in the related contexts, the Similarity Searcher Component is invoked.

- The *Parser and Instantiation Component*: The component should parse the OWL code loaded by the user in order to create an object structure which represents a valid instantiation of the Ontology Model Component. Besides, error codes generated during the parsing process or the creation of an instance are returned to the Coordinator Component. Users should use some ontology editor such as Protégé [16] to avoid syntactic problems.
- The *Ontology Model Component*: This component corresponds to the Java translation [10] of the Ontology.
- The *Similarity Searcher Component*: This component has the task of calculating the similarity values. It uses the Ontology Model Component in order to obtain the common and attribute classes, and the special and datatype properties, and to use them in our similarity method (Figure 1).

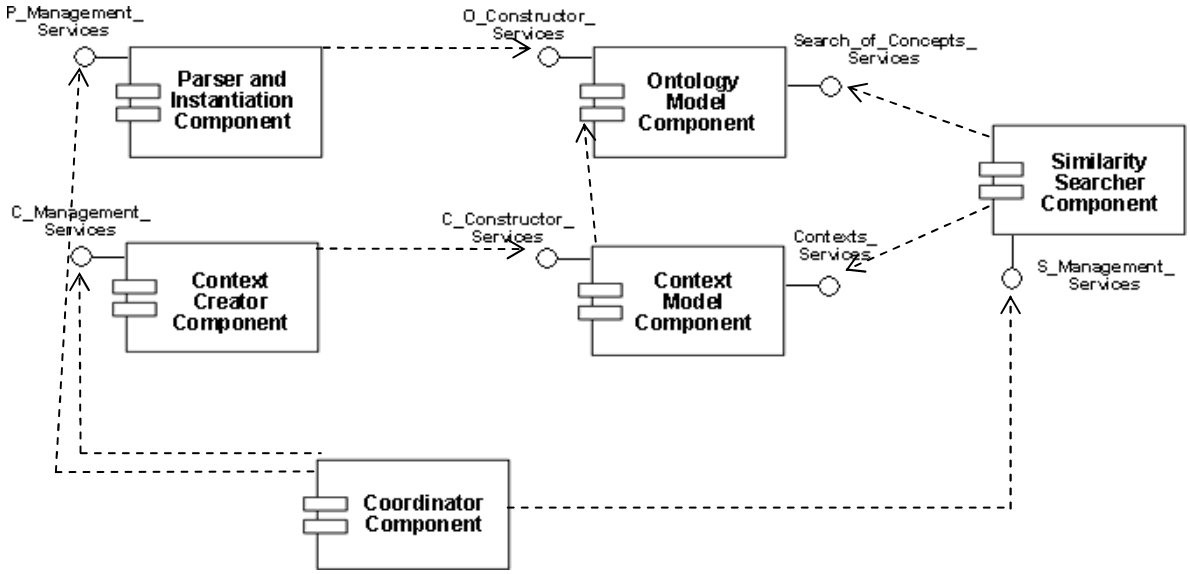

Figure 2. Component diagram of the software tool

In this first stage of our work we do not implement the *Context Creator* and *Context Model* Component. Contexts, as ontologies, are useful tools to model concepts which are in conflict with one another. Possible implementations of these two components and contexts are included in future works.

4. Conclusions

We have briefly presented our three level approach and the structure of software components used to implement the supporting tool that calculates the similarity among concepts included in different ontologies. This stage, called *building the mappings among source ontologies*, has a series of steps involved which should be followed to come up to the final results. The tool widely simplifies the user's work when they are making integration tasks.

Our work is, by now, in a development stage for a number of tasks which are still being developed. As a future work, we are testing our tool with real cases of study and comparing our approach with others in the literature in order to show the advantages and problems associated with it. Also, we are working on the implementation of contexts to obtain more advantages in the process of searching similarities.

5. References

1. Berners-Lee, T. Weaving the Web. Texere Publishing Ltd. ISBN: 0752820907. June 2001.
2. Buccella A., Cechich A. and Brisaboa N.R. Ontology-based Data Integration: Different Approaches and Common Features. *Encyclopedia of Database Technologies and Applications*. Rivero,L., Doorn, J. and Ferraggine, V. Editors. Idea Group Publishing, to appear 2005.
3. Buccella A., Cechich A., and Brisaboa N. A Federated Layer to Integrate Heterogeneous Knowledge, VODCA 2004: First International Workshop on Views on Designing Complex Architectures, Bertinoro, Italia, 11-12 Sept. 2004. Electronic Notes in Theoretical Computer Science, Elsevier Science B.V.
4. Buccella A., Cechich A. "A Three-level Approach to Determine Ontological Similarity". Jornadas Chilenas de Computación. JCC 2004. Arica, Chile, 10-12 de Noviembre 2004.
5. Buccella A., Cechich A. and Brisaboa N.R. An Ontology-based Environment to Data Integration. *VII Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes de Software*. IDEAS 2004, pp. 79-90, 3-7 May, 2004.
6. Buccella A., Cechich A. and Brisaboa N.R. An Ontological Approach to Federated Data Integration. 9° Congreso Argentino en Ciencias de la Computación, CACIC'2003, La Plata, October 6-10, pp. 905-916, 2003.
7. Euzenat, J., Valtchev, P. An integrative proximity measure for ontology alignment. CEUR Workshop Proceedings. Sanibel Island, Florida, October 20, 2003.
8. Eclipse Home Page. http://www.eclipse.org.
9. Fowler, M. and Scott, K. UML distilled, Addison-Wesley 1997.
10. Java SE Platform. http://java.sun.com.
11. Java Servlet. http://java.sun.com/products/servlet/.
12. Hasselbring, W. Information System Integration. *Communications of the ACM*. June 2000.
13. Maedche, A. and Staab, S. Measuring Similarity between Ontologies. In: Proc. Of the European Conference on Knowledge Acquisition and Management - EKAW-2002. Madrid, Spain, October 1-4, 2002. LNCS/LNAI 2473, Springer, 2002, (pp. 251-263).
14. PostgreSQL Home Page. http://www.postgresql.org/.
15. Smith, M.K.,Welty, C., McGuinness, D.L. OWL Web Ontology Language Guide. W3C, http://www.w3.org/TR/2004/REC-owl-guide-20040210/. February 10, 2004.
16. Stanford, U., Protégé 2000, Available at http://protege.stanford.edu/doc/users guide/index.html.