Improving Software Agent Communication with Structural Ontology Alignment Methods

Jairo F. de Souza, DCC/ICE/UFJF, Brazil
Rubens N. Melo, DI/PUC, Brazil
Jonice Oliveira, DCC/IM/UFRJ, Brazil
Jano de Souza, DCC/COPPE/UFRJ, Brazil
Sean Wolfgang M. Siqueira, DIA/UNIRIO, Brazil

ABSTRACT

To perform tasks on the semantic web, software agents must be able to communicate with other agents using domain ontologies, even when considering different ontologies. In this regard, it is necessary to address semantic interoperability to enable agents to recognize common concepts and misunderstandings. In this paper, the authors propose the use of negotiation concepts in business scenarios for addressing concept compatibilization problems in communication between software agents and present an algorithm developed in the GNoSIS system. A validation of this approach is presented.

Keywords: Negotiation Concepts, Ontologies, Semantic Interoperability, Similarity, Software Agent

INTRODUCTION

Dealing with systems interoperability has been a research issue for some time, but the use of a knowledge structure to allow system interoperability – whether in communication between agents, in database integration or still in other scenarios – has still several problems. Interoperability is compromised when different knowledge structures are used and overlapping domain concepts can become a computing issue.

According to O’Hara (2004), the highest layers of the Semantic Web architecture contain social phenomena that cannot be overlooked in computational solutions (such as the trust layer). As the structuring of knowledge is present in the upper layers of the Semantic Web, a genuinely social phenomenon that can be observed is related to the achievement of consensus for the creation and compatibilization of these knowledge structures (in our case, ontologies).

To execute their tasks, software agents need to be interactive and adaptive, that is, they should be capable of receiving and sending

DOI: 10.4018/jitwe.2010070103
messages to other agents or to the environment and should be capable of understanding these messages. The understanding of the messages takes place through a standardization of the vocabulary of the agents. The attaining of such compatibility can be made with the use of domain ontologies. In open environments, however, software agents are subjected to receiving messages from agents that do not share the same standardized vocabulary, which characterizes one of the challenges in this area. The software agents will be responsible for dealing with the harmonization of ontologies (Breiman et al., 2007), discovering similarities between concepts or the wrong interpretation of some concept during the communication with other agents, to execute some task that requires interaction between agents.

However, harmonizing ontologies is a hard task and stills an unsolved issue. The ontological divergences can be divided into (1) divergences on the level of language (differences caused by the use of different formalisms) and (2) divergences on the level of conceptualization (differences related to the structuring of the concepts in the ontology) (Klein, 2001).

Divergences on the level of language are solved with the changing the formalism of one or of the two ontologies. The changing of the formalism also generates new problems, such as those caused by the difference in expressiveness of a formalism in relation to the other but even then this is the most adequate solution to solve this type of divergence. In this work, we adopted OWL language as the standard for ontology description and thus do not deal with divergence issues of this level.

Divergences on the level of conceptualization occur, amongst other cases, due to a difference in coding, use of synonyms, use of distinctive generic ontologies, difference in granularity between the ontologies, etc. These cases demand a comparison of the structure of the concepts and of the context, that is, a semantic comparison. Syntactic comparisons can add good results to semantic comparisons by finding semantic relations between terms, as it happens in many algorithms that mine text corpus (Chakrabarti, 2000; Faatz & Steinmetz, 2002).

This work presents a general architecture that allows the harmonization of ontologies during the communication of software agents. The architecture addresses how agents must encapsulate alignment functions to harmonizing ontologies and also explores negotiation concepts commonly applied in the business scenario and validated in the context of negotiation of meanings with human users (Oliveira et al., 2007; Souza et al., 2006).

In order to deal with ontology compatibility issue in this communication we propose an algorithm that uses both syntactic and semantic techniques in a structural approach, according to the classification proposed in Shvaiko and Euzenat (2005). This algorithm uses resemblance functions and calculates the degree of similarity between the concepts in a recursive manner, calculating the degree of similarity between two concepts based on the degree of total similarity between the concepts that have close kinship.

This algorithm was implemented as part of the GNoSIS system (Oliveira et al., 2007; Souza et al., 2006), created to allow groups of domain specialists to negotiate meanings of concepts as described in ontologies and produce the mappings between the ontologies. The GNoSIS recommends mappings based on the calculation of similarity between concepts and after that the domain specialists negotiate the validity of the mapping and the differences in conceptualization.

NEGOTIATING TO REACH CONSENSUS

Negotiation can be defined as a process through which a group of agents communicates with one another to try and come to a mutually-acceptable agreement on some matter (Lomuscio et al., 2003). This cooperative negotiation is classified as Win/Win. In the definition, the stress is placed on words such as ‘agent’, ‘communicate’, and ‘mutually acceptable’. The parties taking part
in the negotiation process are not necessarily people, but can be any type of actors, such as software agents. These actors communicate according to a negotiation protocol and act according to a strategy. The protocol determines the flow of messages between the negotiating parties and acts as the rules the negotiating parties must abide to if they are to interact. The protocol is public and open. The strategy, on the other hand, is the way in which a given party acts within those rules in an effort to get the best outcome from the negotiation. The strategy of each participant is, therefore, private. Two important variables in negotiation are called BATNA and reserve value.

BATNA (Fisher et al., 1994), the acronym for “Best Alternative To a Negotiated Agreement”, can be identified in any negotiation situation by the question, “What will we do if this negotiation is not successful?”. In the simplest terms, if the proposed agreement is better than your BATNA, then you should accept it. If the agreement is not better than your BATNA, then you should reopen negotiations. If you cannot improve the agreement, then you should at least consider withdrawing from the negotiations and pursuing your alternative (though the costs of doing so must also be considered). One of the main reasons for entering a negotiation is to achieve better results than would be possible without negotiating (Spangler, 2008). The reserve value (also called base or reservation price) represents the least favourable point someone will accept an agreement at.

The reserve value should stem from the BATNA, although not always the best alternative for the value of a negotiated attribute, should an agreement not be reached, is identical to the reserve value set for the same attribute. In a real negotiation about buying cars, for instance, a buyer can define a reserve value that not exceeds $30,000. However, once this price cannot be reached in the negotiation, his BATNA can aim as best alternative not buying the car but invest this amount for some months and wait car’s price decrease.

The BATNA and the reserve value, apart from other components not mentioned here, can be used in computing environments as inputs for the creation of negotiation strategies (Paula, 2006). The relation about these concepts will be discussed in next section. These components are part of the strategy for each agent in the negotiation and therefore are private components that are part of the logic of the agent. The negotiation protocol, as a public element, defines the agent communication style to negotiate the ontology alignments.

## ADDING MEANING NEGOTIATION SKILLS TO MULTI-AGENT SYSTEMS

Software agents’ communication languages, such as KQML, allow describing domain ontologies used in the message content. The negotiation of meanings for software agents takes place through the alignment of their respective domain ontologies. This negotiation of meanings is done immediately after the identification of (1) a term that does not exist in the ontology of the agent that receives the message or (2) homonymous terms. As domains are denoted in ontology languages as namespaces, the terms described in ontologies as classes and that are in distinct namespaces are considered possible homonyms. The Figure 1 shows two messages sent by two agents, A and B, using the standard KQML agent communication language (Finn et al., 1994), where agents use the “process” concept of two ontologies (CYC and SUMO) with distinct namespaces.

In the MSG1, agent (:sender) A request the value of a variable process to agent B (:receiver). The agent A sends the message (:content) that has the term process to agent B. This term is described by the high-level ontology (:ontology) openyc (Lenat & Guha, 1990). Agent B replies to the message (MSG2) answered the query received also using a process term, albeit described in the SUMO high-level ontology (Niles & Pease, 2001). As the ontologies have distinct namespaces, the terms can be constructed as possible homonymous concepts. This approach to identify homonym-
Figure 1. Possible homonymous concept. The message MSG1 has an term “process” from ontology opencyc as its content and the message MSG2 has an homonymous term “process” from ontology SUMO as its content.

<table>
<thead>
<tr>
<th>[Message MSG1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>{evaluate</td>
</tr>
<tr>
<td>:sender A</td>
</tr>
<tr>
<td>:receiver B</td>
</tr>
<tr>
<td>:language KIF</td>
</tr>
<tr>
<td>:ontology opencyc</td>
</tr>
<tr>
<td>:reply-with q1</td>
</tr>
<tr>
<td>:content {val (process m1)}</td>
</tr>
<tr>
<td>} CYC namespace: <a href="http://opencyc.sourceforge.net/daml/cyc">http://opencyc.sourceforge.net/daml/cyc</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[Message MSG2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>{reply</td>
</tr>
<tr>
<td>:sender B</td>
</tr>
<tr>
<td>:receiver A</td>
</tr>
<tr>
<td>:language KIF</td>
</tr>
<tr>
<td>:ontology SUMO</td>
</tr>
<tr>
<td>:in-reply-to q1</td>
</tr>
<tr>
<td>:content {= (process m1) {scalar 12}}</td>
</tr>
<tr>
<td>} SUMO namespace: <a href="http://reliantknowledge.daml/sumo">http://reliantknowledge.daml/sumo</a></td>
</tr>
</tbody>
</table>

Mous concepts is very simplistic but quite useful as distinct ontologies have no indication of correlation between concepts with the same name (providing this correlation is not explicit in the ontology via some type of relation, such as mappings).

The identification of homonymous terms is carried out to identify a need for alignment between the terms. The alignment process should be capable of confirming if the terms truly denote concepts that are different, supplementary, or equal. In some cases, however, the terms can denote equal concepts but with distinct restrictions, such as cardinality restrictions. The problem with the harmonization of concepts is its complexity and therefore it is not a problem that can be fully automatic, requiring human intervention (Klein, 2001). However, it is not reasonable to think that direct and constant human intervention in a multi-agent environment to solve compatibility problems in ontology concepts. Thus, is possible to see that this work, as well as related works on the automatic alignment of concepts in ontologies has a positive outcome, in their majority, in simpler structures and with alignments that are not too complex.

Once a need of alignment is identified, the agents should start this procedure through a process of negotiation defined in the negotiation protocol. The negotiation protocol sets the types of messages that can be sent and how messages are related.

Once an agent A identifies a non-existing term in its domain ontology, it replies to a message (KQML performative: ask-about) to the agent B, sender of the message requesting the definitions of the term contained in the message, that is, its set of restrictions and their hierarchical and non-hierarchical relationships. Agent B, in its turn, replies to the message (KQML performative: reply) to agent A with the term definition (i.e., an excerpt from agent ontology with the concept definitions). If the definition has new, non-identified terms, agent A repeats the process for the new terms until agent B has no more definitions to reply about (ending the negotiation without alignment) or until agent A has no more queries (defining the alignment). This process is based on the technique of negotiation of meanings as described by Teresa Pica (1987), which describes the process through which people try to reach an understanding of words that are not understood, as used during a
dialogue. A simple example of this process for
the negotiation of meanings is given in Figure 2.

In the example above, sender A sends a
message. Receiver B does not know one of its
terms and sends a message querying the mean-
ing from B. The sender and receptor exchange
messages to define the concept received
until receiver B understands the meaning of the
term. Firstly, the sender sends the concepts that
are closest in the hierarchy to the concept being
queried. This way, sender A sent the definition
of Triangle using its Polygon superclass. Poly-
gon is a concept known to receiver B and this,
in its turn, sends a confirmation message.
Sender A then forwards the definition of the
Triangle concept based on its attributes. Re-
ceiver B confirms the sending and creates its
definition of Triangle. Figure 3 shows the
definitions exchanged by the players as descritp-
logic sentences that are described in the
OWL language by the agents and interpreted
by logical reasoning agent for description logi-
cs.

The alignment process, however, most of
the time needs to use more sophisticated tech-
niques to find the correct alignment between
concepts, such as in distinct domains, as the
difficulty to recognize synonyms and hom-
onyms is a challenge. In this case, techniques
for the calculation of similarity are used. The
techniques for the calculation of similarity
usually use the names of concepts and prop-
ties (syntactic techniques) and the ontological
structure and instances (semantic techniques)
to recognize the concepts with a higher degree
of similarity (see “Concept Similarity Analysis
section”).

To provide compatibility to the concepts
during the process of conversation of agents in
a meaning negotiation approach, independently
from the similarity techniques that are used, we
proposed the agent development architecture
from Figure 4. An extension was built of the
Goddard agent architecture (Truszkowski,
2006), with the adding of highlighted boxes
to the agent communication module (Agent
Communication Perceptor/Effector).

The Goddard agent architecture subdivides
the agent in 8 modules and defines the rela-
relationships between these modules. The main
module for the previously described problem is the
communication module (Agent Communication
Perceptor/Elfactor), which is responsible for
interacting with the environment, receiving
messages in an agent communication language.
These messages are sent to a reasoning agent,
responsible for interpreting the message and
processing the contents of the message using
the plan defined in the Planning and Schedul-
ing module and the current state of the agent (Mod-
elling and State module). Once the decision for
the next step of the agent is made, it executes
the action (Execution) and alters the environ-
ment (Effector).

The communication module was expanded,
being detailed into 4 sub-modules. When receiv-
ing a message, the Receptor/Sender sub-module
is responsible for forwarding the concepts that
exist in the message to the sub-module Ontology
Reasoning, which represents the logic inference
machine that is used to process the definition of
the concepts of the ontologies. This sub-module
checks if the concepts contained in the message
are defined in the agent ontology through a
syntactic comparison between the canonical
names of the concepts as described in OWL
(namespace#concept). If it has been defined,
the Receptor/Sender sub-module forwards the
message to the agent reasoning module which
is responsible for the understanding of the mes-
sage and for decision-making pursuant to the
BDI (Belief-Desire-Intention) model (Bratman,
1999). On the other hand, if the concept is not
defined in the domain ontology of the agent,
the Ontology Reasoning sub-module consults
the negotiation plan in the Negotiation Planning
sub-module, which stores the techniques that
will be used to attempt the alignment of the
concepts and the reserve value for the negotia-
tion of the agent. The alignments considered
as valid by the Ontology Reasoning are stored
in the Mapping Base sub-module and used in
the future to avoid repeated negotiations and to
enrich the inference process in new negotiations.
Figure 2. Example of negotiation of meanings

| A: it's the triangle.          | B: triangle?                  |
| A: the triangle is the polygon. | B: ok, the triangle is the polygon. |
| A: the triangle has exactly three sides. | B: ok, the triangle is the polygon with exactly three sides. |

Figure 3. Definitions in description logics

Agent A:
- Triangle ⊆ Polygon
- Triangle ⊆ (≥ 3 hasSide) ∩ (≤ 3 hasSide)

Agent B:
- Triangle ⊆ Polygon ∩ (≥ 3 hasSide) ∩ (≤ 3 hasSide)

Figure 4. Agent architecture. Shows the negotiation of meaning components added in Agent Communication Perceptor/Effect component.

If the agent cannot align the concepts of the ontologies, the negotiation is considered as non-satisfactory. The agent’s BATNA, described in the Negotiation Planning sub-module, defines the best alternative to an unsuccessful negotiation. In this case, the agent should try to use the BATNA to implement an action that may have a relevant outcome. If the agent is making a consultation in another agent to present to the user, for example, the agent can present the data as probable but not reliable results. For agents that receive messages that contain some decision-making in the environment, the use of the BATNA concept cannot be fully applied, given the uncertainty that implementing the action can have when a concept is not satisfactorily aligned.

CONCEPT SIMILARITY ANALYSIS

This section presents the method to calculate the similarity of ontology concepts that we use
during software agent meaning negotiation process. This method is available as alignment method in Negotiation Planning module (Figure 4). The method receives pairs of ontologies (or subontologies) as input and returns a table with the most similar concepts. To fill this table, the module analyzes the ontologies in a syntactic manner (names of concepts and properties) and semantic manner (hierarchy, relationships and property constraints). The value for total similarity between two concepts is recursively calculated based on the hierarchy of concepts and their relationships.

The calculation of similarity is done by using unsymmetrical similarity functions $F(a, b)$, where $F(a, b), i=1,k; F: A \times B \rightarrow [0, 1]$, $A$ is the set of ontology elements of the agent reasoning the ontology and $B$ is the set of ontology elements of the agent that sent its ontology. For example, $F(a, b)$ can be a function such as “the quantity of properties with the same name of a concept”, where value 1 can be returned if the ontologies have the same number of properties with the same name, and 0 if they do not have any identical property. Intermediate values represent the degree of similarity between the concepts and, the highest the value for $F(a, b)$, the greater the similarity between the concepts compared will be.

For each ontological element (class, property, constraints, etc.), a set of similarity functions is applied. At first, all the constructors used to create an ontology can be useful in the calculation of similarity and thus have one or more associated similarity functions. We used similarity functions in this approach applied to concept names, properties, and relationships.

Each property of a concept of the ontology has similarity functions that analyze its name and type. In OWL, the properties can be data type or object properties, represented by constructors owl:DatatypeProperty and owl:ObjectProperty, respectively. Data type properties are those that are intrinsic to the concept, that is, that do not relate to other concepts. For example, data type properties for the “Person” concept can be: name, age, size, etc. In its turn, object properties are those that are extrinsic to the concept, i.e., that relate to other objects. For the same “Person” concept, the examples of object properties can be: “works in” or “is the son of”.

Data type properties have primitive data types and object properties have complex data types, that is, other concepts of the ontology. Thus, for example, property “name”, as mentioned before, allows values as strings (or, in OWL, the type xsd:string) as it is a data property. Similarly, the properties “is the son of” and “works in” allow values as instances of the type “Person” and “Place of Work”, respectively; both types are concepts described in the ontology.

The first of the techniques we used as similarity functions is the edit distance or Levenshtein's distance (Shvaiko & Euzenat, 2005). This distance receives two chains of characters as input and computes the distance between the strings, which is provided by the minimum number of character insertions, eliminations or substitutions as needed to transform a string into another. The edit distance is normalized. The greater the editing distance, the smallest the similarity between the chains of characters is. Thus, we declare a function of similarity as:

Equation 1. Similarity function with normalized edit distance:

$$F(a, b) = 1 - \frac{\text{editDistance}(a, b)}{\max(\text{length}(a), \text{length}(b))}$$

The edit distance is used to compare concept names and property names. We also calculated the similarity between the property types. The similarity function applied to property names is identical to that applied to the name of object and data type properties and, although the similarity function applied to the types of properties is different for the two. Should the properties have primitive data types as a value (round, floating point, etc.) that is identical, they will then be given value for similarity 1. Otherwise, they will be given value for similarity 0. If the properties are relationships between concepts (object property), we then compare the degree of similarity for the
property type by calculating the edit distance between the domains of these properties. We then have it that, if two properties relate to each other with concepts that are similar, then these properties have a certain degree of similarity and consequently the two concepts that have these properties also have a certain degree of similarity.

To give an example of the above two similarity functions consider the concepts “Vehicle” from ontology A and “Car” from ontology B. Concept “Vehicle” has a data type property named “Year of Manufacture” that receives integer values (xsd:int in RDF notation) and an object type property named “has” that relates to the concept “Wheel”. In its turn, the concept “Car” has a data type property named “Date of Manufacture” that receives string values (xsd:string in RDF notation) and an object type property named “contains” that also relates to a concept named “Wheel”, as shown in Figure 5. We can calculate the similarity function for property name analyzing “Year of Manufacture” and “Date of Manufacture”. When applying the edit distance to the name of these properties we will find value \(1 - 4 / 19 = 0.79\); see Equation 1. When applying the similarity function for property type also to “Year of Manufacture” and “Date of Manufacture”, we will find value zero (different data types). The same similarity function when applied to object type properties “has” and “contains” will produce value 1 as it is the return of the edit distance between the two concepts Wheel.

The hierarchy of the concepts is used when comparing their children and parents, i.e., two non-leaf concepts are structurally similar if the set of their immediate children is highly similar. The same idea is also used for the immediate parents of the concepts. This similarity function of hierarchy analyzes the context of the concept, that is, to calculate the degree of similarity of two concepts A and B, it is necessary to calculate the degree of similarity of their immediate parents. In this function, concepts that exist in the leaves, that is, concepts that have no children (subclasses) are computed with a degree of similarity 1. Similarly, root concepts, that is, concepts that have no superclasses, are computed with a degree of similarity equal to 1.

The total similarity degree of two concepts is calculated through equation below as proposed in (Souza, 1986). The equation is used to assess the similarity function \(F_\sigma\) as the summation of the \(m\) similarity functions \(F_\sigma\) applied to the pairs of ontological elements “a” and “b” with their associated weights \(W_\sigma\).

Equation 2 – Summation of the similarity functions:

\[
F_\sigma(a, b) = \sum_{\sigma=1}^{m} F_\sigma(a, b) \times W_\sigma
\]

Figure 6 shows the similarity functions (in the boxes) and the standard weights used in the comparison between the concepts. The weights are used to adjust the algorithm and are normalized to sum 1. The similarity functions return 1 to a perfect similarity and a smaller positive value for pairs with smaller similarities. The equation above is used to aggregate the information on the sub-similarities from the bottom up in the tree shown in the figure below, until the root. That is, the final similarity between two distinct ontology concepts is provided by the sum of their similarity functions. The weights \(W\) as described in the figure will be discussed in “Evaluation” section.

Suppose that two concepts under comparison have one single property and that similarity functions “Name” and “Type” have produced 0.5 and 1, respectively. This information will contribute to the result of the similarity function “Property” with weights 0.6 and 0.4, respectively. Thus, the similarity function “Property” will be calculated as: 0.5*0.6 + 1 * 0.4 = 0.7.

For the cases where more than one related element exists, to be used in the similarity function, that is, concepts that have several properties or concepts with more than a sub-class, the average from the result of the functions is used in the cases where both sides have the same number of elements. If some information is
missing in one of the ontologies, for example, if the number of properties that the two concepts under comparison have is different, we use a technique to produce an average of the weights. The goal is to introduce a penalty (equivalent to a negative weight) that is calculated as follows, where \( L_{\text{min}} \) and \( L_{\text{max}} \) represent the minimum and maximum number of concept properties (or children or parents of the concept), respectively:

**Equation 3** – Similarity function applied in the hierarchy and properties:

\[
\rho = \frac{\sum_{k=0}^{n} F_k(a,b)}{L_{\text{min}} + \text{Penalty} \times (L_{\text{max}} - L_{\text{min}})}
\]

The penalty ranges from 0 (when the difference in the number of elements of a concept
is not important, resulting into a simple arithmetical average) to 1 (when the difference in the number of elements is important).

When analyzing the result of the function applied to two concepts where each concept has more than one property or sub-class, we were faced with the problem of choosing which relationships should be chosen as relevant. For example, imagine concepts A and B, with respective subclasses [a1, a2] and [b1, b2, b3]. The similarity function will be applied to the entire subclass combination to generate, for example, a matrix such as that shown in Equation 4, ordered by degree of similarity:

Equation 4. Matrix with combinations of similarity (Souza, 1986):

\[
\begin{array}{ccc}
  a1 & b2 & 0.8 \\
  a2 & b2 & 0.7 \\
  a1 & b3 & 0.6 \\
  a2 & b1 & 0.4 \\
  a1 & b1 & 0.3 \\
  a2 & b3 & 0.3 \\
\end{array}
\]

The return of this function can happen through the choice of the first combination found, without repeating elements, where we would have [a1, b2, 0.8] and [a2, b1, 0.4] totalling a 1.20 similarity. An algorithm that checks all the combinations to choose the best response possible would select combinations [a2, b2, 0.7] and [a1, b3, 0.6] totalling 1.30. Even with some cases producing a lower value, our algorithm uses the first approach, as (1) our main concern is to find the biggest similarities between the concepts and find a very similar element is more interesting than to find two less similar elements, even if the sum total of the functions is bigger; apart from that, (2) to find the response that produces the highest total value is a NP-complete problem (Souza, 1986) and can prove to be impractical to calculate the response when the quantity of elements is too high.

After the calculation of the similarities, the concepts that meet the equation below are listed to the user as highly similar. In the evaluation described in “Evaluation” section, we consider 0.65 as the value for \( \Omega \).

Equation 5 – Rule to select highly similar concepts

\[
\sum_{i=1}^{m} F_i (a, b) \times W_i \geq \Omega
\]

**EVALUATION**

To assess this approach, three distinct comparisons are carried out. In the first one a section of the ontology is compared, that describes the domain of object-oriented programming with a copy of it that had the names of the concepts arbitrarily altered. Thus, we mean to assess if the algorithm works in the case where the two ontologies have identical structures and different names. The second comparison will be made through the alteration of the structure of the second ontology and with almost identical concept names. Finally, the third comparison will be made again via the alteration of the names of the second ontology, in an arbitrary fashion, thus representing large conceptualization differences between the two ontologies. This last is the worst-case scenario the algorithm may work in. Because of limitation of space in this paper, the evaluation process is summarized in this section. For more information about the evaluation process, see (Souza et al., 2009). All the weights applied to the similarity functions as shown in Figure 6 were arbitrarily set. Possible solutions for the definition of the function weights are discussed in “Conclusions” section.

In our first scenario we compared ontologies with same structure (properties and hierarchy), containing only differences between the names of the concepts that were arbitrarily altered.

The window below (Figure 7) shows the list of all the concepts with higher degrees of similarity calculated for the ObjectReference concept, above the minimum limit of 0.65 (threshold field) and ordered by degree of similarity. The
concept with the highest degree of similarity found was C3, with an approximate value of 0.805. When analyzing the ontologies (Souza et al., 2009), we found that this concept actually corresponds to the ObjectReference concept. The other concepts shown in the list of those most similar to ObjectReference are C1 and C11 with approximate values at 0.70 and 0.67, respectively. C1 corresponds to the DataType concept (ObjectReference parent) and C11 to the Primitive concept (brother to ObjectReference), that is, close concepts in their structure (both in hierarchical as in inherited properties).

Table 1 describes the results of the algorithm with the values found for each two concepts. The table was assembled with each concept of the ontology A and with the most similar concept as pointed by the algorithm. It is possible to see that when we recovered the concepts that were the most similar to each concept of the first ontology, the algorithm is correct in all the cases.

In the algorithm implementation made it is possible to differentiate the values of the main similarity functions applied. The similarity between the two concepts was not total (value 1) due to their difference between the terms. Such difference ends up by influencing, to a certain degree, the similarity functions applied on the hierarchy, on non-hierarchical data and, clearly, the degree of total similarity. On the other hand, these similarity values have values above those considered as acceptable (lower limit of 0.65) due to the high similarity in their structures, which can be more clearly seen in the value 1 returned by the similarity function when applied to concept properties.

This case of clear difference between the names of the two ontologies can be solved when we alter the value for the weights applied by the algorithm to the similarity functions. When applying value 0 (zero) to the similarity functions applied to the name of the concepts and to the name of the properties, we will have in this case more real similarity values. The same adaptation was made in the second scenario (comparison of ontologies with concept names almost identical and its structure slightly altered when we arbitrarily modified some properties). In this scenario, after reducing the weight value to the similarity functions applied to the properties and the hierarchy, the resulting similarity became very high. See (Souza et al., 2009) for more details.

When ontologies became more complex, that is, the ontologies have several differences in the its structure and terms, the resulting similarity decreases substantially. Although, the algorithm shows correct similarity in most of the cases. The concepts that were not duly found by the algorithm correspond to those that have greater divergence regarding their conceptualization. Such concepts are indeed highly complex to identify and human intervention is vital.

**RELATED WORK**

Multi-agent systems that use negotiation concepts are proposed in the literature (Chen et al., 1999; Collins & Gini, 2008; Huang & Sycara, 2002), although each agent has the capacity to consult a single, global domain ontology to avoid communication problems, which makes these approaches applicable only to environments that are more controlled than the Web.

A negotiation agent is considered an agent capable of exchanging offers to reach reasonable consensus to the parties. However, systems that use negotiation agents are usually systems that aim at the object of the negotiation, be it the acquisition of a product or a mutual action that will be implemented.

On the other hand, agents for the negotiation of meaning have been proposed in the literature (Packer et al., 2009; Aschoff et al., 2004; Elst & Abecker, 2002; Tijerino et al., 2004), where the goal of the agent is to solve conflicts with the understanding of concepts. This scenario also differs from this work, for the same reason mentioned earlier, as the goal of these agents is the object of the negotiation. This work, in its turn, uses the negotiation capacity of agents independently from its goals. The negotiation of meanings is part only of the agent commu-
nica tion process, ensuring an understanding of the concepts involved in the message received by the agent.

In the articles mentioned in the previous paragraph, the negotiation of meanings is made to achieve the compatibility of domain ontologies as evolved by human users. In Aschoff et al. (2004) and Elst and Abecker (2002), negotiation performatives are defined and the negotiation protocol, as in our work, describes the process through which agents can communicate on the meaning of the concepts. However, the need of human intervention in the negotiation process of the agents is a point of difference in the application scenario that is described at that work. Apart from that, our proposal extends an architecture for the construction of agents (Truszkowski, 2006) and introduces sub-modules into the communication module to deal with concepts that are typical to negotiation with the objective of allowing the process of communication between agents to be made without the interference in the intention of the agent. Apart from that, the intention is to see this proposal used by agents that communicate between themselves in open environments such as the Web, where constant human intervention is not reasonable.

Approaches to calculate similarity between concepts have been studied by several scholars (Euzenat & Shvaiko, 2007). Some proposals try to solve the problem with the use of semantic comparisons (Bouquet et al., 2003) or syntactic ones. Our approach uses different similarity functions in a set. The main contribution of this proposal is in the separation of the similarity functions in each group of elements of the ontology in a recursive manner over the concepts.

The mapping of ontologies based in instances is a promising set of solutions of the class problems to align ontologies. Some solutions were proposed, using this approach (Breitman et al., 2008; Issac & Meiji, 2008; Todorov & Giebel, 2008). The use of the ABox to calculate similarity between concepts, however, presents problems that were not dealt with in this article. One of the issues with using instances lies with the finding this set of instances, as the ABox can be described both as instances in OWL in the ontology itself as instances in other OWL files, as annotations in a HTML page, text documents, etc. Currently, our ontology alignment approach does not receive instances as input. However, the tree of functions shown in Figure 6 can be modified to receive new similarity functions, which will be added to Equation 2. Indeed, the implementation of the algorithm to evaluate the agent architecture proposed makes provision for new similarity functions to be added with the implementation of a common interface.

CONCLUSION

Two active (and related) research topics in the Semantic Web are (1) finding solutions for the semantic interoperability between the growing number of ontologies that, one imagines, will exponentially increase as the area matures, and (2) allowing agents to communicate with other agents using distinct domain ontologies. This work contributes to these two research topics when proposing a negotiation approach to assist in obtaining consensus during the harmonization of knowledge structures.

The work presented an extension to the Goddard agent architecture to carry out a treatment of the terms defined in ontologies as received during the communication between agents that use distinct domain ontologies. The treatment of the terms is made through a process of negotiation of meanings that results in an alignment between the concepts of the ontologies. The proposed process of negotiation contains elements of negotiation found in real-world negotiations and that were previously applied and tested in the negotiation of meanings (Oliveira et al., 2007; Souza et al., 2006) in a scenario of negotiation with humans.

Together to agent architecture, this article proposed a structural approach for the alignment of ontologies using hierarchical, lexical and relational comparisons between the concepts of the ontologies. The algorithm presented carries out a recursive calculation on the concepts of the ontologies where the partial function for the
Figure 7. Most similar concepts of ontology A#ObjectReference

Table 1. Degree of similarity between the most similar concepts

<table>
<thead>
<tr>
<th>Ontology A</th>
<th>Ontology B</th>
<th>Degree of similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datatype</td>
<td>C1</td>
<td>0.85</td>
</tr>
<tr>
<td>Data</td>
<td>C2</td>
<td>0.85</td>
</tr>
<tr>
<td>ObjectReference</td>
<td>C3</td>
<td>0.805</td>
</tr>
<tr>
<td>Class</td>
<td>C4</td>
<td>0.711</td>
</tr>
<tr>
<td>Inner_Class</td>
<td>C5</td>
<td>0.869</td>
</tr>
<tr>
<td>Subclass</td>
<td>C6</td>
<td>0.823</td>
</tr>
<tr>
<td>Superclass</td>
<td>C7</td>
<td>0.864</td>
</tr>
<tr>
<td>AbstractClass</td>
<td>C8</td>
<td>0.761</td>
</tr>
</tbody>
</table>

The study presented the validation of the algorithm set weights for each similarity function and a penalty applied to similarity functions used in hierarchical and relational comparisons (properties). The weights were arbitrarily set. The calibration issue is a problem that will be dealt with in future studies of this approach in order to cut the time spent in arbitrarily testing different weights.

As future work, an analysis of different methods for the alignment of ontologies that could be better applied during the communication between the agents is suggested, considering similarity calculations that are more specific to the structure of ontologies (that is, considering concepts, properties and rules, as well as pos-
sibly instances), to allow better results that can provide a more significant reserve value to the negotiations using a theoretical framework to evaluate algorithms in the analysis of consensus algorithms for multi-agent networks such as the one proposed in (Olfati-Saber et al., 2007). Currently, an ontology matching system is at development in Ontology Reasoning module (Figure 4). This system will allow to associate different alignment methods from Strategy Planning module (Figure 4) and automatically choose which set of methods and weights are more applicable for a given structure.

REFERENCES


Copyright © 2010, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.


Jairo Francisco de Souza is an Assistant Professor at the Department of Computer Science, Federal University of the Juiz de Fora (UFJF), Brazil, where he teaches courses in Software Engineering and Semantic Web. He holds a M.Sc. (2007) from Federal University of Rio de Janeiro (UFRJ) and he is a Ph.D. candidate in the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil, both in Computer Science area. His research interests include knowledge representation; semantic web; ontologies; information integration; and semantic models.
Rubens N. Melo is a Senior Professor and Researcher in the field of Databases at the Computer Science Department of the Pontifical Catholic University of Rio de Janeiro, PUC-Rio. He holds a B.Sc. in Electronic Engineering (1968), a M.Sc. (1971) and a Ph.D. (1976) degree in Computer Science from the Air Force Institute of Technology (ITA) in Sao Paulo. Currently, he is Associate Professor at PUC-Rio where he leads the Database Research Lab (TecBD). One of his current interests is the application of Database Technology to distance learning. In this field he has also served as Director of the Centre of Distance Education under the Vice-Rectory for Academic Affairs of PUC-Rio.

Jonice Oliveira got her B.Sc. degree in Computer Science (2001) at Federal University of Rio de Janeiro and the M.Sc. degree (2003) at the Department of Computer and Systems Engineering (PESC) on Graduate School and Research in Engineering (COPPE). Also, she obtained her Ph.D. (2007) and her Post-Doctoral (2008) at PESC/COPPE. Now, she is Assistant Professor (in Portuguese: Professora Adjunta) on Federal University of Rio de Janeiro, at Computing Science Department (DCC) of Mathematical Institute (IM). She is researching ways of efficiently manage scientific knowledge and improve knowledge exchange in scientific and academic environments. We can mention as her research interests: Knowledge Management (main area), Social Network Analysis (current focus), Knowledge Visualization, Competitive Intelligence, Customer Relationship Management, e-Science, Ontology, Web Semantic and Autonomic Computing. In 2006, she received an IBM Ph.D. Fellowship Award on Autonomic Computing.

Jano de Souza is a professor of Computer Science at the Graduate School of Engineering (COPPE) of the Federal University of Rio de Janeiro (UFRJ) (Brazil). His area of specialization is Databases, and he is involved in research in fields such as CSCW, Decision Support Systems, Knowledge Management and GIS. Professor de Souza received his bachelor’s degree in Mechanical Engineering (1974) and his master’s degree (1978) in System Engineering from the Federal University of Rio de Janeiro (Brazil), and his Ph.D. in Information Systems (1986) from the University of East Anglia (England). He has published more than 300 papers in Journals and Conferences Proceedings and supervised around sixty Ph.D Thesis and M.Sc. dissertations. Prof. de Souza has strong involvement with industry and has consulted to many private companies and government agencies and ministries.

Sean Wolfgang Matsui Siqueira is an Assistant Professor at the Department of Applied Informatics, Federal University of the State of Rio de Janeiro (UNIRIO), Brazil, where he teaches courses in Databases, Information Systems and Semantic Web. He holds a M.Sc. (1999) and a Ph.D. (2005) in Computer Science, both from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil. His research interests include knowledge representation; semantic web; ontologies; information integration; semantic models; user models; e-learning; social web; and music information systems. He has experience in the Computer Science area, with focus on Information Systems and Technology Enhanced Learning. He has participated in some international research projects and has published more than 70 papers for conferences, journals, and books.