Ontology Alignment as a Basis for Mobile Service Integration and Invocation

Jingshan Huang[1]
Jiangbo Dang
Michael N. Huhns
Computer Science and Engineering Department
University of South Carolina
Columbia, SC 29208, USA
{*huang27, dangj, huhns*}*@engr.sc.edu*

Yongzhen Shao
Software School
Fudan Unversity
Shanghai 200433, China
shaoyz@fudan.edu.cn

Received: December 31 2005; revised: May 07 2006

***Abstract*— The limited capabilities of typical mobile devices can be extended by using services from other devices. To use such services, a mobile device must be able to comprehend their descriptions. Ontologies can aid in this comprehension, but ontologies designed independently for each device would have heterogeneous semantics. This paper presents an automated schema-based approach to align the ontologies from interacting devices as a basis for mobile service invocation. When the ontologies are ambiguous about the services provided, we introduce compatibility vectors as a means of maintaining ontology quality and deciding which service to choose to reduce the ambiguity. Our approach is verified both experimentally and theoretically.**

***Index Terms*— Ontology Merging, Ontology Compatibility, Mobile Service**

I. INTRODUCTION

Mobile computing is becoming more widespread and increasingly important. Mobile portable devices already outnumber traditional desktop computers and are expected to determine the view of computers for future generations. However, mobile devices typically have rather limited capabilities. To overcome this limitation, a mobile device can make use of the functionalities and services provided by other mobile devices, and thereby extend its own capabilities. The first step for mobile devices to achieve this goal will be to understand the descriptions of services that they can provide to each other. Only in this way can the future integration and/or invocation of mobile services take place automatically and successfully.

An ontology serves as a declarative model for the knowledge and capabilities possessed by a device or of interest to a device. It forms the foundation upon which machine understandable description of services can be obtained, and as a result, automatic interaction among mobile devices is enabled. Therefore, adding ontologies into the mobile service scenario will facilitate the extension of mobile device capabilities by providing a more comprehensible and formal semantics. The functionalities and behaviors of mobile services can be described, advertised, discovered, and composed by others through the use of and reference to ontologies. Eventually, these mobile services would be able to interoperate with each other, even though they have not been designed to work together. This is the vision for pervasive computing among mobile devices.

However, because it is impractical to have a global ontology that describes every concept that is or might be included as part of the mobile services, ontologies from different mobile devices typically have heterogeneous semantics. Due to this basic characteristic, mobile devices need to reconcile ontologies and form a mutual understanding when they interact with each other. Only in this sense are mobile services able to comprehend and/or integrate the information from different sources, and enhance service interoperability thereafter.

In this paper, we first present an automated schema-based ontology merging algorithm to align heterogeneous ontologies. Then we focus on an important but mostly neglected research topic - how a mobile device can select suitable ontologies to interact with. We introduce the concept of compatibility vectors as a means of evaluating and maintaining ontology quality, and use this as a basis for suitability of ontology selection. Our approach is able to create and adjust dynamically the compatibilities of mobile devices with regard to the quality

[1]Corresponding author Tel./Fax: +1-803-777-3768/+1-803-777-3767.

and use ontologies according to its own conceptual view of the world. Consequently, ontological heterogeneity among different mobile devices becomes an inherent characteristic in a mobile computing environment. The heterogeneity can occur in two ways: (1) Different ontologies could use different terminologies to describe the same conceptual model. That is, different terms could be used for the same concept or the same term could be used for different concepts. (2) Even if two ontologies use the same name for a certain concept C, its corresponding properties and relationships with other concepts can be different.

Therefore, two major problems are envisioned here. First, during the formation of a MSC, how can it be ensured that all devices within the community have no problem in understanding each other's ontology? Secondly, a mobile device seeking services from outside this community would like to choose those devices that understand its ontology best. How can it ensure this selection is a correct one?

C. Overview of Our Approach

In order to solve the first problem - mutual understanding of ontologies within a MSC, we need an approach to match/align ontologies from different mobile devices. By this means, concepts from communicating devices can comprehend each other, and possible integration of related services can be achieved. In the next section, we present an ontology merging algorithm to reconcile heterogeneous ontologies.

To tackle the second problem - the correct selection of mobile devices that are most acquainted with the ontologies from service-consuming devices, we introduce compatibility vectors as a means of measuring and maintaining the ontology quality. By setting up the compatibility for each mobile device along with the formation of a MSC, not only the mobile devices seeking service from this community are able to select the best service provider(s) with ease, but also a better mutual understanding of ontologies within the MSC is obtained.

IV. A SCHEMA-BASED ONTOLOGY MERGING ALGORITHM

Our goal is to construct a merged ontology from two original ones. Although there does not exist such a global and agreed-upon ontology, we do assume that there is a common metamodel, i.e., OWL DL, for the ontologies to be merged, and we also assume that natural language provides common semantics during the ontology merging process.

A. Top-Level Procedure

The ontology merging is carried out at the schema level, that is, we concentrate on the structure (schema) information of ontologies. Internally we represent an ontology using a directed acyclic graph *G* (*V*, *E*), where *V* is a set of ontology concepts (nodes), and *E* is a set of edges between two concepts, i.e., *E* = {(*u*,*v*)|*u* and *v* belong to *V*, and *u* is a *superclass* of *v*}. In addition, we assume that all ontologies share "Thing" as a common "built-in" root. In order to merge ontologies, G_1 and G_2, we try to relocate each concept (node) from one ontology into the other one. We adopt a breadth-first order to traverse G_1 and pick up a concept *C* as the target to be relocated into G_2. Consequently, at least one member of *C*'s parent set Parent(*C*) in the original graph G_1 has already been put into the suitable place in the destination graph G_2 before the relocation of *C* itself. The pseudocode below describes this top-level procedure, whose time complexity is obviously *O(n^2)*, with *n* the number of concepts in the merged ontology.

Input: Ontology G_1 and G_2
Output: Merged Ontology G_2
begin
  new location of G_1's root = G_2's root
  for each node *C* (except for the root) in G_1
    Parent(*C*) = *C*'s parent set in G_1
    for each member p_i in Parent(*C*)
      p_j = new location of p_i in G_2
      relocate(*C*, p_j)
    end for
  end for
end

Top-Level Procedure - *merge***(**G_1, G_2**)**

An implementation detail is worth mentioning here. Because of the characteristics of traversing a directed acyclic graph, there is a possibility that one or more parents of a certain concept may not have been relocated before that concept is visited. However, at least one of the parents will have been relocated. In this case, we revisit the target concept after all its parents have been visited. Notice that progress is guaranteed, because the graphs in question are acyclic.

B. Relocate Function

The relocate function in the top-level procedure is used to relocate *C* into a subgraph rooted by p_j. The following pseudocode shows the details of this function.

Input: nodes N_1 and N_2
Output: the modified structure of N_2 according to information from N_1
begin
  if there exists any *equivalentclass* of N_1 in the child(ren) of N_2
    merge N_1 with it
  else if there exists any *subclass* of N_1 in the child(ren) of N_2
    Children(N_1) = set of such subclass(es)
    for each member c_i in Children(N_1)
      add links from N_2 to N_1 and from N_1 to c_i
      remove the link from N_2 to c_i
    end for
  else if there exists any *superclass* of N_1 in the child(ren) of N_2

assume that both P_1 and P_2 have ten properties. If there are already nine pairs with a total-match, and furthermore, if we find out that the names in the remaining pair of properties are similar with each other, then it is much more likely that this pair will also have a match, as opposed to the case where only one or two out of ten pairs have a total-match.

2) *name-match*
- The linguistic matching of the property names results in either an exact-match, or a leading-match with $v_{linguistic} \geq$ *threshold*; but
- The data types do not match.

Let v_n = number of properties with a name-match, and f_n=$\frac{v_t+v_n}{n_1}$. Similarly to f_t, f_n also serves as a correcting factor for *datatype-match*.

3) *datatype-match*
Only the data types match. Let v_d = number of properties with a datatype-match.

After we find all the possible matching models in the above order, we can calculate the similarity between the property lists as $\frac{1}{n_1}(v_t \cdot w_1 + v_n(w_2 + f_t \cdot w'_2) + v_d(w_3 + f_n \cdot w'_3))$, where:

- w_i (*i* from 1 to 3) is the weight assigned to each matching model; and
- w'_i (*i* from 2 to 3) is the correcting weight assigned to the matching models of name-match and datatype-match.

E. Domain-Independent Reasoning

Remember that to merge two ontologies, we in essence are to relocate each concept from one ontology into the other one. After we obtain the linguistic and contextual similarities, we apply a domain-independent reasoning rule to infer the relationship between the target concept to be relocated and the candidate concept in the destination ontology.

1) Relationships among Property Lists: Suppose we have two ontologies A and B, each of which is designed according to the OWL DL specification. Furthermore, let *n(A)* and *n(B)* be the sets of concepts in A and B, respectively, with n_i*(A)* and n_j*(B)* be the individual concept for each set ($1 \leq i \leq |n(A)|$ and $1 \leq j \leq |n(B)|$), and P(n_i*(A)*) and P(n_j*(B)*) be the property list for each individual concept.

Consider the property lists P(n_i*(A)*) and P(n_j*(B)*), let s_i and s_j be the set size of these two lists. There are four mutually exclusive possibilities for the relationship between P(n_i*(A)*) and P(n_j*(B)*):

- P(n_i*(A)*) and P(n_j*(B)*) are consistent with each other if and only if

  i. Either $s_i = s_j$ or $\frac{abs(s_i - s_j)}{s_i + s_j} \leq$ *threshold*, and

  ii. $v_{contextual} \geq$ *threshold*

  We denote the corresponding concepts n_i*(A)* and n_j*(B)* by n_i*(A)* $\overset{p}{\leftrightarrow}$ n_j*(B)*;

- P(n_i*(A)*) is a subset of P(n_j*(B)*) if and only if

  i. $s_i \leq s_j$, and
  ii. $v_{contextual} \geq$ *threshold*

  We denote the corresponding concepts n_i*(A)* and n_j*(B)* by n_i*(A)* $\overset{p}{\rightarrow}$ n_j*(B)*;

- P(n_i*(A)*) is a superset of P(n_j*(B)*) if and only if

  i. $s_i \geq s_j$, and
  ii. $v_{contextual} \geq$ *threshold*

  We denote the corresponding concepts n_i*(A)* and n_j*(B)* by n_i*(A)* $\overset{p}{\leftarrow}$ n_j*(B)*;

- Other.

2) Relationships among Concepts: Given any two ontology concepts, we can have the following five mutually exclusive relationships between them:
- *subclass*, denoted by $\subseteq$
- *superclass*, denoted by $\supseteq$
- *equivalentclass*, denoted by $\equiv$
- *sibling*, denoted by $\approx$
- *other*, denoted by $\neq$

3) Reasoning Rule: If two classes share a same parent, then their relationship is one of: *equivalentclass*, *superclass*, *subclass*, and *sibling*.

- Preconditions:
n_{i1}*(A)* $\supseteq$ n_{i2}*(A)* and
n_{j1}*(B)* $\supseteq$ n_{j2}*(B)* and
n_{i1}*(A)* $\equiv$ n_{j1}*(B)* and
1) (the names of n_{i2}*(A)* and n_{j2}*(B)* have either an exact-match, or a leading-match with $v_{linguistic} \geq$ *threshold*) and n_{i2}*(A)* $\overset{p}{\leftrightarrow}$ n_{j2}*(B)*
2) (the name of n_{j2}*(B)* is a sub-match of the name of n_{i2}*(A)*) and n_{i2}*(A)* $\overset{p}{\rightarrow}$ n_{j2}*(B)*
3) (the name of n_{j2}*(B)* is a super-match of the name of n_{i2}*(A)*) and n_{i2}*(A)* $\overset{p}{\leftarrow}$ n_{j2}*(B)*
4) None of above three holds

- Conclusion:
1) n_{i2}*(A)* $\equiv$ n_{j2}*(B)*
2) n_{i2}*(A)* $\supseteq$ n_{j2}*(B)*
3) n_{i2}*(A)* $\subseteq$ n_{j2}*(B)*
4) n_{i2}*(A)* $\approx$ n_{j2}*(B)*

The intuition behind our reasoning rule is as follows. After the linguistic matching phase we obtain three candidate lists for target concept *C*. For each concept *C'* in these lists, we then try to find the contextual similarity between *C* and *C'* to make the final decision.

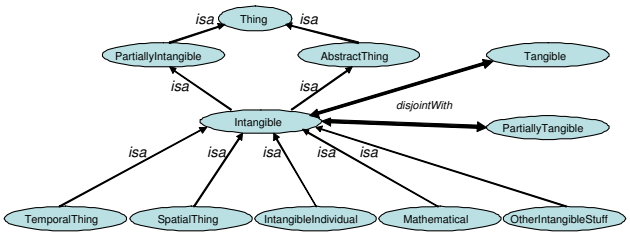


Fig. 2. Graphical Representation for $Center_1$

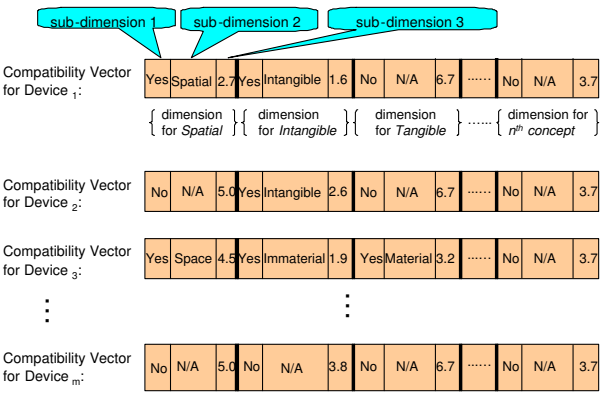


Fig. 3. Compatibility Vectors

the ontology distance between the original ontology and the center one: $D = \sum_{i=1}^{n} (w_i \cdot d_i)$, where d_i is the distance between a pair of concepts, and n is the number of concepts in the center ontology.

Recall that the concept set of the original ontology is a subset of that of the center ontology, and the concept distance is encoded by the missing relationships in the original ontology compared to the center one. The above formula shows that the ontology distance is obtained by the weighted sum of the concept distances between two ontologies. How much a concept contributes to the ontology distance is determined by the importance of that concept in its ontology. We use the percentage of the number of relationships to represent this measurement. For example, if $ontology_1$ has 100 relationships in total, and concept "Spatial" has 15 relationships, then the weight for this concept in $ontology_1$ is 0.15.

3) Compatibility Vectors: Inside the center ontology, there is a set of compatibility vectors, one for each original ontology. A compatibility vector consists of a set of domains, each corresponding to one concept in the center ontology. Therefore, all compatibility vectors have identical dimension, i.e., equaling to the number of the concepts in the center ontology. Each dimension has three sub-dimensions. The first sub-dimension tells us whether the original ontology understands this concept or not; the second sub-dimension records the concept name in the original ontology if the latter does recognize that concept; and the third sub-dimension encodes the distance from the concept of the original ontology to the concept of the center ontology. An example of compatibility vectors is shown in Figure 3.

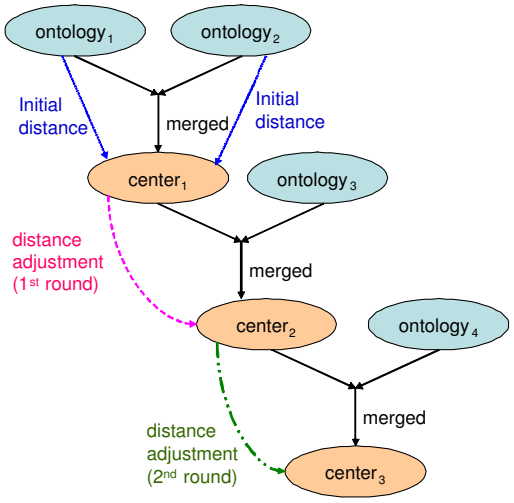


Fig. 4. Dynamic Adjustment of Compatibility Vectors

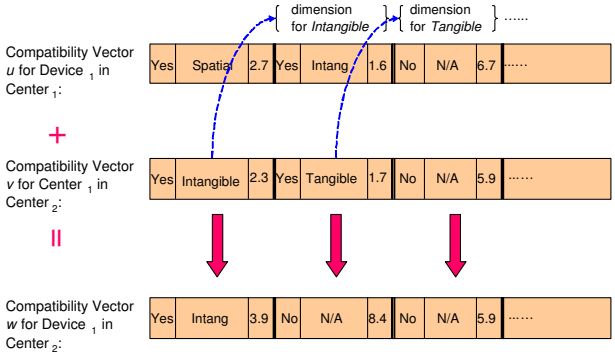


Fig. 5. Example of New Vector Generation

For the first concept ("Spatial") in the center ontology, $device_1$ knows it as "Spatial" and has a concept distance of 2.7; $device_3$ also understands this concept, but with a different name ("Space") and a bigger concept distance of 4.5; neither $device_2$ nor $device_m$ recognizes concept "Spatial", therefore, they have the same concept distance (5.0).

C. Dynamic Adjustment of Compatibility Vectors

As mentioned before, when there is only one mobile device, its compatibility is perfect. In the compatibility vectors stored in the center ontology, each concept distance has a value of zero. However, with the adding of new devices into this MSC, the compatibilities for existing devices might be changed because newly joined devices could contain more accurate and/or complete information regarding the ontology in the same domain.

An example is shown in Figure 4, demonstrating the process of dynamic distance adjustment. After $ontology_1$ and $ontology_2$ are merged to generate $center_1$, the distance between these two original ontologies and the merged one ($center_1$) is calculated and stored in compatibility vectors of $center_1$. Upon the joining of $ontology_3$ and the generation of $center_2$, the distance from $center_1$ to $center_2$ is calculated

To record and maintain the proper compatibility of each device inside a MSC, the key is to obtain a correct center ontology by which to evaluate the distance from it to each original ontology, and thereby acquire the corresponding compatibility vector. When a new device and its associated ontology join the MSC, instead of communicating with each existing device, it only talks with the center ontology. Therefore, if we can prove that the newly merged ontology is a correct new center ontology, the correctness of compatibility vectors is guaranteed.

First, we point out that according to the merging algorithm in this paper, each time we merge two ontologies, the resultant one will contain all information from both original ones. Next, we introduce Lemma 1 and Theorem 1.

Lemma 1. When we merge two ontologies A and B using the algorithm in Section IV, the result is the same regardless of whether we merge A into B or merge B into A.

Proof by induction.

1) Base Case: when both A and B contain two concepts, i.e., besides one common built-in root, "Thing", A contains C_1 and B contains C_2.
   If we merge A into B according to the Top-Level Merging Procedure in Section IV, "Thing" in A is considered equivalent with "Thing" in B; then C_1 is compared with all the direct children of the root in B, in this case C_2, to determine where to put C_1 in B. This is based on the relocate function inside the Top-Level Merging Procedure. On the contrary, if we merge B into A, "Thing" in B is considered equivalent with "Thing" in A; then C_2 is compared with C_1 to determine where to put C_2 in A. Obviously, we obtain the same merged ontology in both cases.
2) Induction: Assume that Lemma 1 holds for all cases where the number of concepts contained in A and B is less than (i+1) and (j+1), respectively. Now consider the case where A and B contain (i+1) and (j+1) concepts, respectively.
   Suppose the superclass set of the $(i+1)^{th}$ concept in A, C_{i+1}, is $P_A(C_{i+1})$, and suppose the location of $P_A(C_{i+1})$ in merged ontology M is $P_M(C_{i+1})$. The position of C_{i+1} in M is determined by the relationships between C_{i+1} and all the direct children of $P_M(C_{i+1})$. From the inductive hypothesis we know that $P_M(C_{i+1})$ is identical no matter whether we merge A into B or merge B into A. Therefore, the position of C_{i+1} in M will also be the same in both situations. That is, C_{i+1}, the $(i+1)^{th}$ concept in A, will be put into the same position in M in both merging orders. Similarly, the $(j+1)^{th}$ concept in B will also be put into the same position in M in both merging orders. So in the case where A and B contain (i+1) and (j+1) concepts, respectively, we still have the same resultant ontology regardless of the merging order taken.

Theorem 1. The final result of merging a number of ontologies is identical no matter by which order the original ontologies are merged using the algorithm in Section IV.

Proof by induction.

1) Base Case: there are two ontologies to be merged. According to Lemma 1, when we merge two ontologies A and B, the result is the same no matter whether we merge A into B, or merge B into A.
2) Induction: Assume that Theorem 1 holds for all cases where the number of ontologies to be merged is less than (n+1). Now consider the case where we merge (n+1) ontologies. Let the indexes of these ontologies be: 1, 2, ..., (n+1).
   Consider two arbitrary orders by which we merge these (n+1) ontologies: $order_1$ and $order_2$. Suppose the last index in $order_1$ and $order_2$ is i and j, respectively.
   - If i equals j, then the first n indexes in $order_1$ and $order_2$ are the same, just in different orders. We merge the first n ontologies to get $Merged_n$. According to the inductive hypothesis, $Merged_n$ in $order_1$ is identical with $Merged_n$ in $order_2$. Then we merge $Merged_n$ with the last ontology in both $order_1$ and $order_2$, and we will get the same result.
   - If i does not equal j, we mutate the first n indexes in $order_1$ and make the n^{th} index be j; then mutate the first n indexes in $order_2$ and make the n^{th} index be i. Now the first (n-1) indexes in $order_1$ and $order_2$ are in common (possibly in different orders), and the last two are (j, i) and (i, j), respectively. Notice that this kind of mutation will not affect the merging result of the first n ontologies according to our inductive hypothesis. We then merge the first (n-1) ontologies to get $Merged_{n-1}$. According to the hypothesis, $Merged_{n-1}$ in $order_1$ is identical with $Merged_{n-1}$ in $order_2$. Finally we merge $Merged_{n-1}$ with the last two ontologies in both $order_1$ and $order_2$, and we will get the same result.

2) Complexity of Compatibility Vectors - An Efficient Approach: The time complexity of establishing a MSC, along with the achievement of mutual understanding of ontological concepts, is in the order of $O(mn^2)$. For the ontology merging, $O(mn^2)$ is needed, because we need to merge *m* ontologies, and each merging procedure takes time $O(n^2)$ as described in Section IV. In addition, in order to dynamically update the compatibility vectors, extra time will be spent. According to the previous analysis, $O(nlogn)$ is needed for updating one device, so the time for extra work for all services is $O(mnlogn)$. Therefore, the total complexity becomes $O(m(n^2 + nlogn))$, which is in the same order of $O(mn^2)$.

For device selection, the time complexity is $O(n^2)$. We only need to compare the ontology from the service-consuming device with the center ontology.

VI. EXPERIMENTAL RESULTS

A. Test ontologies

A collection of sixteen ontologies for the domain of "Building" were constructed by graduate students in computer

future work includes (1) how to maintain compatibility vectors when existing mobile devices modify their corresponding ontologies, and (2) what kind of mechanism is suitable if we simultaneously consider qualities of both ontologies and services.

REFERENCES

[1] J. Huang, R. Zavala Gutiérrez, B. Mendoza, and M. N. Huhns. *Sharing Ontology Schema Information for Web Service Integration*, in: *Proceedings of 5th International Conference on Computer and Information Technology (CIT 2005)*, Shanghai, China, 2005.

[2] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. *Learning to match ontologies on the Semantic Web*, in: *The VLDB Journal*, Vol. 12. Springer-Verlag 303–319, 2003.

[3] H. H. Do, S. Melnik, E. Rahm. *Comparison of schema matching evaluations*, in: *Proceedings of workshop on Web and Databases*, 2002.

[4] J. Madhavan, P. A. Bernstein, and E. Rahm. *Generic Schema Matching with Cupid*, in: *Proceedings of the 27th VLDB Conference*, Springer-Verlag, 2001.

[5] N. F. Noy, M. A. Musen. *Anchor-PROMPT: Using Non-Local Context for Semantic Matching*, in: *Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*. Seattle, WA, 2001.

[6] S. Melnik, H. Garcia-Molina, and E. Rahm. 2002. *Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching*, in: *Proceedings of the 18th International Conference on Data Engineering*. IEEE Computer Society Press, 2002.

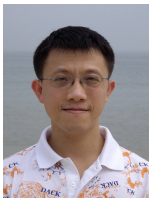
[7] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. *S-Match: an algorithm and an implementation of semantic matching*, in: *Proceedings of the 1st European Semantic Web Symposium*, Vol. 3053. Springer-Verlag 61–75, 2004.

[8] JWNL, "Java WordNet Library - JWNL 1.3", http://sourceforge.net/projects/jwordnet/, 2005.

[9] A. S. Bilgin. and M. P. Singh. *A DAML-based repository for QoS-aware semantic web service selection*, presented at *IEEE International Conference on Web Services*, 2004.

[10] C. Zhou, L.-T. Chia, and B. S. Lee. *DAML-QoS ontology for web services*, presented at *IEEE International Conference on Web Services*, 2004.

[11] S. Kalepu, S. Krishnaswamy, and S. W. Loke. *Reputation = f(user ranking, compliance, verity)*, presented at IEEE International Conference on Web Services, 2004.



Jingshan Huang is a Ph.D. candidate in Computer Science and Engineering Department at the University of South Carolina. Mr. Huang is a member of IEEE, AAAI, SIAM, and a review board member of Journal of Open Research on Information Systems (JORIS). He has served as a program committee member for several international conferences and is a technical paper reviewer for many journals and conferences. Mr. Huang's research interests include ontology matching/aligning, ontology quality, semantic integration, Web services, and service-oriented computing. He can be reached at huang27@sc.edu.

Jiangbo Dang earned his Ph.D. degree in computer science from Computer Science and Engineering Department at the University of South Carolina. Dr. Dang's research interests include distributed artificial intelligence and multiagent systems, service-oriented computing, business process and workflow management, and knowledge discovery, data mining, and machine learning. He can be reached at dangj@engr.sc.edu.

Michael N. Huhns is a professor of Computer Science and Engineering Department at the University of South Carolina, where he also directs the Center for Information Technology. Dr. Huhns is a Fellow of the IEEE and a member of Sigma Xi, Tau Beta Pi, Eta Kappa Nu, ACM, Upsilon Pi Epsilon, and AAAI. He is the author of over 200 technical papers and four books in machine intelligence, including the recently coauthored (with Prof. Munindar P. Singh of NCSU) textbook *Service-Oriented Computing: Semantics, Processes, Agents* for John Wiley Publishing Co. Dr. Huhns is on the editorial boards of seven journals and is a founder and board member for the International Foundation for Cooperative Information Systems and the International Foundation for Multiagent Systems. Hi sresearch interests include multiagent systems, service-orinted computing, and semantic Web services. He can be reached at huhns@sc.edu.

Yongzhen Shao is pursuing a Master's degree in the Institute of E-Business of Software School at Fudan University, China. Mr. Shao's research interests include E-Business, Web services, and ontology management. He can be reached via email: shaoyz@fudan.edu.cn.