Intelligent Automation and Soft Computing, Vol. X, No. X, pp. X-X, 1999

ONTOLOGY TOOLS FOR SEMANTIC RECONCILIATION IN DISTRIBUTED HETEROGENEOUS INFORMATION ENVIRONMENTS

KUHANANDHA MAHALINGAM AND MICHAEL N. HUHNS
CENTER FOR INFORMATION TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF SOUTH CAROLINA
COLUMBIA, SC 29208
huhns@sc.edu

ABSTRACT—This paper describes how information spaces stored as ontologies are best suited for semantic reconciliation. In addition, it takes a brief look at several advantages of using ontologies in a distributed, heterogeneous, and dynamic environment such as the Internet. It also examines the construction and evaluation of an ontology-based distributed information system developed using the Java language. It discusses issues related to software tools that operate in a distributed environment such as the Internet and how the client-server architecture and the agent technology used by the Java language can perform effectively in such environments. The software is applied to an information system for healthcare administrators, which spans hospitals, clinics, and governmental health departments.

Key Words: semantic reconciliation, ontology, Java, healthcare information systems, group editing, distributed database systems, agent technology.

1. INTRODUCTION

An approach to achieving interoperation among distributed and heterogeneous information sources is to introduce software agents to serve as mediators, translators, and information brokers—this is the essence of a cooperative information systems approach. The major task for the agents is to reconcile the varied semantics of the mostly autonomous resources. We have developed a tool for constructing and browsing the ontologies that can serve as a basis for semantic reconciliation. This paper describes the tool and its use in a representative application.

1.1 Use of Ontologies as a Basis for Semantic Reconciliation

A common problem for Internet users is how to search for and retrieve necessary information from the large number of information sources available. The information provided by the sources is not just simple text, but also includes multimedia, forms, structured data, and executable code—it has become much more complex than before. As a result, old methods for manipulating these sources are no longer appropriate. To keep pace with the growth of the Internet, mechanisms are needed that allow efficient querying on diverse information sources that support structured as well as unstructured information. Surprisingly, structured data has become more difficult to find and retrieve than unstructured text, because keyword searches over previously indexed documents, which work well for text, are unsuitable for data. Data retrieval requires schemas, which are often unavailable, incomplete, or incomprehensible. We believe that in a heterogeneous environment, ontology-based manipulation of these diverse sources is the most desirable solution for semantic reconciliation. Furthermore, ontologies are suited for both information storage and retrieval.

1.2 What is an Ontology

Out of many definitions for an ontology, we prefer the following [1]:

> *"An ontology is a model of some portion of the world and is described by defining a set of representational terms. In an ontology, definitions associate the names of entities in a universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms."*

Ontologies can be viewed also as extensions of object-oriented (OO) models. In OO models, definitions provide associations among the names of entities in a universe of discourse by means of objects, attributes, relations, functions, and constraints. However, OO models, as well as entity-relationship and frame-slot models, neither provide the necessary axioms that constrain the interpretation and well-formed use of these terms, nor support other ontological constructs, such as metamodels. In contrast, the Nijssen Information Analysis Methodology (NIAM) [2] supports many of the necessary ontological constructs mentioned above. Any of the models, with suitable enhancements, can be used to represent ontologies. The choice of the model is in the hands of the designer, based on the environment where the ontologies will be used.

The ER model is closely related to the relational databases that are the most commonly used type. As a result, ontologies based on extensions to the ER model can be used not only to organize concepts among tables and fields in a relational database, but also to organize keywords by capturing the semantic relationships among them. In addition, in a distributed environment such as the Internet, ontologies can be used for knowledge sharing.

1.3 Advantages of Using Ontologies

Ontologies have an advantage over unstructured text-based information spaces in terms of value mapping. When a result set is returned from an information source, the recipient might not know the units for that set. For example, when the salaries of employees are requested and the results are returned, the result set does not provide any information about whether the salaries are in dollars or pounds or both. It might be assumed to be the default currency for the country of concern. However, accepting the default is not always safe, particularly given the distribution of information, where a query might be generated in one country with one unit of currency and the result might come from another country with a different unit of currency.

A solution to this problem using an ontology is described in Figure 1 below, where "*factor*" is used as an attribute of the entity "*salary*" that is used to map one currency to the other. When the result is returned, the value of the attribute "*factor*" would indicate the mapping factor used in the result set, and the system that issued the query can convert the results to the correct currency. Even though this is a simple example, the same concept can be used to solve more complex problems.

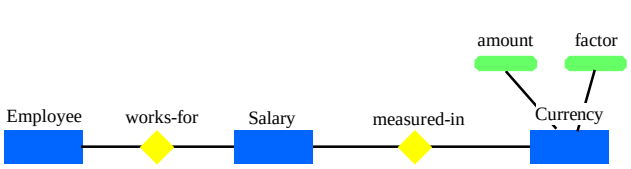

TABLE: *Currency*

currency name	factor
US $	1.00
DM	0.62
others..	…

Figure 1. Value mapping example using *Employee-Salary-Currency* Tables

Since ontologies support the structure of information, they can be represented graphically, as in an entity-relationship diagram. Graphical representations are easier than textual representations for most users to comprehend. Large and complex information sources can be effectively displayed by the use of graphical abstraction mechanisms. Ontologies can also be used to eliminate the confusion and redundancy inherent in unstructured text. In graphical displays, a user can form queries by simple mouse clicks, whereas in a textual representation the user is expected to type a query. Furthermore, the hierarchical information of ontologies can be represented with greater clarity by graphs than by text. Without structured data, it is hard to capture the semantics of information spaces, and without the proper semantics, semantic reconciliation can not be carried out in an effective manner.

Ontologies can grow and shrink as necessary based on the context where they are being used. In a new context, part of one ontology can be hidden or another made visible, so that a new view of the same information space can be generated to suit a certain audience. Commercial databases usually provide this procedure. Ontologies provide it in large information spaces spanning many resources. In addition, portions of ontologies created by experts from a variety of fields can be merged to create larger ontologies.

2. BACKGROUND

There have been several attempts to accomplish the task of implementing distributed ontology-based information systems. These include MCC's Carnot and Cyc projects, Stanford University's Ontolingua, and SRI International's GKB (Generic Knowledge Base) editor.

The Carnot [3] project was initiated in 1990 with the goal of addressing the problem of logically unifying physically distributed, enterprise-wide, heterogeneous information. The Model Integration Software Tool, developed as part of this project, is a graphical tool that assists a user in the integration of different databases via a common ontology that serves as an enterprise model.

The Cyc project is an ongoing attempt at building a large-scale knowledge base. Doug Lenat began it as a research initiative in 1984 at MCC. The Cyc common-sense knowledge base is the result of a large effort to encode a general ontology of the world, along with the rules that govern the common-sense relationships among the components of the ontology [4]. The primary task of the project is codifying a vast amount of knowledge that is considered as "consensus reality"—the background knowledge possessed by a typical person. In addition to this knowledge, Cyc contains a wide variety of reasoning mechanisms for generalized deduction and analogical inference.

Ontolingua [5] is a set of tools, written in Common Lisp, for analyzing and translating ontologies. It uses KIF [6] (Knowledge Interchange Format) as an interlingua and is portable over several representation systems. It includes a KIF parser and syntax checker, a cross-reference utility, a set of translators into implemented representation systems, and an HTML report generator.

The GKB-Editor [7] (Generic Knowledge Base Editor) is a tool for graphically browsing and editing knowledge bases across multiple frame representation systems in a uniform manner. It offers an intuitive user interface, in which objects and data items are represented as nodes in a graph, with the relationships between them forming the edges. Users edit a KB through direct pictorial manipulation using a mouse. An incremental browsing facility allows a user to selectively display only that region of a KB that is currently of interest, even as that region changes.

The InfoSleuth project [8], based on MCC's Carnot technology, was created with the intention of developing and deploying technologies for finding information in corporate and external networks. The InfoSleuth architecture is a collection of agents that represent users, information sources, ontologies, and query engines, and that cooperate in finding and fusing information.

Even though semantic networks have been shown to be excellent vehicles for modeling controlled vocabularies, they often lack the necessary access flexibility and robustness required by external agents, such as intelligent information locators and decision-support systems. The OOHVR project [9] is an attempt to solve this problem by mapping an existing medical vocabulary based on a semantic network model into an object-oriented database system.

The goal of the TSIMMIS Project [10] is to develop tools that facilitate the rapid integration of heterogeneous information sources that may include both structured and unstructured data. TSIMMIS has components that extract properties from unstructured objects, translate information into a common object model, combine information from several sources, allow browsing of information, and manage constraints across heterogeneous sites.

3. JAVA ONTOLOGY EDITOR (*JOE*) DEVELOPMENT

JOE is a software tool, written in Sun's Java language and developed using Microsoft's Visual J++ developer studio, that (a) provides a graphical user interface (GUI) to represent ontologies and (b) allows users to make queries on them.

3.1 Use of Java

The decision to use Java as the development language was influenced by many of its features that fit well in a heterogeneous, networked, and distributed environment. First and foremost, Java code is architecture-neutral; thus, Java applications are ideal for a diverse environment like the Internet. Secondly, Java programs are multithreaded and, as a result, are capable of generating better interactive responsiveness and real-time behavior. Furthermore, the Java Database Connectivity (JDBC) toolkit allows database access in a straightforward manner.

Another desirable feature is that Java applets can be downloaded anywhere at any time to a Java-compatible browser. This means that the same ontology can be simultaneously viewed and edited by more than one user. This

group-editing feature eliminates the problem of keeping different copies of the same ontology up to date, since only one correct version need be saved. Experts from different fields can jointly build an ontology, possibly by merging smaller ontologies. This capability is useful in large enterprises.

One of the disadvantages in using Java as a development language is that most web browsers, for security, prohibit applets from making socket connections, except to the server address (i.e., to the domain address from where the applets were downloaded originally). Another is that, in order to safeguard client sites, applets are prevented from reading or writing to the client's host computer. In addition, Java programs exhibit generally poor real-time performance.

Sun's solution to the first two problems above is the RMI [11] (Remote Method Invocation) API. RMI enables the programmer to create distributed Java-to-Java applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. An applet downloaded on a client site can talk to an applet on the server site in a secure fashion via a client-server mechanism.

JOE is implemented as a collection of interacting Java applets. It has two main applet components [12], namely an Ontology Editor (OE) and a Query Editor (QE). The OE provides a user interface where a user can create a new or edit an old ontology by adding entities, attributes, and relations. The QE is a user interface that allows a user to build queries on the information space that is displayed in the Ontology Editor. Ontologies are given a graphical ER representation, because of its familiarity and its closeness to relational databases. However, JOE does not yet provide mechanisms for displaying axioms and constraints in a graphical format.

3.2 Abstraction Mechanisms

Graphical ontology editors typically do not work well when there are a large number of nodes or links (arcs that represent relationships among nodes), because of the limited viewing area of computer monitors. In addition, navigating in such large and complex ontologies can be confusing. JOE handles these problems through innovative abstraction mechanisms.

Selective viewing - JOE allows the user to view an ontology with complete details or with only selected types of nodes in the display: a user can view entities only, entities and attributes, or entities and relations. This option can reduce the complexity and confusion involved in a large ontology. In Figure 2, the left side shows just entities and attributes and the right side shows entities and relations.

Searching - JOE provides a window, as shown on the left side of Figure 6, with a listing of all available nodes in an ontology. The user can locate a node by just double clicking on the name of that node on the displayed list, and JOE will automatically scroll the viewing window to that particular node no matter where it is located. With this option, users do not need to search randomly through a large ontology for some specific node.

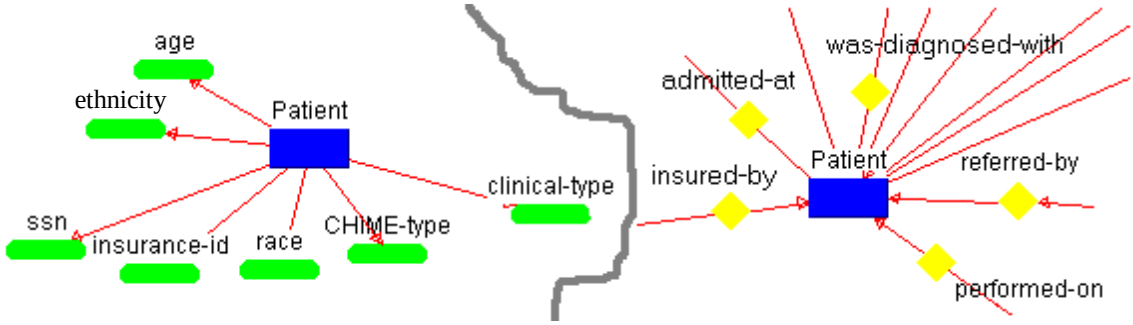

Figure 2. JOE with two snapshots of selective viewing
[entities and attributes on the left, and entities and relations on the right]

Hierarchical information - JOE displays hierarchical information about an ontology using a tree-like structure. This is useful for ontologies having a large number of super/sub-class relationships. A user can selectively view (expand) only a part of an ontology and hide (collapse) the rest.

In Figure 3, the object hierarchy portion of an ontology is represented as a tree. (Multiple inheritance is also supported, although not shown.) The "+" sign indicates that a node can be further expanded to display other subclasses. This feature can be used to avoid cluttering the viewing area with unnecessary nodes that may cause confusion. Here the user is interested in obtaining information only about two classes, '*Cars*' and '*Trucks*'.

Furthermore, this feature can be used to scale very large ontologies to different abstraction levels for selective viewing.

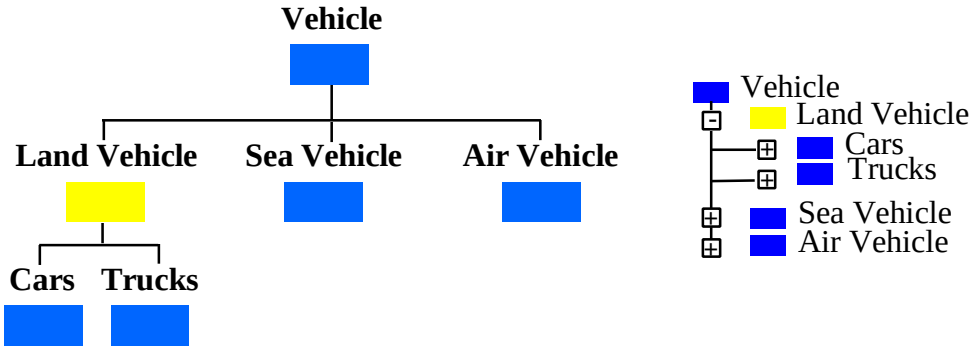

Figure 3. A vehicle ontology and its corresponding hierarchy tree

Zooming - JOE can display an entire ontology inside the current window by zooming out appropriately. Figure 4 shows the entire healthcare sample ontology sized to fit the current window.

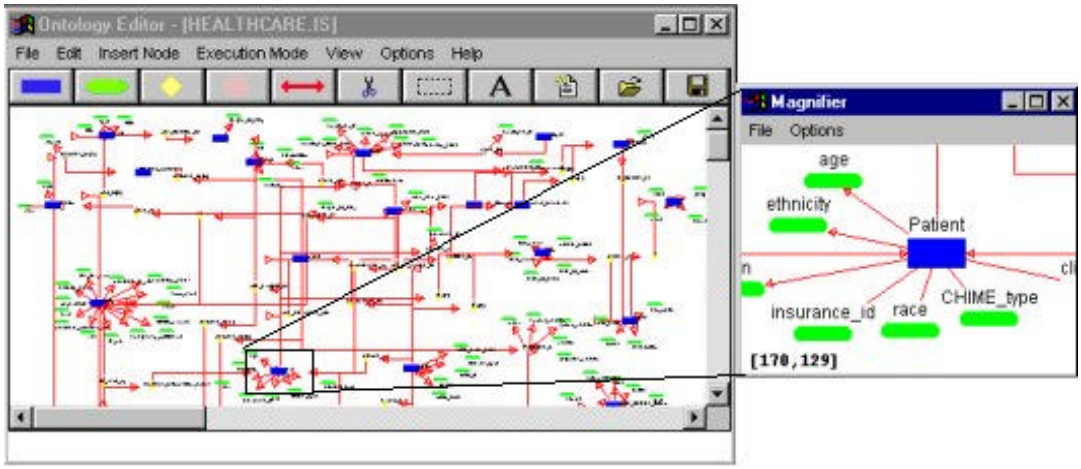

Figure 4. JOE displaying an entire ontology within the current window with a magnified view of the selected area on the right

Magnification - When the entire ontology is displayed as described above, JOE also provides a "magnifying glass" that will magnify a small portion of the ontology under the mouse. This is necessary if the ontology is too large for detailed information to be displayed in the zoomed-out representation. This magnified portion is displayed in a separate window, as shown in the right side of Figure 4, giving the user a sense of which region of the ontology he or she is viewing at that moment. If the user clicked the mouse anywhere inside the window, then the viewing mode will be set to normal and the window will be scrolled automatically to display that region of the ontology.

4. A TEST APPLICATION

The development of JOE has been targeted towards a healthcare application that is part of the Healthcare Information Infrastructure Technology (HIIT) project [13] funded under the NIST Advanced Technology Program. The healthcare industry provides many opportunities in which tools such as JOE can be used. Even though there are many types of hospitals and healthcare providers all over the country, their information needs are essentially the same. Yet, it is difficult for one facility to request information from another, because the architecture used to represent such information is different from one facility to another and it is difficult to translate among them.

The idea behind our application is to represent, simply, an abstract view of the information fundamental to all healthcare industries by a common ontology, so that queries made on this ontology will support a uniform standard irrespective of individual healthcare facilities. These queries, eventually, would be further refined through intermediate "translation agents" before being processed by individual healthcare providers. By accepting a common standard, all healthcare providers will be able to communicate freely with each other, while at the same time continuing to maintain their individual information source architectures. This is not only a feasible solution, but also an economical one, since the cost involved in reengineering each facility to adhere to a new standard would be very expensive.

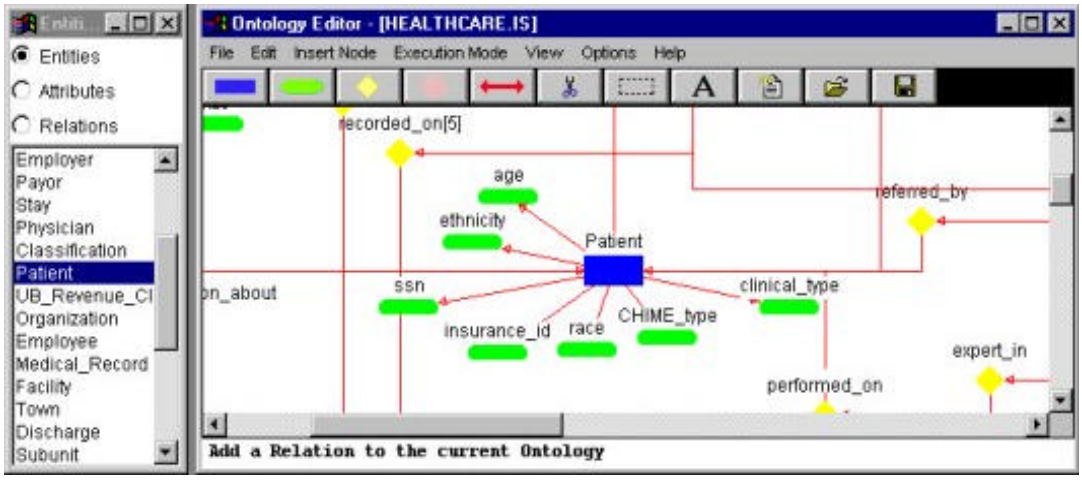

Figure 5. JOE executing in the editor mode

Figures 5 and 6 show JOE executing in its editor mode and its query mode, respectively. A part of the information space for a healthcare facility is displayed as a graph showing the Patient table, all of its columns, and a few of its relations. On the left side of the main window is a list of all tables, attributes, and relations, which a user can directly go to just by clicking a mouse.

When working in the query mode, a user can construct queries by setting constraints on the displayed attributes (shown by green ovals). The constructed query will be displayed on a separate window to the right of the main window. All entities, attributes and relations used in the query are displayed in a different color to indicate that they are part of the current query. JOE will internally translate the graphical query to a standard SQL statement as the user builds the query. The user can submit the query by choosing the "submit" option in the "Query" menu and the results will be displayed in a separate window. JOE also provides an editor where the user, if he or she is an expert in SQL, can directly modify or type in a new SQL statement for execution.

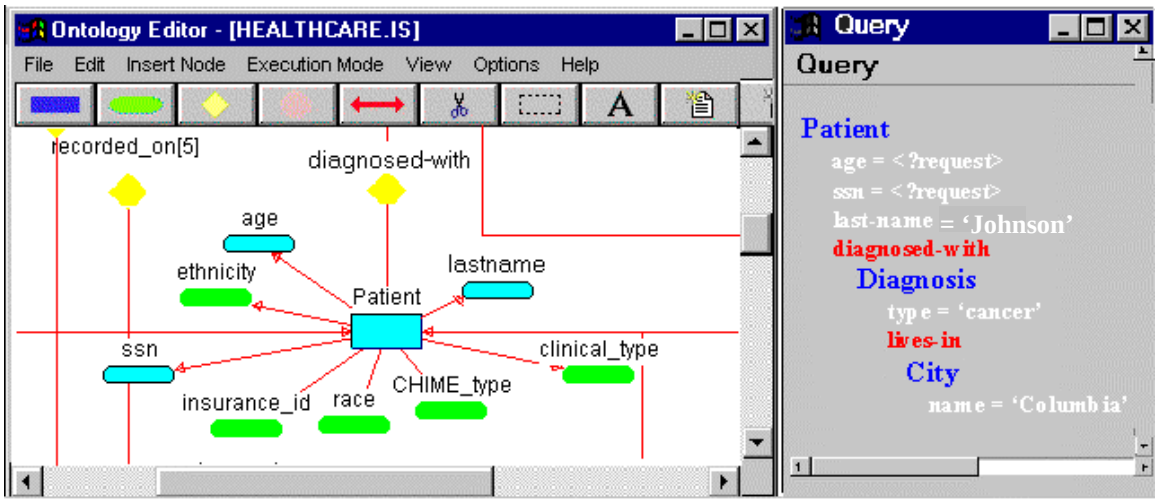

Figure 6. JOE executing in the query mode, with a partial query shown

In the above diagram, the displayed query can be stated as follows:

*"Get social security numbers **(ssn)** and the **ages** of all the **Patients** whose **lastname** is 'Johnson' and who were diagnosed-with a **Diagnosis** named 'cancer' and who **live in** a **City** named 'Columbia'."*

As can be seen, the query shown in the display can be easily translated to a regular sentence. For another example, consider the following query:

```
Patient
    age = ?.request.?
    has
     Medical-Condition
          condition-name = ?.request.?
```

This query can be translated to:

*"Get the **age** of the **Patient** and the **name** of the **Medical-Condition** for each **Patient**"*

5. DISCUSSION AND CONCLUSIONS

With the use of Java's RMI client-server architecture and object serialization mechanisms, ontologies can be saved when the applet runs as a stand-alone application. But until browser developers provide RMI support in their software, we cannot store or retrieve ontologies when the applet executes inside a browser. We store a few sample ontologies on our web site, so that users can open them remotely and edit them if necessary. With the help of RMI, JOE allows users to save ontologies at the server or, if desired, at the client site. In a similar fashion, users can open existing ontologies located either at a remote server or at the client site.

We are working on a better representation for the queries in JOE, so that the constructed queries will be easier to read. In addition, we are providing support for libraries of queries, where they can be stored for later use, just as in the case of ontologies. We are also working on a better layout of nodes and a few more editing features based on user requirements.

We believe that ontology-based representation of information spaces can be effectively used to achieve interoperation among distributed and heterogeneous information sources. In addition, ontologies can be used as tools for semantic reconciliation among software agents. Our tool, JOE, is an attempt to provide a GUI in Java to create and edit such ontologies.

Using JOE, any user (or even software agents) can then access the ontology on the Internet to copy it or make modifications to it. By use of Java's security features, access to such ontologies can be monitored to avoid misuse. Those users who have access to ontologies can make queries on them by use of point-and-click approaches. Our ultimate goal is to provide a generic, easy-to-use graphical tool that is not targeted to any specific domain or any domain model. Users could then get desired information with this tool in an efficient manner and without special training.

REFERENCES

1. T. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220, 1993. [http://www-ksl.stanford.edu/kst/what-is-an-ontology.html]
2. G. M. Nijssen and T. A. Halpin, "Conceptual Schema and Relational Databse Design", Prentice Hall, Victoria, Australia, 1989.
3. M. N. Huhns, N. Jacobs, T. Ksiezyk, W. M. Shen, M. Singh, and P. Cannata, 1992. "Enterprise Information Modeling and Model Integration in Carnot," *Enterprise Integration Modeling: Proceedings of the First International Conference*, The MIT Press. [http://www.mcc.com/projects/carnot]
4. D. Lenat and R. V. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*, Addison-Wesley Publishing Company, Inc., Reading, MA, 1990. [http://www.cyc.com/]
5. The Knowledge Systems Lab (KSL) at Stanford University [http://www-ksl.stanford.edu/knowledge-sharing/ontolingua/ontolingua.html]
6. M. Genesereth, "Knowledge Interchange Format," *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, Cambridge, MA, pages 599-600, Morgan Kaufmann, 1991. [http://www-ksl.stanford.edu/knowledge-sharing/kif/]
7. P.D. Karp, K. Myers, and T. Gruber, "The generic frame protocol," in *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*, pp. 768--774, 1995. [http://www.ai.sri.com/~gkb/]
8. D. Woelk, M. Huhns, and C. Tomlinson. "Uncovering the Next Generation of Active Objects," *Object Magazine*, vol. 5, no. 4, pp. 32-40, July/August 1995. [http://www.mcc.com/projects/infosleuth/]
9. "Modeling a Vocabulary in an Object-Oriented Database - OOHVR Query Interface Project," New Jersey Institute of Technology Technical Report. [http://object.njit.edu:2000/~oohvr/QIP.html]
10. H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaratnam, Y. Sagiv, Ullman, and J. Widom, "The TSIMMIS Approach to Mediation: Data Models and Languages," *Proceedings of the NGITS (Next Generation Information Technologies and Systems)*, June 1995.
11. Java(TM) Remote Method Invocation Specification. http://chatsubo.javasoft.com/current/doc/rmi-spec/rmiTOC.doc.html
12. Kuhanandha Mahalingam, "The Java Ontology Editor (JOE)," Center for Information Technology, Electrical and Computer Engineering Department, University of South Carolina, 1996. [http://www.ece.sc.edu/Labs/hiit/html/imts-new.html]
13. The Healthcare Information Infrastructure Technology Program [*http://host.scra.org/hiit.html]*