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, U.S.A. 29208
(803) 777-5921 or kuha@sc.edu
(803) 777-8045 FAX

Keywords: Semantic Reconciliation, Ontology, Java, Healthcare Information Systems, Group Editing, Distributed Database Systems, Agent Technology.
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 in 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 Sun Microsystems’ 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 be used effectively in such environments. The software is applied to an information system for healthcare administrators, which spans hospitals, clinics, and governmental health departments.
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 this tool and its use in a representative application.

1.1 Use of ontologies as a basis for semantic reconciliation

As a consequence of the Internet’s rapid growth, its users are faced with many new problems. One of the most frequently encountered problems is how to search for and retrieve necessary information from the large number of information sources available on the Internet. The information provided by the sources is no longer just simple text, but now 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 or even efficient. In order to keep pace with the growth of the Internet, we need not only suitable storage and retrieval mechanisms, but also efficient search tools that can harvest the necessary information from these sources. 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. Mechanisms are needed that allow efficient querying on
diverse information sources that support structured as well as unstructured information.

In such complex and heterogeneous environments, ontologies are best suited for
information storage and retrieval. In contrast to unstructured data, data stored as
ontologies can capture the structure in addition to the semantics of information spaces.
We believe that in a distributed and heterogeneous environment such as the Internet,
ontology-based manipulation of these diverse sources is the most desirable solution for
semantic reconciliation.

1.2 What is an ontology

From an artificial intelligence viewpoint, an ontology is defined as follows [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.”

From a different viewpoint, ontologies can be viewed 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, most OO frameworks, such as entity-relationship (ER) or frame-
slot (FS) 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 (i.e., defining a model by using the model itself). As opposed to ER or FS
models, the Nijssen Information Analysis Methodology (NIAM) [2] supports many of the
necessary ontological constructs mentioned above. As a result of similarities between such models and ontologies, ontologies can be easily represented in an easy-to-read graphical format by using any one of the models mentioned above. The choice of the model is ultimately in the hands of the designer and the environment where these ontologies will be used.

As opposed to others, the ER model is closely related to the relational databases that are the most commonly used databases today. As a result of this similarity, ontologies can be used effectively not only to organize database 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. In such an environment, if necessary, such options as merging or splitting ontologies can be used to manage large information sources that are prevalent.

1.3 Advantages of using ontologies

1.3.1 Provisions for value mapping

Using ontologies has an added advantage over unstructured text-based information spaces in terms of value mapping. Currently there are no tools that can map values from one unit to another, so that when a result set is returned the user knows the corresponding 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 is usually assumed to be the default currency used in the country of concern. However, accepting the default is not always safe,
particularly given the distributed nature of the environment, 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. Worse, results might come from
databases in several countries. In a distributed and heterogeneous information
environment, there are many such problems in mapping values of data from one to
another.

With an ontology, there are several ways to find a suitable solution to this problem.
One such solution 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. Thus, when the result is
returned, the value of the attribute “factor” would indicate the mapping factor used in the
result set and the system or person who made the query can easily convert the results to
the correct currency. Even though the example we gave here is a very simple one, the
concept can be used to solve more complex problems in a similar manner. This advantage

1.32 Suitable for graphical representations

Since ontologies support the structure of information, they can very easily be
represented graphically, as in the Entity-Relationship diagrams mentioned above. In
reality, graphical representations are much easier to comprehend by an average user than textual representations of the same information. Large and complex information sources can be effectively displayed by use of various graphical abstraction mechanisms in an easy to comprehend format. Ontologies can also be used to eliminate the confusion and redundancy inherent in unstructured plain textual representation. In addition, in graphical displays a user can form queries by simple mouse clicks whereas in a textual representation the user is expected to type in the query. Furthermore, the hierarchical information of ontologies can be represented with greater clarity by graphs than by just text alone. Without structured data, it is hard to capture the semantics of information spaces, and without the proper semantics, semantic reconciliation cannot be carried out in an effective manner.

1.3.3 Ability to view at various abstraction levels and ability to scale

Ontologies can grow and shrink as necessary based on the context where they are being used. In a different 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. In large databases, this is a common procedure that is executed often. With ontologies it can be done efficiently and faster. In addition, sub-ontologies created by experts from a variety of fields can be merged effortlessly to create super-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.

2.1 Carnot Project and MIST

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 (MIST), developed as part of this project, is a graphical user interface that assists a user in the integration of different databases via a common ontology that serves as an enterprise model.

2.2 Cyc

The Cyc project is an ongoing attempt at building a large-scale knowledge base. Doug Lenat began it as ten-year 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 the encoded “common-sense” knowledge, the Cyc system contains a wide range of reasoning mechanisms for the purpose of generalized deduction and analogical inference.
2.3 Ontolingua

Ontolingua [5] is a set of tools, written in Common Lisp, for analyzing and translating ontologies developed by the Knowledge Systems Lab (KSL) at Stanford University. 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 from KIF into implemented representation systems, and an HTML report generator.

2.4 Generic Knowledge Base Editor

The GKB-Editor [7] (Generic Knowledge Base Editor) developed by SRI International, 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 or pen. An incremental browsing facility allows the user to selectively display only that region of a KB that is currently of interest, even as that region changes.

2.5 InfoSleuth Project

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 in 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.
2.6 Object Oriented Healthcare Vocabulary Repository (OOHVR)

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 at the New Jersey Institute of Technology [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.

2.7 TSIMMIS Project

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)

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. Unlike other languages, Java has many advantages when used in a distributed environment of autonomous and heterogeneous information resources, which characterizes our application domain—the healthcare industry. The long-term goal is to integrate JOE into MCC’s InfoSleuth and NJIT’s OOHVR projects.
3.1 Why Java—the advantages

The decision to use Java as the development language was influenced by many of its desirable 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. In addition, Java is a dynamic language that was designed to adapt to an evolving environment, such as the Internet. Furthermore, Sun’s Java Database Connectivity (JDBC) tool kit allows database access in a quite simple and straightforward manner.

Another very desirable feature of Java is Java's applets that can be downloaded anywhere at any time as long as the user has access 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 has many advantages. It saves storage space since several users can work on copies of a single original ontology. It also eliminates the problem of keeping different copies of the same ontology up to date since only one correct version is saved. At the same time, experts from different fields can jointly build an ontology over a length of time or if desired merge various sub-ontologies to create one large super-ontology. This feature is very desirable in large enterprises where such joint ventures take place on a regular basis.
3.2 Disadvantages

There are quite a few, though temporary disadvantages in using Java as a development language. Most web browsers, such as Netscape and Microsoft Internet Explorer, as a security measure prohibit applets from making socket connections, except to the server address (i.e., to the domain address from where the applets were downloaded originally). Apart from that, in order to safeguard client-site security, applets are also prevented from reading or writing to the host computer on the client site. In addition, even though stand-alone Java programs exhibit good real-time behavior, running on top of other systems like Unix, Windows, the Macintosh, or Windows NT, the real-time responsiveness is limited by that of the underlying system.

Sun’s solution to the two problems mentioned above is the RMI [11] (Remote Method Invocation) API that was introduced with their later versions of Java JDK (i.e., JDK 1.1 and higher). 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. Thus, in essence, it provides a mechanism for the downloaded applet on the client site to talk to the applet on the server site in a secure fashion through the use of a client-server mechanism.

4. System Architecture of JOE

4.1 Applet framework

JOE is implemented as a collection of interacting Java applets. The current version of JOE’s main applet [12] has two major components, namely an Ontology Editor (OE) and a Query Editor (QE), as shown in Figure 2. 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 also a user interface that allows a user to build queries on the information space that is displayed on the Ontology Editor.

Currently ontologies are represented by means of ER model because of the model’s suitability for graphical representations and its closeness to relational databases. However in the current version, JOE does not provide mechanisms for displaying axioms and constraints in a graphical format. Nevertheless, work is in progress to add such features as setting constraints and defining axioms to support many ontological constructs that are missing in the ER model as mentioned previously.

![Figure 2. The applet architecture for JOE](image)

4.2 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 is usually not an easy task. In order to provide some feasible solutions to these kinds of problems, JOE has a few new features targeted mainly toward useful abstraction mechanisms.

**Selective viewing** - JOE allows the user to view the Ontology with complete details or if desired, with only selected types of nodes in the ontology. In other words, the user can
view only entities, entities and attributes, or entities and relations. This option to selectively view nodes can reduce the complexity and confusion involved in a large ontology. In Figure 3 below, the left side shows only the entities and attributes and the right side shows only the entities and relations.

Figure 3. JOE with two snapshots of selective viewing
[entities & attributes only on the left and entities & relations on the right]

**Searching** - JOE provides a window, as shown on the left side of Figure 8, with a listing of all available nodes in the 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. This option will eliminate the need for searching a specific node in a large ontology with a large number of nodes.
Hierarchical information - JOE, in the future, will provide a user with the hierarchical information of a given ontology in a tree-like structure (similar to the MS Windows file manager format). This feature is designed for ontologies that exhibit a large number of super/sub class relationships. Given such ontologies, it would be desirable to selectively view (expand) only a certain part of the entire ontology and hide (collapse) the rest so that it would be easier for the user to comprehend. In other words, this feature provides the user an option to view the large and complex ontology at a level of abstraction at which he/she feels comfortable without adding any more confusion.
Currently, the tool mentioned above is an independent Java applet and Figure 4 & Figure 5 are two of its graphical displays showing two different ontologies. In Figure 4, the object hierarchy captured in an ontology describing bibliographical information is represented in a tree. The “+” sign indicates that a node can be further expanded to display other subclasses. Multiple inheritance is also supported as in the node “bibliotext” that inherits both from the node “string” and the node “individual.” This feature can be effectively used to avoid clustering the viewing area with unnecessary nodes that may add more confusion to the user. Furthermore, this feature can be used to scale down very large ontologies to different abstraction levels, thus allowing the user to view only that part of the ontology as mentioned earlier.

For example, in Figure 6, a sample ontology describing different vehicles along with the class hierarchy is shown. JOE will display only the expanded nodes and not the others. In this example, we assume that the user is interested in obtaining information only about two classes, “Cars” and “Trucks” as shown below. Therefore it is not necessary to display any information about other classes in the ontology. It is not only redundant as far as the user is concerned, but also would introduce unwanted complexity.

![Vehicle Ontology Diagram](image)

**Figure 6.** A vehicle ontology and its corresponding hierarchy tree
**Zooming** - JOE can display an entire ontology inside the current window by zooming out appropriately. This feature allows the user to view the complete ontology in full at any time. In Figure 7 below, the entire healthcare sample ontology is shown zoomed out to fit the current window.

![Image](image.png)

**Figure 7. JOE displaying an entire ontology within the current window and the 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 quite large and detailed information can not be displayed in the zoomed-out image of the ontology. This magnified portion is displayed in a separate window, as shown in the right side of Figure 7 above, 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.

In addition to the above abstraction mechanisms, JOE also supports a basic editing functionality, such as selecting, cutting, and moving. However, the current version of JOE
does not support undoing an operation, copying, or pasting. These operations will be added in the near future.

5. A Test Application

Currently, the development of JOE is targeted towards a healthcare application that is part of the Healthcare Industry: The Healthcare Information Infrastructure Technology (HIIT) project [13] funded under the Department of Commerce Advanced Technology Program.

The healthcare industry provides many opportunities in which such tools as JOE can be used. First and foremost, even though there are so many hospitals and healthcare providers all over the country, the information fundamental to all these facilities is the same no matter where they are located. Yet, it is not possible for one facility to request information from another in a straightforward manner. It is mainly because there are no globally accepted standard architectures. The architecture used to represent such information is different from one to the other and it is difficult to translate from one facility to another.

The idea behind our application is to represent, simply, the 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 “translating-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.

Figures 8 and 9 show JOE executing in its editor mode and its query mode, respectively. A part of a healthcare facility information space 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 a list of all tables, attributes, and relations is provided so that the user can easily go to any node in the healthcare information space just by double clicks.

![Figure 8. JOE executing in the editor mode](image)

When executing in the query mode, the users can construct queries by setting constraints on the displayed attributes (shown in green round-rectangles). 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 standard SQL statement as the user builds the query. The user can simply submit the query by choosing the “submit” option in the “Query” menu and the results will be displayed on
another 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. Currently, only very simple queries can be translated to actual SQL statements. However, the work is in progress where all queries created in JOE will be mapped to proper SQL statements no matter how complicated they may be.

Figure 9. JOE executing in the query mode with a partial query
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 Diagnosis named ‘cancer’ and who lives in a City named ‘Columbia’”

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

```
Patient
age = ?.request.?
```

Here “?.request.? means that the user wants the corresponding attribute (i.e. “age”) from the Patient table in the result set.

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

This query can be translated to:

“get the age of the Patient and the name of the Medical-Condition for each Patient”
6. Discussion and Conclusions

6.1 Plan for the future

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. Until the web browser developers provide RMI support in their software, we can not store or retrieve ontologies within a web browser when the applet is run inside it. Currently few sample ontologies are stored temporarily (simply for demonstration purpose) as Java classes so that users can open them remotely and edit them if necessary. The user, even though allowed to edit the ontologies, can not save them. With the help of RMI, JOE will allow users to save ontologies at the server or if desired locally at the client site. In a similar fashion, the users can open any existing ontologies located either at a remote server or at the client site. In the latest version of JDK (i.e., 1.1.3) RMI is distributed as one of the core packages and only Sun’s own Web browser called HotJava supports all of Java’s new features. In the near future, all web browsers will provide support for RMI and only then can JOE, like all other applets, take advantage of the RMI functionality to the fullest extent.

We are also planning to represent all ontologies in KIF (Knowledge Interchange Format) or another similar standard format and allow translation capabilities so that JOE will support a common standard that suits a distributed environment. The hierarchical tree of bibliographical ontology as shown in Figure 4 above was created by parsing the representation of bibliographical Information Space in the KIF format. This ontology is
We are also 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 where queries can be stored so that they can be used by anyone at a later time just as in the case of ontologies. We are also working on a better layout of nodes in the ontologies and few more editing features based on user requirements.

6.2 In conclusion

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.

The framework of JOE provides a standard but simple GUI that can be used to create or edit any ontology. Any user (even other software agents) can then access the ontology on the Internet to copy it or make modifications to it, depending on their individual needs. By use of Java’s security features, access to such ontologies can be monitored to avoid misuse. Those users who have access to these ontologies can make queries on them by use of point-and-click approaches. It is our intention that JOE be user friendly enough that all users can benefit by using it. Our ultimate goal is to provide users 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 any specific expertise.
References


