Web-based Service Exchange System for Agents and Humans Alike

Evens Jean, Machigar Ongtang, Ali R. Hurson
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16801

Abstract

Semantic Web research aims at bridging the gap between how humans and agents process information readily available on the Internet. One of the great challenges to this goal lies in the fact that humans, contrary to agents, can extract the meaning of words based on its context. This work introduces a service exchange system for the web that allows agents to intelligently process information, as would humans. This is achieved by the use of thesauri to help agents resolve semantic heterogeneity in the information being processed. The framework for the exchange system has been realized under the Aglet platform on the Secure Aglet Server SAS, thus providing it with a secure execution environment. This article discusses the exchange system and presents a prototyped pharmaceutical marketplace built using the exchange system framework. The prototype showcases the ability of the system to support intelligent information processing by agents and humans alike.

1. Introduction

Extending the World Wide Web (WWW) to support a host of agents interacting and processing information in order to assist humans is the focus of Semantic Web research. WWW has been designed primarily as an interaction medium for humans. As such, it presents numerous challenges to the introduction of agents to visualize, interpret and gather the information readily available. Research efforts to address the challenges in introducing agents to the WWW have resulted in many proposals. One such proposal is the Resource Description Framework (RDF) [10], a language for representing resource information on the Internet. RDF Schema (RDFS) is a language designed to represent information in the web [2]. As an attempt to improve RDFS, a modeling architecture has been proposed [12].

The proposals thus far discussed have attacked the issue from the perspective of semantics of languages. Other proposals have also surfaced, focusing on different aspects of the challenges of Semantic Web. Various efforts have addressed the support of knowledge sharing, typically achieved through ontologies. Such works have studied different aspects of ontology varying from learning approaches to representation languages [13]. As semantic web aims at supporting the deployment of agents, the issues of proofs and trust also surfaced in the research area, although they have not been the focus of major research efforts as of yet [15, 13]. In general, semantic web research has typically tackled the challenges of the field from three standpoints, namely:

- Semantics of languages for the semantic web
- Ontology development
- Proofs and trust

While the typical approach to semantic web research is quite promising, it still suffers from several limitations. One such limitation is the adoption of a standard way to represent and process information available on the web. In allowing agents and humans to co-exist on the web, we have taken a novel approach through the use of thesauri to assist agents in understanding the information at hand. This work introduces a service exchange system for the web that allows agents to intelligently process information, as would humans. The system has been realized through the use of an online thesaurus. The thesaurus contains domain-specific pre-defined terms organized into a hierarchical structure based on their semantic relationship, i.e. synonyms, hypernyms (broader terms), and hyponyms (narrower terms). It also provides semantic similarity measures in terms of semantic distance between terms. Thus, the thesaurus can mask out the heterogeneity among the information from different service providers, including the difference in classification criteria and vocabularies, naming as well as relationship conflicts. By incorporating the thesaurus into the

1530-1605/08 $25.00 © 2008 IEEE

system, the agents can reason about the classification of related concepts and properties, and semantic connection among its requirements and the available services.

The information present in the system is presentable to both humans and agents, through reliance on the Extensible Markup Language (XML) [1]. Furthermore, the system is made up of various agent entities coordinating to provide a secure environment where services can be advertised, located and processed on behalf of human users. An application framework of the system has been developed and a prototyped pharmaceutical marketplace is herein discussed.

The remainder of this article starts out by presenting the necessary background information in section 2. Section 3 discusses the overall architecture and requirements of the system, while section 4 showcases the prototyped application. We conclude our presentation in section 5 and highlight our future works.

2. Background

In this section, we provide some important concepts and relevant works that have been proposed in this area.

2.1 Semantic Web Technologies

WWW technology is continuously evolving. In addition to web interface that interacts with human, machine-to-machine interactions is made possible using Web Services (WS) technologies. Moreover, the integration of semantic information into the web content allows the webpage to be both human-understandable and readable by machine or software agent, known as semantic web. The further extension along this line is the development of Semantic Web Services (SWS), which enable web service's automation though the application of agents and ontology.

Our proposed architecture for web-based service exchange provides the functionalities of the Semantic Web Service and the Semantic Web as will be explained in section 3. However, instead of using services ontology, the key component for semantic feature of our scheme is an online *thesaurus* [3, 19, 9]. Our system also provides service discovery in a manner similar to directory service in WS.

2.2 Mobile Agents in Semantic Web

A mobile agent refers to a software entity that can halt its execution, move to a new environment, and resume its execution. This programming paradigm is appropriate for performing web-based service in an open environment, where the available services reside in different hosts in the network. A common application of such autonomous agents is to travel through the Internet to search for a specific group of products, provide their user with the product details and prices, and direct the user to the vendors, known as *shop bots*. More intelligent shop bots can also make purchasing decision on behalf of their user.

2.2 Related Works

Several previous works utilizes semantic data in performing web-based transactions. Most of these works are based on web service technologies. To our knowledge, there is no existing work that utilizes Thesaurus in Semantic Web or Semantic Web Services.

Mediator-wrapper architecture [14] was proposed for semantic integration of multiple heterogeneous data sources in the semantic web context. Using OWL-based approach, the scheme provides a mediation system to overcome the semantic heterogeneity between local systems and to allow transparent interoperability among different independent local data sources.

Agent-based Web Service [5] further expands the semantic web services by separating communicative intent of the message from the domain-specific content of the message by utilizing Agent Communication Language (ACL). This concept stems from the multi-agent systems where the intention of the agent (e.g. request or assertion) is separated from its content.

Intelligence Commerce System (ICS) [16, 17] is an agent-oriented B2B e-commerce designed for an open environment such as Internet. The ICS architecture is composed of an ontology repository containing the domain's ontologies, a stereotype database storing users' profile, and several marketplaces. Each marketplace acts as a web service, which groups buyers and customers each represented by a mobile agent. ICS features the matchmaking agent, which is responsible for making queries in advertises database residing in each marketplace to match business partners and enable them to start their negotiation. The matchmaking agent is implemented with semantic matching engine and DAML-S parser. It returns the degree of matching, namely exact, subsume, plug-in, or fail.

Our service discovery scheme is different from ICS as it is not based on Web Service technologies. We use thesaurus to resolve semantic relationship between terms instead of ontology. Through the use of thesaurus, our approach yields several advantages. (i) It can describe the degree of semantic similarity in a more fine-grained measure. The semantic distance can be ranged from 0 (i.e. for synonyms) to infinity (i.e. for anonyms). (ii) The users can freely fine-tune their search by specifying appropriate semantic distance. (iii) The relationships between terms in ICS must be explicitly defined in its ontology; otherwise, they would not be recognized. In contrast, these relationships are implicitly inferred in our proposal from the thesaurus. There are some well-known thesauri for general English available such as Roget's Thesaurus [4] and WordNet [11, 18].

The use of thesaurus has been presented in several works. An agent-based information retrieval system called MAMDAS (Mobile Agent-based Mobile Data Access System) [9] exploited a thesaurus as a plug-in to its search engine to resolve semantically imprecise queries. It used a hierarchical multidatabase federation model, called the Summary Schemas Model (SSM) as its platform. The goal of MAMDAS is to provide access to several heterogeneous data sources residing in multiple machines or nodes. Such data sources may have different information models, representations, and classification criteria. Each node publishes its schema to the SSM, which captures semantic information of data objects in the underlying data sources at different levels of abstractions. In response to all queries, a thesaurus server accessible through ThesAgent provides semantic distance between terms in the query and terms in the node in the SSM, which is the abstraction of the data items in the data sources under that node. Thus, the agent corresponding to that query can infer the degree of relevance between the query and the data items.

A medical thesaurus called MEDTHES [19] was developed to address the issue of semantic-based information retrieval of biomedical data. It follows ANSI/NISO standard and was realized as a plug-in to MAMDAS to build a biomedical search engine. Three semantic distance calculation algorithms, namely Edge Count Algorithm, Leacock & Chodorow algorithm, and Wu & Palmer algorithm were quantitatively studied to examine the correlation and appropriate range of semantic distances.

3. System Architecture

In developing the exchange system, we were inspired by current-day shopping malls where individuals can conduct various transactions. Our exchange system is thus modeled to support the exchange of any services, such as the sale of goods, information etc. The core of the exchange system is realized through the use of multiple interacting agents emulating the entities typically encountered in shopping malls. Some of the agents present in the system were introduced to address issues specific to agent systems. To sum up, the exchange system consists of 6 interacting agents. Section 3.1 describes the agents and their responsibility while we defer the discussion about their interactions to section 3.2.

3.1. Agent Responsibilities

The Exchange system essentially consists of agents representing the various entities typically found in shopping centers. As such, the traditional customer is generalized and modeled as a ConsumerAgent, while the shop owners or sellers are modeled as ProducerAgent. Shopping centers typically have security personnel to ensure that customers, or sellers for that matter, do not misbehave; we thus introduced a SecurityAgent to the system to undertake the task of security personnel. The introduced SecurityAgent also handles the task of controlling what ProducerAgent or ConsumerAgent should be allowed to execute in the system, and thus, also acts as the traditional manager of shopping centers. While not necessarily represented by human instances, shopping centers typically have some way of guiding shoppers to the appropriate shops. This typically occurs either through the use of an information desk or simple posted directories/maps. DirectoryAgent undertake the task of directing ConsumerAgent to ProducerAgent in the system.

As thus far laid out, the exchange system consists of all entities typically interacting in traditional shopping centers. However, there are two major issues that must be tackled in the design of an exchange system made up of software agents, namely semantic and security. By semantic, we refer to the fact that most of the agents will be implemented by different individuals and will yet have the need to interact even with possibly varying representations of the same information. The second issue, security,

stems from the realization that the system must somehow address the issue of repudiation where agents could possibly deny having partaken in a transaction.

In order to address the aforementioned issues two new agents are introduced in the system. ThesaurusAgent deals with the issue of semantic heterogeneity of the various agents in the system. The security issue is dealt with by the introduction of a BrokerAgent to provide non-repudiation whenever two entities enter into a contract.

We herein provide a detailed list of the responsibilities associated with each agent in the system followed by a discussion highlighting the agent interactions in the proposed exchange system.

SecurityAgent
- Register and manage agents in the system

ProducerAgent
- Advertise services being provided
- Manage inventory, if applicable
- Perform comparison shopping, if applicable, to remain competitive

ConsumerAgent
- Interact with user to determine service(s) of interest
- Locate and present to user list of potential choices where the service could be provided

DirectoryAgent
- Register agents in the system along with the services being offered by such agents
- Act as matchmaker for agents

BrokerAgent
- Prevent repudiation of contracts by either parties involved

ThesaurusAgent
- Resolve semantic heterogeneity in the system

3.2. Agent Interactions

Having presented the responsibilities of the different agents in the system, the next step is to define the interactions of such agents to provide an exchange system. In general, agent interaction in the system occurs through message passing. Table 1 provides an overview of the different messages that may be exchanged in the system.

In discussing the interactions of the agents, we will start with the SecurityAgent, being that it is responsible for tracking the agents in the system. Once a new agent arrives, as that may be the case for ConsumerAgent, ProviderAgent and BrokerAgent, the SecurityAgent determines whether or not they should be allowed to execute based on security parameters defined in the current host. It communicates with the arriving agent to determine its role. In the case of brokers, an administrator must have statically specified the arriving agent as a BrokerAgent in order to prevent impersonation. The SecurityAgent will then associate a trust value with the new agents. The trust value will be used in determining whether an agent should be terminated and is heuristically proposed to be a value between 1 and 100. Any agents through communication with the SecurityAgent can access the minimum and maximum trust values enforced in the current marketplace. BrokerAgents can file complaints to the SecurityAgent, and such complaints will negatively affect the trust value of the accused. The SecurityAgent periodically updates the trust values of the agents in the system if there has not been any recent complaint against them. The update occurs by incrementing the current trust value (n) by 1/n. The choice of 1/n is justified by the fact that we want to avoid rewarding idle agents waiting for their trust value to go over a certain threshold in order to carry an attack. The period at which updates occur is left as an implementation detail.

Once the SecurityAgent has decided to grant execution rights to an agent, it will contact the DirectoryAgent and provide it with the means to contact the new agent. The DirectoryAgent may then communicate with the new agent and acquire pertinent information such as services available as well as the schema of the agent if applicable through the GetServices and GetSchema messages. The GetServices message is applicable to BrokerAgents to determine the type of contracts that they are able to service. As such the service should specify the kind of encryption offered as well as the duration for which the broker is willing keep records of transactions. If the newly arrived agent is a ProducerAgent, the DirectoryAgent will process the obtained schema to update the maintained directory of the agents used to enable it to act as a matchmaker. To update the maintained directory, communication with the ThesaurusAgent is required to resolve semantic heterogeneities.

The ThesaurusAgent essentially works as an interface to a thesaurus server. It resolves semantic heterogeneity whenever contacted and provides mechanisms to either extract the relationship between two words or determine the hypernyms and hyponyms of a word. Such mechanisms are provided with the implementation of GetSemanticDist, GetHypernym and GetHyponym respectively.

Table 1: Communication Messages between Agents

Agent Messages		
Sender	***Receiver***	***Message Type***
SecurityAgent	Any Arriving agent	*GetRole*
SecurityAgent	DirectoryAgent	*RegisterAgent*
BrokerAgent	SecurityAgent	*UpdateTrust*
DirectoryAgent	BrokerAgent	*GetServices*
DirectoryAgent	ProducerAgent	*GetSchema*
Any agent	SecurityAgent	*GetMinTrustValue*
Any agent	ThesaurusAgent	*GetSemanticDist*
Any agent	ThesaurusAgent	*GetHypernym*
Any agent	ThesaurusAgent	*GetHyponym*
Any agent	SecurityAgent	*GetMaxTrustValue*
Any agent	SecurityAgent	*GetAgentTrustValue*
ProducerAgent	DirectoryAgent	*GetBroker*
ProducerAgent	ConsumerAgent	*AcceptContract*
ProducerAgent	ConsumerAgent	*AcceptBroker*
ProducerAgent, ConsumerAgent	BrokerAgent	*SignContract*
ProducerAgent, ConsumerAgent	BrokerAgent	*GetContractSignature*
ProducerAgent, ConsumerAgent	BrokerAgent	*ContractRepudiated*
ConsumerAgent	DirectoryAgent	*IsRegisteredBroker*
ConsumerAgent	ProducerAgent	*IsItemAvailable*
ConsumerAgent	ProducerAgent	*ProposedPrice*
ConsumerAgent	ProducerAgent	*GetContract*
ConsumerAgent	ProducerAgent	*ProposedContract*
ConsumerAgent	ProducerAgent	*GetItemPrice*
ConsumerAgent	ProducerAgent	*AcceptContract*
ConsumerAgent	DirectoryAgent	*GetProducers*
ConsumerAgent	ProducerAgent	*GetBroker*
ConsumerAgent	ProducerAgent	*ProposedBroker*
ConsumerAgent	ProducerAgent	*AcceptBroker*

The ThesaurusAgent interacts not only with the DirectoryAgent, but also with the ConsumerAgent as the latter processes results obtained from ProducerAgents.

The ConsumerAgents, based on its interaction with a user, has a defined goal and upon arrival on a host contacts the DirectoryAgent through the *GetProducers* message. Interaction with the DirectoryAgent is aimed at identifying ProducerAgents that may be able to help in accomplishing its goal. The notion of goal in the system is not clearly specified to allow for flexibility. Once a set of ProducerAgents has been identified as having the means to help the ConsumerAgents, the identified set of agents is contacted using their respective semantics, as obtained from the DirectoryAgent in reply to the *GetProducers* message. As the replies of each ProducerAgent are based on their personal semantics, interaction with the ThesaurusAgents is thus required during processing of such replies. The ConsumerAgents may contact the producers and determine if a particular item is available along with the quantity desired, inquire about the price of the item and propose a purchasing price using the *IsItemAvailable*, *GetItemPrice*, and *ProposedPrice* messages, respectively. If a suitable price is found, a request can then be sent for a contract, a new one can also be proposed through the *GetContract* and *ProposedContract* messages. The *AcceptContract* message is used as a handshake to signal that both parties have agreed to a contract. To allow for flexibility in the implementations of the system, we refrain from specifying the representations of contracts. We however note that any contract should provide the mechanisms for brokers to determine if it is equal to another. Furthermore, contracts should allow for brokers to determine if a violation has occurred along with the associated penalty for such violations. Upon completion of

the handshake, negotiation may occur to determine a suitable broker for the contract through the *GetBroker*, *ProposeBroker*, and *AcceptBroker* messages. Note that the *AcceptBroker* message is used as another handshake to finalize the choice of brokers. Once a BrokerAgent has been identified, the involved parties may contact the broker independently and have it sign the contract, retrieve the contract signature and notify the broker of repudiation occurrences if necessary. The messages used to accomplish this step are respectively *SignContract*, *GetContractSignature*, and *ContractRepudiated*.

Upon receiving the requests of the ConsumerAgents, the ProducerAgent forms a result set that is communicated back to the ConsumerAgent based on its inventory, if applicable. At the ConsumerAgent's discretion, the ProducerAgent may be contacted once more to engage in negotiation or to formulate a viable contract, as described earlier. Both parties will present such contracts to a BrokerAgent chosen based on the recommendation of the DirectoryAgent and the security requirements of both parties.

Once both copies of a contract are received, the BrokerAgent verifies that they are identical. It will then digitally sign and return the signed copies to their owners upon receiving a *GetContractSignature*. Any further communication between the two parties that have brokered a deal must go through the BrokerAgent if such communication may affect the trust value of one of the agents. This is achieved by implementing the *ContractRepudiated* message as a special kind of message capable of encapsulating another message. *ContractRepudiated* can only be instantiated by an agent, and the BrokerAgent always ensures that the instantiating agent is one of the entities of the contract. Moreover, it is required that every contract provides the means for the Broker to determine foul play by specifying future messages that may be communicated along with the conditions that would signal a breach of the contract. If the BrokerAgent detects any breach of contract, it reports any such instances to the SecurityAgent along with the specified penalty associated with the violation. The associated penalty for violations is an integral part of contracts. Figure 1 provides a pictorial view of the system architecture and of the different interactions amongst the agents.

4. Exchange System Framework

In implementing the framework of the proposed system, we chose to use the Aglet platform due to the fact that it is available as open source, has received great press coverage and has also been the subject of recent work aimed at improving the security of the platform. One such study has resulted in the introduction of a new server, aptly named Secure Aglet Server (SAS) [8]. SAS provides secured communication through SSL, makes use of the Java Cryptographic Extension (JCE) [7] to support the notion of Read-Only Data thereby providing agents in the system with the ability to verify the integrity of collected data. Furthermore, SAS introduces the notion of a MonitorAglet capable of preventing Aglets from initiating a Denial of Service (DoS) attack on host through seemingly normal transition from one lifecycle state to another. The security features present in SAS are of primordial importance to the herein proposed exchange system as both marketplaces and agents are able to subsist in a suitably secure environment.

As the implemented system framework is based on SAS, the interacting agents in the web

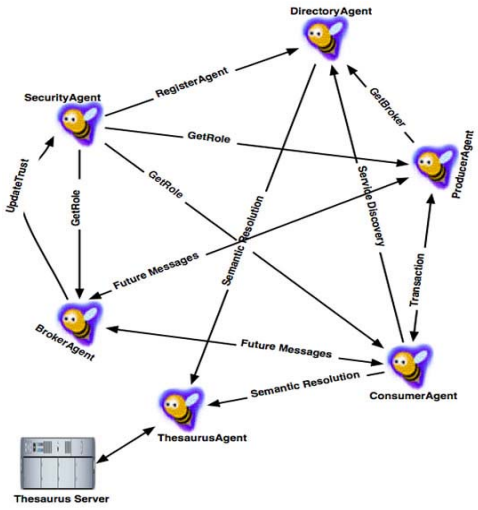

Figure 1: Interaction amongst system entities

exchange system are implemented as Aglets. Interaction occurs through the creation and passing of the messages described in section 3.2. The SecurityAgent uses the listener interfaces provided in the Aglet framework to monitor the arrival of new agents in the marketplace and register such agents. A thorough understanding of how the system was realized can be acquired through the case study presented in the following

section (4.1) describing the representation of contracts and services as well as other concepts left as implementation details in the proposed framework.

4.1. Pharmaceutical Marketplace: A Case Study

In order to ascertain the power and flexibility of the prototyped web service exchange system, we opted for the realization of a real-world application able to interface with both humans and agents alike. For simplicity, we considered a marketplace where the producers and consumers are exclusively interested in pharmaceutical products. A marketplace in our implementation is a host providing the necessary environment for the exchange system. A marketplace is thus made up of all the entities in the system including 0 or more Consumer and Producer agents at any given time. We herein present the design decisions that guided our implementation.

The web service exchange system framework provided the backbone of our application. Google Web Toolkit (GWT) [6] was also used to implement the web user interface component of the marketplace. Our design decisions were focused on issues not addressed in the framework as an attempt to support flexible application development. Such issues surfaced in our implementation as we represented the notions of services, schemas, and contracts. A discussion of the implementation of the aforementioned notions follows. We also discuss how transactions and the use of the thesaurus in the prototype were realized.

4.1.1. Broker Services. Once the SecurityAgent has determined that a BrokerAgent has the right to execute in the system, it signals the DirectoryAgent of the existence of the broker. The DirectoryAgent then contacts the BrokerAgent to retrieve the services provided by the broker, to be used in recommending brokers in future transactions. The definition of what type of services that a broker may provide required that we looked at not only the functions of a broker but also the expectations of consumers and producers. Based on the functions of a broker, a service is simply the algorithm that the broker uses to digitally sign a contract. The knowledge of what algorithm will be used is important to the parties involved as it determines the likeliness that a signature could be forged. From the standpoint of consumers and producers, we noted that two extra components come into play: The first one is clear-cut as it pertains to the length of time that a broker is willing to act as a mediator to a particular contract. This is an issue as a broker may decide to migrate, reduce its resource consumption or dispose of itself. The second component is less obvious and is dependent upon the type of transaction being conducted. A broker might specialize in brokering deals related to auctions, while another one might specialize in cash transactions and so on. To sum up, a service in the pharmaceutical marketplace encapsulates three types of information; the first being the signature algorithm in use. The second is the length of time the records of contract will be maintained and finally the "expertise" of the broker in terms of transaction types.

4.1.2. ProducerAgent Schemas. Although the producers in the system can store their local data in any form such as in their local database, they use XML to format their data to be exchanged. In our prototype, all local data are stored in XML files. Figure 2 shows an example of XML data representing the producer's local data.

```
…
<producer>
<producer_name>producerA</producer_name>

<product>
<item>ID10000</item>
<category>Asthma</category>
<price>20.2</price>
<available_amount>447</available_amount>
<warranty>30 days</warranty>
<alternatePrice>16.61</alternatePrice>
</product>
…
```

Figure 2: Example of Producer's XML data

When contacted by the DirectoryAgent upon their arrival on a host, producers need to publish the categories of their available products and services they wish to advertise to the directory. This is referred to as *advertisement*. The advertisement may not cover all available product/service categories, in that no restriction is made on the producer to advertise all of its products to a DirectoryAgent. The advertisement to be sent to the DirectoryAgent can be of any format as long as the DirectoryAgent can recognize it. For instance, the advertisement in XML format can be recognized if its corresponding XML schema is provided. Note that heterogeneity of the terms in the advertisements can be reconciled using the

thesaurus. Within our implementation of the prototype, we opted, for simplicity, to have the format of the advertisements be uniform across all producers.

4.1.3. Representation of Contracts. In order to prototype the pharmaceutical marketplace, we had to have a clear definition of what constitutes a contract between consumers and producers. The obvious requirement is that the contract must identify both parties involved. Furthermore, as was briefly introduced in section 3.2, the contract needs to contain all information necessary to identify any possible breach and moreover. In our prototype, the contract consists of the IDs of the producers and consumers along with a set of messages that may be communicated between the two parties in the future. Within the scope of the pharmaceutical marketplace, we implemented two such messages to represent the fact that a consumer may ask for a refund or an exchange. Either one of these requests should only be valid as long as the return policy expiration date has not been reached. The set of messages that may be communicated in the future also encapsulate the answer expected of the receiver so that the broker may detect violations. Moreover, the messages also contain the associated penalty. Lastly, the contract contains two extra pieces of information namely the broker of the contract along with the requirements of the contract in terms of services that must be offered by the broker. Such information is contained within a contract to allow for either party to determine at any point during negotiations if a broker is appropriate to serve as a third party.

4.1.4. Exchange System Transactions. System transactions refer to the set of messages exchanged by consumers and producers in order to determine the availability of a service of interest, as shown in figure 3. In our prototype, producers offer various medicines organized by categories. Upon arrival in a new marketplace, once the consumer has been allowed to execute, it contacts the directory agent with the term specifying the product of interest. Upon receiving the reply from the DirectoryAgent consisting of all qualified producers, the consumer may then decide to contact one or more producers using the producers' semantics in referring to the item. At this point, the consumer sends an *IsItemAvailable* message to the producer to determine the availability of the item of interest. If the item is available, a *GetPrice* message may follow. The consumer may still send *ProposedPrice* messages to try and negotiate a better price. If the consumer wishes to purchase the service/item, a *GetContract* is sent to the producer, thereby initiating a transaction between the consumer and the producer. The consumer may not accept the contract and propose a new one using the *ProposedContract* message. If the consumer accepts the contract or the *ProposedContract* message returns true; the consumer may then choose to initialize the broker handshake, accomplished through the *AcceptContract* messages exchanged between both parties. At this point, the transaction moves to the second step.

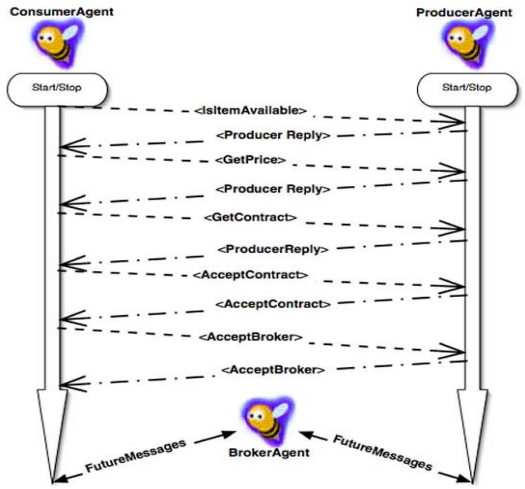

Figure 3: Flowchart of Consumer and Producer Agents Interaction

Once both parties have accepted a contract, the consumer may then request for a broker as a trusted third party using *GetBroker*. The producer replies with a broker; the consumer has the option of proposing a broker other than the one suggested by the producer. In deciding whether to accept or reject a broker, the consumer contacts the DirectoryAgent and ensures that the proposed broker is a registered broker and also obtains the services that the broker may provide to ensure that it meets the contract requirements. This is accomplished through the *IsRegisteredBroker* message. Note that the producer takes similar steps if it receives a *ProposedBroker* message. Once a broker has been agreed upon, the consumer must initiate an *AcceptBroker* handshake, which moves the transaction to the next step. At this point in the transaction, the communication between the two entities that affect the transaction is through the

broker using *ContractRepudiated*. The transaction remains in the current step until it becomes invalid (e.g. warranty expired). One feature implemented in transactions is that they have a *TimeToLive* attribute, designed to prevent either parties from indefinitely waiting to complete a handshake. The *TimeToLive* attribute allows for a transaction to be cancelled if it has lasted longer than the attribute and has not yet reached an agreement on a broker.

4.1.5 Thesaurus. We use the thesaurus-based search engine identical to those presented in MAMDAS [9] and MEDTHES [19]. However, the medical terms included in MEDTHES' thesaurus could not directly be applied to our system. Thus, we enhanced the thesaurus using terms obtained from WordNet [11, 18] thesaurus. The added terms relate to various symptoms and abnormalities, which are used as the categories of products available from the producers. For example, the term 'migraine' is used as a product category. Different producers may offer different products for migraine; furthermore, a producer may have more than one product related to migraine. Figure 4 gives an example of term included in the thesaurus. It is shown that the term 'Headache' has 'Ache' as its broader term (BT), and has 'Migraine', 'Hemicrania', 'Sinus', 'Tension headache', and 'Histamine headache' as its narrower term (NT). In our implementation, we have only one semantic category (SC), namely medicine.

```
Headache
BT: Ache
NT: Migraine
NT: Hemicrania
NT: Sinus
NT: Tension headache
NT: Histamine headache
SC: Medicine
```

Figure 4: Example of Term in Thesaurus

4.2. Implementation Results

This section presents some results from our prototype. Once a user decides to dispatch an agent to a marketplace to search for an item of interest, the dispatched agent operates autonomously to accomplish the task. The agent collects information such as the location of the marketplace, the name (category) of the item, the price and quantity desired as well as the maximum semantic distance to be considered. Upon arrival to the marketplace, the agent locates the suitable producers in order to engage in interactions. If the user's agent finds an item whose category is within the semantic distance specified and with a price cheaper than or equal to that specified, the agent emulates the actual purchasing of the item. Within our implementation, the purchased item is the cheapest item available. Note that other implementations could focus on accuracy by choosing the closest match. A web report is then generated and presented to the user once the agent migrates back to its original location as shown in figure 5.

Figure 5: Consumer Agent Web Report

The producer's product data in XML format is decorated with XSL and presented as the web catalog in the producer's website as shown in figure 6. From the producer's website, humans can view the available items in the inventory and decide whether or not to purchase an item. The web interface presents to the user pertinent information such as warranty information, quantity available, as well as purchasing price. The combination of agents automatically resolving semantic heterogeneity to determine the item of interest along with the producer's website encapsulates the notion of semantic web. This is achieved by allowing the same

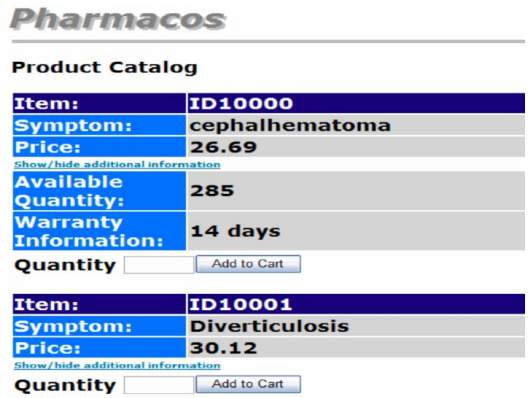

Figure 6: Producer Agent Interface

information to be intelligently processed by both humans and agents.

5. Conclusion

This work has introduced a novel approach towards bridging the gap between how humans and agents process information on the Internet. The proposed system makes use of a thesaurus to resolve semantic heterogeneities, thereby allowing agents to extract the meaning of information being processed. This results in a highly flexible system capable of hosting agent entities with possibly various representations of related data. A framework has been implemented to support the basic mechanisms for entities within the system. Furthermore, the prototype of a pharmaceutical marketplace has been implemented on the framework showcasing the flexibility and robustness of the system.

6. Acknowledgments

The National Science Foundation under the contract IIS-0324835 in part has supported this work.

7. References

[1] T. Bray, J. Paoli, J., C. Sperberg-McQueen, E. Maler, F. Yergeau. Extensible Markup Language (XML) 1.0. W3C Recommendation 16 August 2006. Cambridge, MA: W3C. Retrieved June 5th 2007, from http://www.w3.org/XML/Core/#Publications

[2] D. Brickley, and R. Guha. Resource description framework (RDF) schema specification. W3C Recommendation 10 February 2004 Cambridge, MA: W3C. Retrieved June 5th 2007, from http://www.w3.org/TR/rdf-schema.

[3] C. Byrne, S.A. McCracken, An adaptive thesaurus employing semantic distance, relational inheritance and post-coordination for linguistic support of information search and retrieval, Journal of Information Science, 25(2), 1999.

[4] R. Chapman (revised), Roget's International Thesaurus (4th edition), New York: Thomas Y. Crowell Company, 1977.

[5] N. Gibbins, S. Harris, N. Shadbolt, Agent-based Web Services, Proceedings of International World Wide Web Conference (WWW'03), page(s) 710–717.

[6] Google Web Toolkit, Google Inc. Retrieved June 1st 2007, http://code.google.com/webtoolkit/

[7] JCE Internet Reference Guide. (n.d). Retrieved December 5th 2006, from http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html

[8] E. Jean, Y. Jiao, A.R. Hurson, and T.E. Potok, "SAS: A secure aglet server," In *Proc. of Computer Security Conference* 2007,

[9] Y. Jiao and A.R. Hurson, Application of mobile agents in mobile data access systems – A prototype. Journal of Database Management, 15(4), page(s) 1-24.

[10] O. Lassila, and R. Swick, Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation 10 February 2004 Cambridge, MA: W3C. Retrieved June 5th 2007, from http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

[11] G.A. Miller (Ed) Five papers on WordNet, International Journal of Lexicology, 3(4).

[12] J. Pan, and I. Horrocks. "Metamodeling architecture of web ontology languages" In: *Proceedings. of the First Semantic Web Working Symposium* SWWS 2001, Stanford University, California. pp. 131-149. Heidelberg: Sprnger-Verlap Heidelberg.

[13] L. Shiyong, D. Ming and F. Farshad (2002) "The Semantic Web: opportunities and challenges for next-generation Web applications." Information Research 7(4), Available at: http://InformationR.net/ir/7-4/paper134.html

[14] S. Suwanmanee, P. Thiran, OWL-Based Approach for Semantic Interoperability, Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA' 05), 2005, page(s) 145-150.

[15] A.M. Tjoa, A. Andjomshoaa, F. Shayeganfar, and R. Wagner, Semantic Web challenges and new requirements, In Sixteenth International Workshop on Database and Expert Systems Applications, 22-26 Aug 2005.

[16] R.F. Tomaz, S. Labidi, B. Wanghon, A semantic matching method for clustering traders in B2B systems, Proceedings. First Latin American Web Congress, 2003, page(s): 144 – 153

[17] R.F. Tomaz, S. Labidi, Increasing matchmaking semantics in intelligent commerce system, Proceedings. IEEE/WIC International Conference on Web Intelligence (WI' 2003), page(s): 616 – 619.

[18] WordNet, A lexical Database for the English, Cognitive Science Laboratory, Princeton University. Retrieved June 10th 2007, from http://wordnet.princeton.edu/

[19] P. Yan, Y. Jiao, A.R. Hurson, T.E. Potok, Semantic-Based Information Retrieval of Biomedical Data, SAC'06, page(s) 1700-1704.