Multi-Agent System for DAML based Web Services (MWS)

JOHN O. AKINYELE

Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD. 21250
jakiny1@umbc.edu

ABSTRACT

The term web services has been gaining visibility. It is defined as a new breed of web application that are self-contained, self-describing, modular applications that can be published, located, and invoked across the web [8] and they have the ability to perform functions, which can be anything from simple requests to complicated business process. The simplicity that web services suggest has sparked off interest in utilizing software agents in accessing them. Furthermore, web services that incorporates some type of intelligent design in its definition would potentially reduce communication impediments among agents and would also reduce the necessity for a lot of intelligence in the software agent itself.

INTRODUCTION

The world wide web as we know it today has evolved from simply being a repository of information to a major commodity in the economic industry. In recent times, we have seen major market players become more visible on the web by providing electronic commerce (e-commerce) services such as stock-trading sites, auction sites, and storefronts that include bookstores, apparel stores amongst others. This increase of major market activity on the web has created new marketing and cooperating strategies such as Business to Business (B2B) and Business to Customers (B2C). These terms are becoming increasingly popular and indicate the rising need for businesses to automate cooperation. B2B transactions typically involve businesses accessing their services via web-front ends. This strategy makes it convenient for these businesses to create automated applications that access these online services.

While these ideas are becoming more attractive, they can prove difficult to implemented. This difficulty stems from the limitations of HTML\(^1\) which is the language that was originally used to mark-up online information. HTML is the *lingua franca* for publishing hypertext on the World Wide Web. It is a non-proprietary format based upon SGML, and can be created and processed by a wide range of tools, from simple plain text editors. The focus of HTML is to describe the presentation of the contents of a document without implying any description of the content. An HTML document, by itself, provides no basis for content determination or comparison to another HTML document without some type of external knowledge gained by comparison made by a human interface. This has been somewhat of an imediment to automated online-services because developers need to gain some knowledge of the services they are accessing before developing the automated applications that utilize them. It becomes even more difficult when companies give their web-sites a “face-lift” by re-designing the presentation.

\(^1\) Hyper Text Markup Language – [http://www.w3.org/markup](http://www.w3.org/markup)
The limiting nature of HTML prompted the introduction of a new markup language XML\(^2\). XML is the universal format for structured documents and data on the web because it extends HTML by allowing web developers to define their own markup tags and to separately define the presentation of that document. XML is a good idea and has its benefits; however, it still does not make B2B convenient. Web developers can certainly create their own markup while separating the description of the presentation, but it lacks global consistency in the definition of these markups. In other words, various web developers might use the same markups to describe contents that have different interpretations. This has further prompted the definition of another markup language RDF\(^3\). RDF integrates a variety of applications from library catalogs and world-wide directories to syndication and aggregation of news, software, and content to personal collections of music, photos, and events using XML as an interchange syntax. The RDF specifications provide a lightweight ontology system to support the exchange of knowledge on the Web. An ontology is described as an explicit formal specification of how to represent the objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them. RDF, therefore, provides a means through which web sites with meaningful and globally consistent contents can be defined.

RDF is the brains behind the idea of the semantic web. The semantic web is the idea of having data on the web defined and linked in a way that it can be used by machines for various applications [6]. The semantic web portrays the idea of the expression of meaning and of knowledge representation in web documents by using available and previously defined ontologies. This would present a medium where automated intelligent applications (software agents) can access this information or service.

The idea of the work described in this paper is to create a community where agents can congregate and access web services. This community can be described as a “shopping mall” where services “advertises” their web services in the mall directory to make them accessible to consumer/personal agents based on some guarantees provided by the mall agent. The mall agent categorizes the advertisement of each service based on the product it knows the service provides. This allows personal agents to access services that advertise products that interests them – very much like what is to be expected in a real shopping mall.

**BACKGROUND**

Since the advent of XML and RDF there has been various efforts at creating web services. Such efforts include DARPA\(^4\) Agent Markup Language (DAML) which is a specific kind of RDF markup [1]. DAML is being developed as an extension to XML and RDF by using ontologies to provide a powerful way to describe objects and their relationships to other objects [1]. DAML is becoming increasingly popular and is being

---

\(^2\) Extended Markup Language – [http://www.w3.org/xml](http://www.w3.org/xml)
\(^3\) Resource Description Framework – [http://www.w3.org/rdf](http://www.w3.org/rdf)
\(^4\) Defense Advanced Research Projects Agency
used in creating applications in the academic community with applications such as ITTALKS\textsuperscript{5} developed UMBC, WebScripter\textsuperscript{6} developed Information Sciences Institute.

\section*{DAML-S}

Another effort within the DAML project is DAML-S which is an ontology of services that is based on DAML. The idea of DAML-S is to enable the creation of services that have particular properties which software agents can discover, invoke, compose, and monitor [3]. DAML-S describes a \textit{Service} object that is an upper ontology for web services. This class contain properties that are normally associated with all kinds of services [3]. The Service class seeks to answer three major questions about the presented service:

1. What does the service require?
2. How does it work?
3. How is it used?

The first property of the service object, \textit{presents}, answers the first question by presenting a \textit{Profile} object. The service profile tells what the service does by giving the type of information needed by a service-seeking agent to determine whether the service meets its needs [3]. Two important subclasses of the profile class is the \textit{Advertisement} class – used by services in “advertising” services and the \textit{Request} class – used by agents in specifying their requests. Expectedly, the profile contains the contact information for the service provider or the service requester and other parameters. The \textit{Advertisement} object “advertises” all the range of input parameters that is acceptable its service models and also all the range of output parameters that can be expected. The \textit{Request} object details the inputs which are the requests that the user wishes to purchase.

The second property of the service object, \textit{describedBy}, answers the second question by presenting a \textit{ServiceModel} object. The service model defines how the service works by describing what happens when the service is carried out [3]. This object is very instrumental in defining the processes used by the service. It is the equivalent of a “skeleton” and/or “stub” object that is very common to remote usage of objects. It describes the input and output parameters, conditions, effects for the processes and also the sequence in which they can be invoked. One of the best advantage that this object provides is the ability to use it to describe processes that range over various domains because the implementation of the process is irrelevant. The process ontology include such objects as Process, CompositeProcess, Sequence, Condition, If-Then-Else, Iterate, Repeat-Until among others.

The third property of the service class, \textit{support}, answers the third question by presenting a \textit{ServiceGrounding} class. The service grounding specifies the details of how an agent can access a service [3]. The grounding will specify the communication protocols and

\textsuperscript{5} ITTALKS is a web portal that offers access to information about talks, seminars, colloquia, and other IT related events – http://www.ittalks.org
\textsuperscript{6} Application used to assemble reports extracting and fusing information from multiple, heterogeneous web sources – created from DAML files.
other service-specific details that will be used in accessing the service. The grounding will also specify the details of data exchange.

Little information about the service grounding has been presented in the current release of DAML-S. The authors of the project expect that a relatively small set of groundings will come to be widely used in conjunction with services such that groundings will be specified at various well-known URIs [3]. For the purposes of this project, the grounding will be done via JADE. In other words, agents running on a specified JADE platform would exchange DAML strings and/or files that represent an invocation of a service process.

**MWS**

MWS is a Multi-Agent System for DAML-based Web Services. It specifically uses services based on the DAML-S markup language. The closest description is that it is a “shopping mall” where agents congregate to access services that the mall provides. The mall provides a list of services that are categorized by the product type so that agents can request a list of services that offer particular products. The implementation of the system is based on the JADE platform. In implementing this system, some assumptions are made. It is assumed that it is safe for agents to exchange data and perform transactions with other agents because a more elaborate implementation would require security layering in exchanging data and also utilize some encryption scheme. Some other things such as revenue generation are rather assumed than implemented. The main entry point to the mall is the mall manager. It contains a list of services that provide some services. A JADE platform could contain multiple “malls” by spawning multiple mall managers. The following is a description of the entities in the system.

**SERVICES**

The mall defines an ontology that includes a subclass of the Service object called MallService. Its profile is inherited from the Advertisement object and is called MallAdvertisement. The profile contains contact information for the provider of the service, the category of products and the type of service. The MallAdvertisement class requires its service category property to specify a product list which is a list of product types that the service offers or sells. This list is instrumental in the categorization that occurs when the service seeks to register itself in the mall community. The service also contains service model and grounding that the personal agents can use to decide if the service is accessible or not.

**ONTOGONY PARSER**

The DAML-S project does not have yet have a parser developed for it, but any DAML/RDF parser would suffice. The parser implemented for this project allows agents to query the properties of services and also to perform matches between DAML encoded files. The parser can be used by customer agents seeking to access services and by service agents in checking the rules governing processes that customer agents can invoke.

**MALL MANAGER**

The first type of agent is the Mall Manager. It is responsible for “book-keeping” activities of web services and provides a directory of services for the personal agents that wish to trade in the mall. The Mall Manager makes a guarantee that the services in its directory
provide certain products. The mall makes this guarantee by providing an ontology of products that services should provide. This idea is similar to what one might expect upon entering a shopping mall.

**PERSONAL AGENTS**

The second type of agent is the Personal Agent. This agent represents a customer that wishes to trade in the mall. This agent contains a DAML encoded profile of the customer which describes the customer's request and other information required to enable it to trade. When the agent starts up, it requests a list of services that matches its requests from the Mall Manager. It then traverses the list to determine which services actually provide the products it seeks. This determination includes checking if there is understanding in payment methods, shipment methods and communication methods. Services define acceptable payments and shipping methods, so it is left to the personal agents to verify that it can access services with the parameters (input and output) listed in its profile.

**SERVICE AGENTS**

The third type of agent is the Service Agent. This can be an optional agent if the service decides to complete transactions with an agent. The service agent initially seeks to register it's service with the Mall Manager by providing the mall name and the Uniform Resource Locator (URI) of its service. Once it receives a registration confirmation it can deploy itself to await agents that seek to perform transactions with it.

**IMPLEMENTATION**

The implementation of the system was done with the Java 2 programming language in conjunction with multi-agent platforms and expert system libraries.

**JADE**

JADE is the Java Agent Development Framework. It is a software framework fully implemented in Java which simplifies the implementation of multi-agent systems through a FIPA compliant middleware. The agents in this system are implemented based on JADE.

**DAMLJessKB**

DAMLJessKB is a DAML Jess Knowledge Base package developed with the Java programming language. The objective of this package is to facilitate the reading of DAML files and allowing the user to query on that information. It reads in the DAML file as a collection of RDF triples via an RDF parser and uses Java Expert Shell System (JESS) as a forward chaining production system which carries out the rules of the DAML language. The implementation of the ontology parser is based on the DAMLJessKB.

**ARCHITECTURE**

*Inter-agent communication*

For the purposes of this project, inter-agent communication is done via DAML encoded strings and files. Agents exchange DAML encoded strings and/or files so that each agent can use the ontology parser to decide what to do. The Mall has its own service that describes the process to be used in registering a service in its mall directory and
also the process to be used in requesting services from its mall directory. Service and Personal agents can access this service definition which would be used in generating the request to the mall agent.

**AgentMall class**

The AgentMall class is an agent that implements the mall manager. It is a subclass of the *jade.core.Agent* class. This object contains a directory of services that have registered themselves. The primary responsibility of this object is to accept the registration of services and also to suggest, to personal agents, a list of services that would possibly provide the type of services they need. The AgentMall categorizes the services into known categories to assist it in making appropriate suggestions to the clients.

**ServiceAgent class**

The ServiceAgent class implements the service agents that wish to register its services in the mall directory and to optionally await requests from customers. It is a subclass of the *jade.core.Agent* class. For the purposes of this project, all services will utilize service agents in processing requests via DAML encoded strings. In other words, services used in this experimentation will specify a grounding that supplies the jade location of its agent. The reason for this choice is that the implementers of DAML-S have not explicitly written a specification for grounding methodology, choosing rather to allow developers to develop grounding methodologies so that useful ones become more acceptable.

**PersonalAgent class**

The PersonalAgent class implements the customer agents that wish to trade in the mall. It is a subclass of the *jade.core.Agent* class. The primary functionality of this object is to process the user’s request and determine which service categories are needed to fulfill the requests. It does this by interrogating the categories of the products contained in it’s request and it uses this as a criteria in requesting a service list from the mall manager. Upon obtaining a list of services, it interrogates the service model and grounding specified for the service and determines if it should purchase the product or not. It utilizes the DAMLQuery object for all its interrogation.

**DAMLQuery class**

The DAMLQuery class implements the ontology parser. It utilizes the DAMLJessKB package and some jess rules and templates. This object is used in querying expected information about a DAML-S file. It is also used to generate an appropriate request and/or response to a service after it interrogates the process which the service supports.

**Discussion**

The choice of DAML as a means of encoding services and requests reduces the necessity of implementing a lot of intelligence in the software agent itself. By its very nature, a DAML service can use well established ontologies to describe its services and processes such that customer agents can easily interrogate and determine the feasibility of a service in meeting its needs. As I mentioned earlier, the service model represents a
“skeleton” or “stub” object that describes the methods/function calls that the service has. The downside of this is that generating the service model objects would be a separate process from the actual implementation. This could lead to seemingly minute errors that would compromise its integrity such as mislabeling input and output parameters. There would have to be some converters that convert the actual implementation into DAML service models. This can quickly lead into complications as companies that use different languages for their implementations would need to have converters for each language.

**Future Work**

A desirable, and initially intended, characteristic of the system is an implementation of a collaboration scheme between the personal agents. Agents might find situations in which they decide to collaborate because of discounts for bulk purchases advertised by the service or for reduced shipping costs in the event of a bulk purchases. This can be implemented by specifying a process in finding out the shipping costs, so that a customer agent could “ask” how much it would cost to buy/and or ship a given number of items. The agent would be able to figure out if it would be more profitable if bulk purchases are made. This would trigger it to seek to negotiate with other agents on the means of collaboration. This could be accomplished if the service advertises discounts and specials in its advertisements. The collaboration would be implemented via Intermediate agents which the mall manager spawns on behalf of an agent seeking collaboration. The mall manager would keep a list of intermediate agents and their intentions so that agents seeking similar services would collaborate via a similar intermediate agent. The initial agent that made the request for the intermediate agent specifies a wait-time value that determines how long it wishes to wait for agents that desire to collaborate.

**Related Work**

There is a lot of work in the area of multi-agent trading systems that use semantic web technologies and others that exclusively use agent communication technologies.

**Trading Agents Scenario**

In a paper that researched on improving the scalability of multi-agent systems, Phillip J. Turner and Nicholas R. Jennings describe a trading agents scenario. This work, to an extent, resembles the work described in this paper. It presents an agent community or “market” that consists of three types of agents: Supplier agents ; Customer agents that are capable of dynamically building preferences for suppliers on a per commodity and general basis and propagating information about suppliers and their preferences to other customers. Customer agents are also capable of forming co-operative groups to make collective bulk purchases and can also maintain models of suppliers wares and prices. The co-operation of agents can be achieved via Intermediate agents or communication between customer agents themselves. In exploring and measuring scalability, the agents dynamically choose which forms of communication to use. The choice of the form would be based on a vote by the agents so that the chosen form would improve the performance of the whole system at that particular instance. Communication is not implemented a semantic web service implementation.
**AGENT CITIES**

Agent cities is a new initiative aiming to build a worldwide, publicly accessible, test bed for the deployment of dynamic, composable, agent based services [4]. This agents in such a system would not necessarily be trading nor would they exclusively be trading agents. It will rather be built on a wealth of innovative technologies such as agent technology, Semantic Web technologies, UDDI discovery services, eBusiness standards and Grid Computing. Some examples of application areas that are being planned range from eHealth and eLearning to manufacturing control, digital libraries, travel and entertainment services [4]. On a very small scale, the MWS system bears some resemblance to this initiative.

**SEMANTIC WEB TECHNOLOGIES**

There are other semantic web technology efforts such as Web Services Description Language (WSDL). WDSL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information [7]. There are other efforts such as ones that use DAML to create architectures that can be used for querying the web [10]. These efforts bear a lot of resemblance to the objectives of the DAML-S project and would probably provide similar benefits when integrated with expert system packages.

**CONCLUSION**

The problem with any standard is global acceptability. The industry is replete with quite a number of different web service implementations. However, as the benefits of the different service implementations are evaluated and analyzed the industry should begin to move towards an acceptable standard, or at the very least a few of them. Such global acceptability will foster the utility of software agents in accessing the services. Software agent developers would develop libraries, packages and tools that would make it relatively easy for webmasters to describe their services and for personal agents to access those services.
REFERENCES


2. http://www.daml.org/services/daml-s

3. DAML-S: Semantic Markup for Web Services http://www.daml.org/services/daml-s/2001/05/daml-s.html


6. Tim Berners-Lee, James Hendler and Ora Lassila. The Semantic Web

7. R. Scott Cost et al. ITTALKS: A Case Study in the Semantic Web and DAML


9. Phillip J. Turner and Nicholas R. Jennings. Improving the Scalability of Multi-agent Systems