Agent Approach for Service Discovery and Utilization

Paul Palathingal
Research Associate
Advanced Software Engineering Research Group
Bldg 6025, Room 14L, PO Box 2008
Oak Ridge National Laboratory
Oakridge, TN
Ph: 865-574-9790
Email: palathingalp@ornl.gov

Sandeep Chandra
Research Assistant
Advanced Software Engineering Research Group
Bldg 6025, Room 10K, PO Box 2008
Oak Ridge National Laboratory
Oakridge, TN
Ph: 865-576-1339
Email: chandras@ornl.gov

Abstract

There is an extensive set of published and usable services on the internet. Human based approach to discover and utilize these services is not only time consuming, but also requires continuous user interaction. In this paper we demonstrate the use of agent technology and Web Service standards to enable automatic service discovery and utilization. Our approach called the Agent Approach for Service Discovery and Utilization (AASDU) focuses on using light weight autonomous agents, built into a multi-agent community called the Multi Agent Referral System and Web Service standards namely UDDI, SOAP, WSDL and XML.

In the AASDU approach, agents interact with the end user to discover services, to specific queries, and efficiently manage their utilization. It uses intrinsic multi-agent properties to allow agents to communicate and cooperate with one another. As the system is agent driven, each agent conforms to a communication protocol that allows it to send and receive messages from another agent, without needing to know the address of the receiving agent. This paper also demonstrates the interoperability achieved between heterogeneous software components through the use of Web Service standards and protocols. We describe in detail the agent technology, Web Service standards and the AASDU approach.

1 Introduction

Consider the following scenario. A Geographical Information System (GIS) analyst is analyzing an environmental project for a particular geographical region. They would need the following data 1. Orthographic images of the region, 2. Articles or documents on any activity or construction in the region and 3. A visualization tool for this data. For these tools to be located and used, they need to be registered as services with public registries (e.g. UDDI) [1] on the internet. Current approach requires the user to search for these services manually. In the near future, similar services are expected to grow exponentially thereby making manual search extremely time consuming and laborious.

To address these issues we propose an agent-based approach through two concepts namely *autonomous and intelligence*. Our approach is autonomous in that the agents act on behalf of the user. The agents search for heterogeneous services based on the user query. It entails intelligence because the agents are able to compose services based on a particular query and provide results to the user. In our GIS analyst scenario, the user agents talk to service registries, locating services that provide data sets 1, 2 and 3. The services are then negotiated, composed and invoked by a composing agent, which utilizes the Web Service protocols and returns the result to the end user.

2 Background

2.1 Agent Technology

Software agents are the subject of research in many inter-related fields. They are long-lived, persistent computations that can perceive, reason, act, and communicate [2]. They have the ability to make decisions independently without human intervention and without influence from other agents.

Agent architectures provide several advantages over existing object-oriented technologies. In object-oriented systems, objects communicate through messages. The sender object must know the address of the receiver object, and the public methods of the receiver object. On the other hand, agents conform to a communication protocol and language (FIPA ACL) that allows an agent to send a message to agent(s) without needing to know the address of the agent(s) or the specific methods available to the agent. This allows agents to move, yet still be in contact with other agents, and allows for agents to broadcast requests to some or all other agents.

Current agent research has concentrated on building exemplar agent systems, defining theory of agent behavior and inter-agent communications. Most resulting systems are closed systems which work well within an enterprise or in a homogeneous environment like VIPAR [3]. There has been less emphasis on the interoperation of agents in a heterogeneous and ever changing environment like the web. Also, agent systems that

work across enterprise boundaries need to overcome several challenges in service discovery, communication and utilization. Right now there are only partial solutions to the above problem, and mainly applicable to a closed multi-agent system.

2.2 Web Service

Web Services are software components that are self-containing, self-describing modular applications that can be published, located, and invoked across the Web. They allow applications to interoperate in a loosely coupled environment, discovering, and connecting dynamically to services without any previous agreements having been established between them. More importantly, a Web Service may combine several applications that a user needs. For the end-user, however, the entire infrastructure will appear as a single application. Web Services encompass just about any application available over or delivered via the Web using standard protocols like Simple Object Access Protocol (SOAP) [4], Universal Description, Discovery, and Integration (UDDI), Web Services Description Language (WSDL) [5] and Extensible Markup Language (XML) [6]. The Web Service itself is really nothing more than a software program; say a java server page or a java servlet. The Web Services architecture is simply a wrapper for accessing this pre-existing code in a platform and language-independent manner. These technologies provide a standard means of communication among different software applications involved in presenting dynamic context-driven information to the user.

Current work in the area of Web Services comprises of vendors and service providers developing and publishing specialized services with private or public registries depending on its usage [7, 8]. Also available are tools and standards that are built on top of current standards to enable various inter-organizational and inter-disciplinary architectures. In some cases agent technology has been used in collaboration with Web Service standards, where semantic markup of Web services (DAMLS) enables a wide variety of agent technologies to automatically discover and utilize services [9].

3 Agent Approach for Service Discovery and Utilization (AASDU)

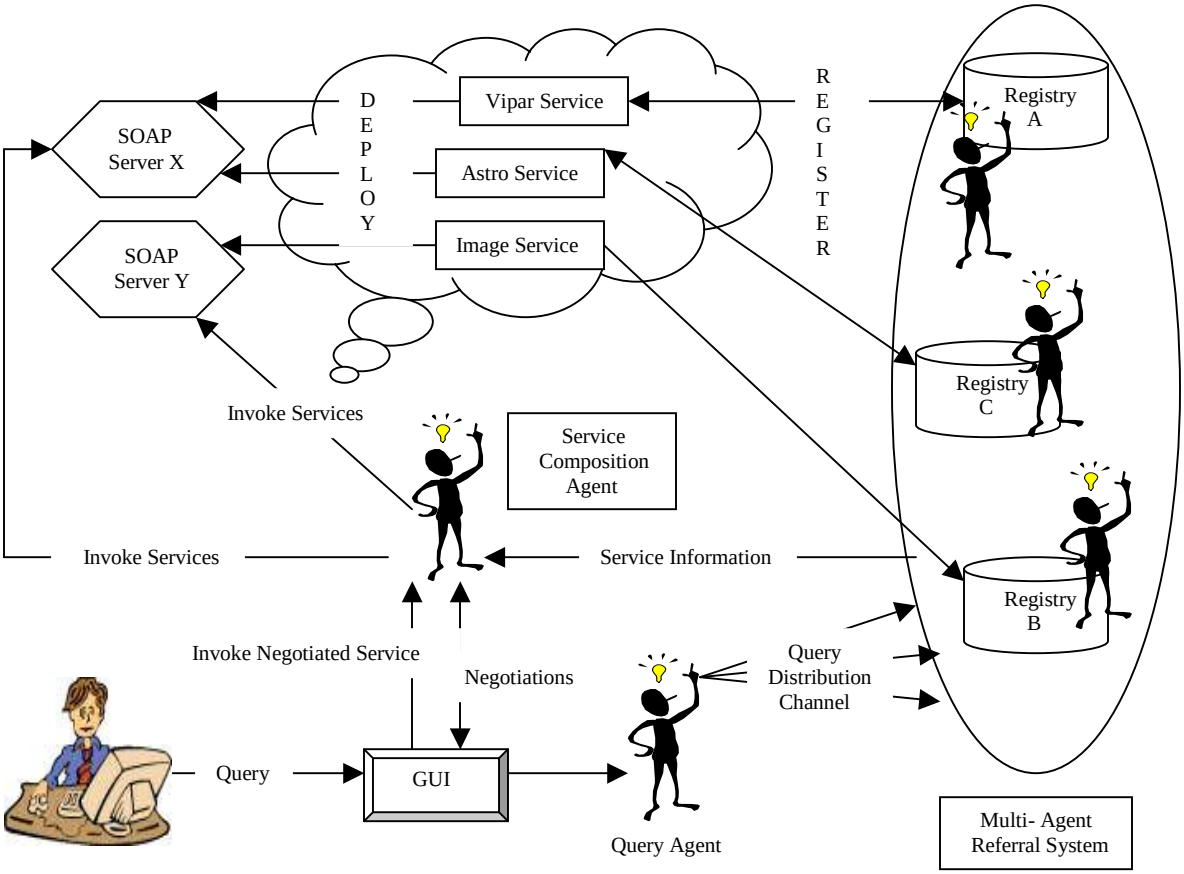

FIGURE 1

The AASDU approach consists of three core elements in its Architecture:

1. Client Graphical User Interface (GUI) and Query Analyzer Agent (QAA)
2. Multi-Agent Referral System and Service Lookup
3. Service Registry, Negotiation and Composition

Refer to figure 1 for a detailed architecture.

3.1 Graphical User Interface (GUI) and Query Analyzer Agent (QAA)

The GUI is integral to the approach as it allows the user to interact with the system. The GUI is implemented using java swing components. It provides a text field for the user to input their query. To assist agents in the query lookup users are provided with an option to select a specific domain. For example, in our GIS scenario the scientist might enter a query in the geographical analysis domain.

Service invocation requires the user to enter method parameters within the service. For example a data visualization service has a method called movie producer that requires the number of time steps of data from a simulation. To incorporate entering parameters for service invocation the GUI will have fields dynamically created for every input parameter required by the composition agent.

The query is processed by a QAA. The QAA looks at each query and extracts service information from them [10]. For example a query “satellite images in the Kentucky area” has the term “images” in it. The QAA looks at the query and infers the relevance to the “image to intelligence” web service that is registered on one of the registries.

The QAA uses a simple variant of the Term Frequency Inverse Document Frequency (TFIDF) approach to index the query and rank the results [11]. An indexing table contains a list of keywords in a particular domain. The agents from the multi-agent referral system described in the next section create agent vectors. Every vector contains a value for each keyword in the indexing table. Each agent is associated with a certain set of registries. If a particular keyword occurs in more than one registry on one or more services, the number of times it occurs is entered as the value in the agent vector.

The QAA processing occurs in three phases. In the first phase the query is run through a term stemmer that returns a query that is stemmed and contains only potential key words. A query vector is created based on the terms in the indexing table, where each scalar in the vector takes on values based on the number of occurrences of an indexing term in the query vector. In the next phase each word is compared to the agent vectors in the agent vector repository. The QAA uses the TFIDF approach to determine the similarity of a term to each agent vector. If the similarity is above a threshold value, the term is treated as a query to a service in the registry associated with that agent.

The QAA framework is shown in figure 2.

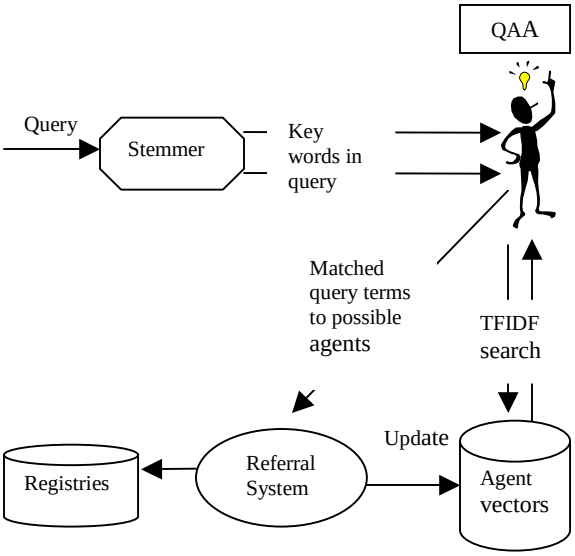

FIGURE 2

3.2 Multi-Agent Referral System and Service Lookup

We use a multi-agent referral system to evolve an agent social community that communicates with each other to effectively determine the service registry to look for a particular service or services in a particular domain. Each agent maintains a profile of itself and its immediate neighbors to assist in the referral chain [12]. The profile contains the Agent ID, Neighbor agent list with profile, Services in the registry associated with the agent and keywords in the registry associated with the agent. This is stored as a simple text file local to each agent.

When a term from the query is input to an agent, it first looks up its own services list in its profile. If it finds a match it will go out to the registry and lookup the service. If the term is a keyword, the agent looks up the keywords list in its profile. If it finds a match it

will go out to the registry, search through all the possible services on that registry and come up with a matching service. If however the term is in neither of the lists, the agent looks up its neighbor agent list and the profile associated with each neighbor. The neighbor profile contains a value called the knowledge value of a particular service or keyword. If any of the neighbor agents contains a knowledge value for the term and is above a threshold, the term is forwarded to that agent. To see the real usefulness of the referral system, if the neighbor agent has inadvertently advertised a wrong value for the knowledge value, it could look up in its own neighbor profile and check for possible neighbors who have a neighbor value for the query term. The matching neighbor agent list is returned to the querying agent, who adds the new agents to his neighbor profile and forwards the query term to the new agents. The referral chain is forwarded for a maximum of six steps. The referral system framework is shown in figure 3.

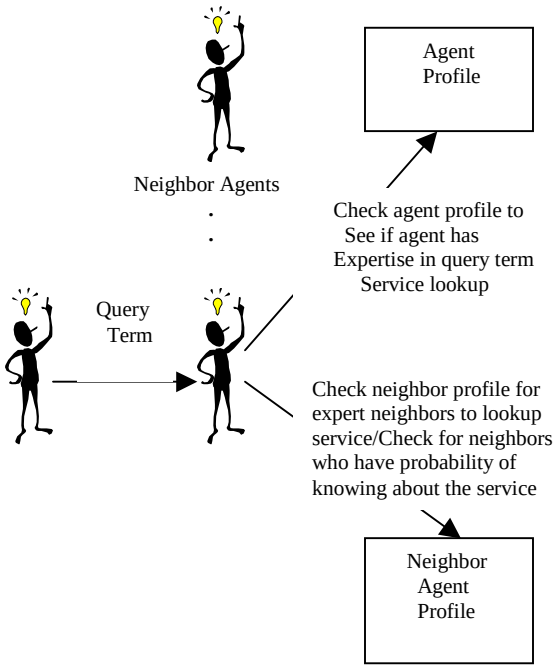

FIGURE 3

An example scenario of the referral system:

Agent ID	Profile	Immediate Neighbor Profile
1	Queries on Orthographic images. Looks up registry C	2, Queries on articles. Looks up Registry A.
2	Queries on Articles. Looks up registry B	3, Queries on Visualization tools. Looks up Registry B.
3	Queries on Visualization Tools. Looks up registry A	No Neighbors

Consider the case where agent 1 would like to locate services on visualization. 1 refers 2 to find that 2 has 3 in its immediate neighbor list and 3's profile matches queries on visualization. 1 then asks 3 for a registry name which it uses to look up services for visualization. Also at this point 3 is added to 1's immediate neighbor list. The results of the lookup are service information or description in a Web Service Definition Language format. The description contains service parameters such as 1. Service URL 2. Service methods and parameters 3. Other relevant service invocation information.

3.4 Service Registry and Deployment

Every Organization exposes and registers its services, which may have been developed on different platforms, with the UDDI registry. The service provider has to publish specifications, and the messages it can exchange, about the service namely access point (where it's been hosted and deployed), methods implemented, input/output parameters, negotiable quality of service parameters and other service related information. This can be a description in natural language, an XML document or in Web Services Definition

Language. The service is deployed to a SOAP router. The SOAP router is a Remote Procedure Call (RPC) router that accepts and invokes agent requests.

Each organization can access and use a template to create an agent that can easily join the multi agent community. The created agent will take the registry information with itself and join the multi agent community. This will help in evolving and growing the agent community with information on multiple registries. In essence, we will have an agent for every organization that would plug itself into the multi agent community.

The client can use the agent system to query the UDDI, find services of interest and compose and execute services. To find services our approach uses the Java API for XML Registries (JAXR), which provides a uniform and standard Java API for accessing different kinds of XML registries including the UDDI registries. The client need not know the implementation details of a service. The agent gathers service parameters, from the service description file, namely service name, method name, input method parameters and SOAP URL, and creates a SOAP call that is sent to the SOAP router. The SOAP router calls the appropriate service based on the parameters it receives in the SOAP message. Finally, it returns the result to the agent who passes it to the end user.

3.3 Service Negotiation

Consider a case where we have two web services that perform a clustering operation on documents. One service uses the genetic algorithm and the other uses a hierarchical

approach model in the background. Both these web services are registered with the UDDI registry and deployed to a SOAP router. Once the client inputs a query to look for services that perform clustering, the query agent distributes this to the agents in the multi-agent referral system. The agents in the multi-agent system interpret the query and search for the required service in their profile or their neighbor's profile. Once the services are found, the agents retrieve service information published in the WSDL and supply it to the service composition agent. The service composition agent carries out the negotiation with the client. Parameters that are part of the negotiation could be quality of service, response time, maximum load, etc. In our example two services will be retrieved. Their service details and negotiation parameters are passed to the service composition agent. The service composition agent would pass these parameters to end user and negotiate on a particular service to be selected for invocation. Once the end user selects a service out of the available services, the composing agent invokes the service. The service composition agent will also temporarily keep the details of the backup services and automatically invoke them incase the main service fails, thus demonstrating that the system is fail safe.

3.5 Service Composition

The service composition agent can assist the end user create a pipeline of services. Users can specify the services and the order in which they would want to invoke them. The composition agent will negotiate the services to invoke; it would validate the input and output parameters, check for data type matching and pipeline the services in the order.

Once the pipeline is created from the selected services the agent would invoke the service and pass the result to the client.

4 Discussion

There are a number of architectures that are trying to address the same problem of service discovery and utilization. One such work is the markup of web services in the DAML family of semantic web markup languages. The markup proposed enables agent technologies for web service discovery and execution [9]. Our approach is similar in the use of agents and service descriptions to support interaction with the services. We focus on quicker and more efficient lookup of web services. With the above approach laying much emphasis on markup, we concentrated our attention more towards the agents being associated with individual registries and the agent maintaining information of individual services in the registry. The referral system module provides easier lookup of services and greater scalability. Every time a new registry is created all it needs to do is create an agent in the referral community for itself. The agent gets plugged into the community with a default set of neighboring agents and their respective registries. Another module that impacts the service composition in particular is the query analyzer agent. Using a vector space model in the QAA guarantees quicker breakdown of a query into possible service lookup and in composing resultant services.

5 Conclusion

We have introduced a novel approach for a multi-agent based system for automatic service discovery and utilization. The approach is flexible and scalable allowing for new services and agents to be inserted into the system. The proposal to integrate a referral system to our approach gives the system greater power in the discovery of services.

References

[1] Universal Description, Discovery and Integration, Executive White Paper, 14 Nov 2001. http://www.uddi.org/pubs/UDDI_Executive_White_Paper.pdf

[2] Michael N. Huhns and Munindar P. Singh. Agents and multiagent systems: Themes, approaches, and challenges. In [9], chapter 1, pages 1–23. Morgan Kaufmann, 1998.

[3] Thomas E. Potok, Mark Elmore, Joel Reed, and Frederick T. Sheldon, "VIPAR: Advanced Information Agents discovering knowledge in an open and changing environment", Proceedings of the IIIS Agent Based Computing, Orlando, July 27-30, 2003.

[4] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Satish Thatte, Dave Winer and Henrik Nielsen. “Simple Object Access Protocol (SOAP)” 08-May-2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[5] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana “Web Services Description Language (WSDL)”, 15-Mar-2001, http://www.w3.org/TR/wsdl

[6] Tim Bray, Jen Paoli, C. M. Sperberg-McQueen, Eve Malor “Extensible Markup Language (XML) 1.0, 6-Oct-2000, http://www.w3.org/TR/REC-xml

[7] Huhns M. N, “Agents as Web services”, Dept. of Computer Science, South Carolina Univ., Columbia, SC. Internet Computing, IEEE On page(s): 93- 95 Volume: 6, Issue: 4, Jul/Aug 2002

[8] Sheng-Tzong Cheng; Jian-Pei Liu; Jian-Lun Kao; Chia-Mei Chen, A new framework for mobile Web services Applications and the Internet (SAINT) Workshops, 2002. Proceedings. 2002 Symposium on, 28 Jan.-1 Feb. 2002

[9] Mcllraith S. A, Son T. C., Honglei Zeng, “Semantic Web services” Knowledge Systems Lab., Stanford Univ., CA; Intelligent Systems, IEEE On page(s): 46-53 Volume: 16, Issue: 2, Mar/Apr 2001

[10] Tak W. Yan and Hector Garcia-Molina, “Index structures for information filtering under the vector space model”, Technical report, Department of Computer Science, Stanford University, Stanford CA

[11] Todd A. Letsche and Michael W. Berry, “Large-scale information retrieval with latent semantic indexing”, Information Sciences, 100:105-137, 1997

[12] Bin Yu and Munindar P. Singh, "Emergence of Agent-based Referral Networks." in Proceedings of AAMAS-2002