Towards Agent Based Web Service

Manish Chhabra, Hongen Lu
Department of Computer Science and Computer Engineering
La Trobe University
Bundoora, Melbourne
VIC 3086, AUSTRALIA
m2chhabra@students.latrobe.edu.au, helu@cs.latrobe.edu.au

Abstract

Web Service (WS) technology represents a fundamental shift in the way web applications are developed for e-business. It is becoming the next paradigm to deploy business services on the Web. Developing WS is a new challenge due to the complexity of various business processes and different communication protocols. In this paper, we propose a methodology to efficiently develop Web Services by wrapping software agents. This agent oriented method uses software agents as building blocks of WS, and exploits the commonly available agent development tools to accelerate the whole development cycle. Software agents in this methodology not only implement the business processes, but also enrich the functions of WS. Our approach provides a solution for Web Service engineering. A case study in holiday booking service is presented to show the benefits and advantages of this agent oriented service engineering methodology.

1. Introduction

The World Wide Web is evolving from a sea of information to a service oriented marketplaces and Web Service (WS) technology is the next wave of Internet computing. WS is one of the fastest growing areas of information technology in recent years. Web Services expose business processes over the Internet and promise more business opportunities by providing a common protocol that can be used by web applications to communicate with each other over the web. Web services are described in XML and are communicated over existing HTTP infrastructure using SOAP (Simple Object Access Protocol). The foundation of Web Services is laid by three significant standards: SOAP describes message format; WSDL (Web Service Definition Language) gives self-describing interfaces of Web Services; and UDDI (Universal Description, Discovery, and Integration) provides means to locate appropriate web services. Publicizing Web Services is also done using UDDI. More widespread definition is described in [4] as:

A WS is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet Protocols.

While WS is bringing revolution in the Web Wide Web by making it a more dynamic environment however still there is an increasing need to apprehend its current capability before exploiting its full potential [3]. With the help of software agents (SA) wrapped with WS, the information provided by WS can be made more efficient and more dynamic in nature. SA can not only filter out the unnecessary data collected from different sources and come up with the precise piece of information, but they can also help in discovery of other useful WS which is currently a big challenge with the increasing number of WS. Software agents are significant in many domains of computer science. Although there is no precise definition of SA, the present belief describes them as computer system situation in some environment capable of flexible autonomous actions in order to meet their design objectives [1].

Another benefit of wrapping agents into web services is that developers can easily and efficiently reuse widely accepted existing agent based systems as WS. We emphasize on using existing agent development tools which would speed up the development process of such a design.

2. Software agent and Web service

WS technology represents a fundamental shift in the way web applications are developed for e-business. Sure, there will still be web pages and HTML, and JavaScript authoring, but many e-business applications will more and more rely on programmatic interfaces that tie Internet-based applications together on fundamental level. Figure 1 shows the basic relationship between Requesting Entity and Provider Entity of a Web Service.
client services. It is an XML document that defines messages the WS understands and the format of its responses to those messages, protocol that the service supports, where to send messages.

Software agent provides a prominent solution to improve the Web Service standards with the benefits they provide to the user of Web Services. Agents take input from the environment based on the changes in the environment and perform actions based on the input. BDI (Belief-Desire-Intention) model as described in [5] is the most popular model for designing agent based systems. Beliefs represent an agent’s local knowledge base that is the agent’s information about the world which may also be incomplete or incorrect; Desires represent what an agent is trying to achieve, that is states of affairs that an agent would be brought about and they should be consistent with one another; Intentions are the currently adopted plans. Plans are predetermined sequences of actions that can accomplish specified tasks. These are desires which an agent has committed to achieve. In general, an agent can’t achieve all its desires. Figure 3 illustrate a clear picture of the model.

At the simplest level agents execute straightforward tasks based on specified rules and assumptions, like letting the manager know about the unusually decreasing stock quantity. Agents can also be service executing well defined tasks and can be predictive as well, like an agent which monitors the Newsgroups and returns the relevant information that would be of interest to a user. Software agents are also capable of notifying the agent owner with system-generated messages. As [6] explains, each agent may be equipped with an on-error module, which automatically executes when any fatal error occurs, and informing agent owner in an individual manner about the error and its consequences. In addition, a recovery action may be programmed, which improves the level of an agent’s autonomy. During the recovery, an agent may be self-reprogrammed to deal with the error during future executions.

Java-based agents are very popular, as they are easy to develop, and many tools and techniques exist extending the basic standard. Moreover, Java agents need no special runtime environment to be remotely executed, because
Java Virtual Machines exist on most of the network hosts. There are many agent frameworks, languages and protocols available today. JACK is an agent development tool provides an environment for building, running and integrating commercial-grade multi-agent systems using a component-based approach [7]. The JACK agent language is a programming language that extends Java with agent-oriented concepts. JACK source code is first compiled into regular Java code before being executed. Agents in JACK model reasoning behavior according to the theoretical BDI model.

With the demand of more powerful, efficient and versatile agents, effective communication among agents is required to work cooperatively. Knowledge Query and Manipulation Language (KQML) is a language used for agent communication and is most useful for communication among agent-based programs, in the sense that programs are autonomous and asynchronous. Autonomy entails that agents may have different or even conflicting agendas [8]; thus the meaning of KQML message is defined in terms of the message sender rather than message receiver. This allows message receiver to choose a course of action that is compatible with other aspects of its function.

3. Current challenges in Web services

One of the challenging problems that Web service technology faces is the ability to effectively discover services. UDDI registries are for the service provider to register their services under the predefined industry taxonomy and the registries provide the facilities to search the registered services. The search is mostly keyword search in the name or text description of the services. The underlying assumption of this approach is that the service providers will mark up their service profiles using English descriptions so that they could be easily understood by application developers who try to integrate the service into their applications, and that the service provider will register his service properly in the UDDI registry and provide enough text description so that one can search and locate their services. This however makes automated discovery much harder as English text descriptions are not machine interpretable. This has lead to a slew of research efforts aimed at "extracting" higher level descriptions and service classifications from WSDL descriptions.

Current survey of UDDI registries [15] illustrates further problems such as the URLs do not point to the WSDL files but rather to the introductory html pages of the services, in such automated program or a Software Agent can be used to find the WSDL file URL in the pointed page. Agents can also help to filter the invalid registry information which is very common in all the registries. Some of the registered services might not have a valid WSDL file entry that is the WSDL file is not a well-formed XML Document, or the WSDL file does not conform to the WSDL standards. There are also a lot of duplicates among the service entries. Communication between WSs should be more natural and easy to incorporate domain knowledge.

4. What agents can help in WS?

Software Agents can drive Web Services – both to implement them and access them as useful resources over the web. Integrating Web services and software agents brings about the immediate benefits of connecting application domains hosting one or the other technology. Therefore, it is not only the web services that provide agent functionality as a service over the web but agents can help to face the current challenges in Web Services. They can ease the number of problems currently we face with automatic service discovery.

Recently, we have seen an explosion of interest in ontologies as artifacts to represent human knowledge and as a critical component in several applications; among these the Web Services [17]. Ontologies are proposed as means to address semantic heterogeneity among Web accessible information sources and services. The Semantic Web services paradigm promises to take Web technologies a step further on providing rich and machine-understandable representation of services properties, capabilities, and behavior as well as reasoning mechanisms to support automation activities.

Moreover the “marriage” between agents and ontologies seems to be the kind of technology that can significantly change the face of enterprise software. On the one hand, ontologies should accomplish the task of giving a precious support to solve two tricky problems: how to efficiently discover Web services and how to make possible the interoperability of heterogeneous Web services [18]. In order to facilitate the resolution of such a structural and semantic heterogeneity, Web services, which play the role of workflow components, will have their interfaces semantically described by ontological concepts. On the other hand, ontologies enable agents to communicate in a semantic way, exchanging messages which convey information according to explicit domain ontologies.

In this scenario agents represent the “glue” that could hold these pieces together and make them perform properly. There is evidence from several research studies [16] that agents represent one of the most suitable technologies which can be used to meet the needs of Semantic Web Services. The goal of such efforts has been to make Web-Services agent-oriented rather than human-oriented, so that invocation of services can be automated.
Software agents could then discover information, execute processes and make selection of actions to take.

5. WAiWS methodology

To meet the increasing demand of various WSs, an engineering method is much needed to systematically develop reliable WSs. In this paper, we present an agent oriented methodology for Web Service Engineering, Wrapping Agents into Web Services (WAiWS). In WAiWS method, software agents are not only used for creating, storing, and providing information as they are done by a WS itself, but agents can also be used for continuous monitoring of changes in the information provided by WS and can put together the information gathered from different sources in a more resourceful manner.

Figure 4 illustrates Agent A wrapped partially with WS A, which is realized with the help of an adapter. Agent A provides its inherent features to WS and is able to communicate with other in-house agents like Agent B in Figure 4 using agent communication language like KQML. This figure also shows that the communication of an agent is not limited to other software agents; instead agents can gather information from external Web Services by SOAP messaging.

![Software Agent Wrapped with WS](image)

**Fig. 4. Software Agent Wrapped with WS**

As suggested [6] that with the current trend in WS, the level of service personalization must be restricted, because of the trade-off between service complexities and abundance of different users’ requirements. The unsatisfied users must perform additional data processing at user side. This design also fulfills the requirement of customization and service users are able to fit the service into their individual requirements and expectations for the reason that the processing can now be done within the service provider’s vicinity. Hence, the wrapping results in more competent Web Services. The following five steps are given to implement our proposed wrapping methodology:

1) Define the adapter interface. This adapter is used to communicate between agent and WS. The method bodies of this adapter interface will be implemented in the agent. They generate the events which in turn invoke the agent’s plan associated with these events.

2) In this step, we define the events that agent will handle. These events are generated by adapter methods for the agent to process. Agent oriented WS is active, it can handle and create events for others to process. This feature of agent makes the whole WS more dynamic and adaptive for any changes in the environment.

3) Next is the business process which brings into play when the events are generated. This is the plan of the agent which is associated with the events posted in the environment. Any business process is associated with a plan for agents to carry out. The correspondence of business process and agent plan makes the service flexible and adaptable for any future changes. Without rewriting other parts of the service, developers can simply modify plans to reflect the changes needed.

4) Implement the agent and within the agent give the method bodies defined in adapter interface in step 1. In the body part of an agent plans to execute business processes are given. In BDI model, agent plans are closely related to business processes. Even more, plans are reusable. Developers can composite or extended current existing plans. This will significantly shorten the development period. Reusing already existed and tested business plans also increase the reliability of the whole services.

5) Finally we create the WS and the agents are invoked either when the request is generated to WS or in an asynchronous manner. The calls to agents are made through the adapter.

6. Case study

In this section, we present a case study on a holiday web service which offers a complete holiday package through as one service using the help of agents, practically processing and evaluating the best deals for the user based on the input such as location and the amount they wish to spend. By deploying the holiday booking service online, it is a win-win situation for both the user and the suppliers. Users get the best deals available which includes a complete holiday package that is the air tickets, hotel availability and the local car hire for the place they wish to visit. The suppliers such as hotels and car rental industry have the advantage of reaching out to more people in a superior way.

6.1 Online holiday booking service

Holiday Booking Service (HBS) is an online service for the users to book their next holiday. This service defines a method searchHolidayDeals ( ) which takes the
relevant information of the user looking for good holiday deal. This includes the location and the maximum amount they wish to spend. The method generates an event of new holiday booking hence invoking the agent Holiday Agent. Information required by this service is gathered by the agent and presents the best deal within the given amount for the location desired. Although the Holiday Agent is wrapped with HBS, but it also communicates with other in-house agents, hence the service is managed by four agents working co-operatively and using the help of external WS.

Fig. 5. Online Holiday Booking Service

6.2. Software agents

To achieve this functionality Holiday Agent (aHa) is wrapped with the service using the methodology described in previous section. The agent with its accustomed capabilities is able to communicate with other in-house agents that provide with the flight information, hotel availability and local car hire for the place the user intends to visit. Since this information is not static, agents will need to access external services as shown in Fig. 5.

The agents form a system to process the holiday enquiry. Agent Flight Information (aFI) checks the different flight services and finds out the availability and price involved in traveling. Agent Hotel Availability (aHa) checks for the hotels within the budget of the user to find out the best deals available for the user. The car hire agent looks for the deals available for user in the area specified for holiday. Hence the overall system provides a complete holiday package as one web service using the functionality of different agents working together. Using the proposed methodology the whole system can be made online as a web service with the help of these software agents. Following is the code segment of HolidayAgent implemented in JACK [7].

```java
agent HolidayAgent extends Agent
    implements HolidayAgentAdapter
    {
        #handles event HolidayBooking;
        #uses plan ProcessHoliday;
        #posts event FlightInformation fi;
        #posts event HotelAvailability ha;
        #posts event CarHire ch;

        HolidayAgent (String name)
        {
            super(name);
        }

        void holidaySearch(/data parameters/) {
            postEvent(fi(/data parameters/));
            postEvent(ha(/data parameters/));
            postEvent(ch(/data parameters/));
        }
    }
```

6.3. Adapter

As discussed in WAiWS methodology, the wrapping between the WS and Agent is done with the help of adapter. The adapter HolidayAgentAdapter defines the method holidaySearch() which takes the customer details as data parameters. The adapter is implemented by aHa and generates the events for the agents to start processing the information. The set of actions that need to be performed using this data are defined as the plan of agent. Holiday Booking Web Service, which deploys the whole system online, will define the method to receive all the customer details and pass them to the agent using the adapter. The adapter is a key component to connect the WS and agent. It wraps an agent into a WS so that the business processes are exposed on the Web.

6.4. Agent plan

Finally, the plan where the business process resides is executed whenever the event is invoked, the plan is to process the user enquiry about the holiday and present with the best deals available based on the data members (information about the client). The plans of other SA in the picture also start their processing and use the help of external WS to get the deals available and then make the judgment about the best deals. The external WS in the picture include the airline services, the hotel services and the car hire services.
7. Conclusion

Currently Web Services are developed in an ad hoc manner. To meet the increasing demand of Web Services, an engineering approach is much needed. In this paper we propose an agent oriented methodology for Web Service engineering. In this approach software agents are the building blocks for Web services. Steps are given and described on how to build Web Services. Wrapping software agents with Web Services can churn out more efficient way of providing information through Web Services, which are capable to respond to frequent changes in the environment. We use autonomous and goal directed behavior of software agents to provide such information. The paper also presents the idea to use agents to provide information which is retrieved from combination of several independent Web Services. The case study clearly presents more understanding of their imperative behavior. We also demonstrate how this approach can be implemented using existing tools and technology. The wrapping methodology is easy to follow, and effective to build web based systems systematically.

Web services with the help of software agents can negotiate on our behalf for goods we intend to purchase with the use of agents who have set of rules specified on the basis of which they carry out their tasks; they can discover the precise information the user is searching for with the use of agents to perform a well specified task;

References