IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITB12875

This paper appears in the book, *Emerging Trends and Challenges in Information Technology Management, Volume 1 and Volume 2* edited by Mehdi Khosrow-Pour © 2006, Idea Group Inc.

Hybrid Agent Web Service Engineering: A Case Study in Financial Application Domain

Sujan Pradhan, Department of Computer Science and Computer Engineering, La Trobe University, Bundoora, Melbourne, VIC 3086, Australia, sujan@cs.latrobe.edu.au

Hongen Lu, Department of Computer Science and Computer Engineering, La Trobe University, Bundoora, Melbourne, VIC 3086, Australia, helu@cs.latrobe.edu.au

ABSTRACT

The adaptation of Web Service (WS) by the financial services industry is inherently becoming quite common in the new architectural paradigm-Service Oriented Architecture (SOA). However, such proliferation has had its share of limitations in building new loosely coupled, intelligent and dynamic applications. This paper proposes to extend these limitations by introducing a traditional concept of software agents which are wrapped around by WS. These agents capture the following essences which the WS lacks- autonomous behavior, ability to build up intelligence over time, and able to hold conversation-like transactions. These multilingual agents are hybrids, which profoundly fill in the few gaps of several WS shortcomings. In order to build such hybridity the following are proposed: during WS invocations SOAP is used as the Agent Communication Language (ACL) to communicate between agents (or between an agent and a WS) since it is readily available over HTTP; and during local (or distributed) access of resources some other traditional established form of ACL is used. By incorporating such hybrid characteristics new web applications are now able to utilize and capture the versatility of WS while maintaining dynamism, intelligence, transactional semantics and autonomous behavior within the application. Furthermore, the integration of these agents help construct a well defined architecture, as they are built from proven formal engineering methodologies and standards, which are not yet available to WS. A model in financial services application domain is used to exemplify the above.

1. INTRODUCTION

As WS is becoming more mainstream in the financial services industry there is an increasing need to apprehend its current capability before exploiting its full potential. A WS is described as a software system designed to support interoperable machine-to-machine interaction over a network [1]. This is among many definitions that have been used to describe a WS in the past.

The promise and advent of WS is that communication between human and machine will eventually be replaced by machine to machine communication. For example, a user who searches for the best widget will now be replaced by an agent, which in turn will search on the user's behalf. In this sense, the use of an intelligent agent to discover, publish and bind services become more prominent than just using plain Remote Procedure Calls (RPC); agents simply have the following added advantages among others- able to act autonomously and the ability to negotiate contracts through synchronous communication. They also have the ability to build up knowledge-base and intelligence over time. In order for these agents to communicate effectively, one common language must be shared. However, the sharing of a language is limited to common agents; this means it is possible for an agent to communicate in different languages with other agents which lie in separate domains. These are said to be hybrid agents. [13] uses the following definition: Hybrid agents refer to those whose constitution is a combination of two or more agent philosophies within a singular agent. The key hypothesis for having hybrid agents or architectures is the belief that, for some application, the benefits accrued from having the combination of philosophies within a singular agent is greater than the gains obtained from the same agent based entirely on a singular philosophy.

In the context of the current WS environment SOAP is the obvious choice of ACL since it is readily available and currently used as the de-facto language for communication between WSs. In combining agents and WS, [2] proposes these two fundamentally separate paradigms are closely related. Similarly, during local or distributed transactions where proprietary systems are in place an ACL such as KIF will be used. Using the above definitions we propose a new application model, STM (Stock Trade Model), for stock trading domain. This model represents the use of hybrid agent which communicates with separate WSs while maintaining close contact with other in-house agents.

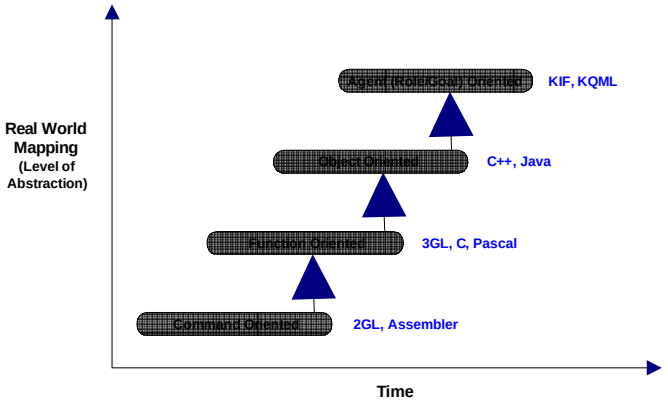
2. AGENT LANGUAGES USED FOR COMMUNICATION

Agent languages are an integral part of any agent-oriented framework. [14] states that agents are role/goal oriented. Figure 1 depicts this phenomenon in comparison to other languages. As such, agent languages differ depending on the agent's role or a task. These agents must be able to communicate in a language that all agents can understand or agree; if varying heterogeneous agents exist a multilingual or hybrid agent is needed.

2.1. SOAP as ACL

SOAP along with WSDL (Web Service Description Language) and UDDI (Universal Description, Discovery, and Integration) is the latest in RPC

Figure 1. Design/language paradigm [14]



technologies. It is known to seamlessly integrate distributed systems, applications, processes or functions over the Internet; its versatility to penetrate through firewalls is unmatched. This three-part XML protocol contains rules on descriptions of data types, and how and where to make RPCs and responses. The significance of SOAP in the WS-agent paradigm is unparalleled. WSs wrapped around agents can make use of SOAP as an ACL during machine to machine communications. The only restriction for these machines is that they must be able to communicate over the Internet.

Protocols of traditional ACLs are normally blocked by firewalls and proxy servers during RPC. Although these protocols provide a basic architecture for knowledge sharing through a special class of agent called communication facilitators, which coordinate the interactions of other agents [4], they do not communicate via HTTP like SOAP. It has already been established that HTTP is a better way to communicate between applications as it is supported by all Internet browsers and servers [7] [11]. Furthermore, SOAP is language and platform independent.

2.2. KIF as ACL

Unlike SOAP KIF is a traditional ACL which is designed for use in the interchange of knowledge among disparate computer systems (created by different programmers, at different times, in different languages, and so forth); however it is not intended as a primary language for interaction with human users (though it can be used for this purpose) [15] [3]. Albeit other ACL (i.e. Knowledge Query and Manipulation Language) exists KIF was chosen for implementing STM because of its ability to flawlessly integrate legacy systems, see Figure 2.

3. HYBRID AGENTS IN WS

Agents typically have these characteristics- communicative, capable, autonomous, and adaptive; when any one of these characteristics is repeated the agent becomes hybrid; the term hybrid here is applied using the definition set forth by [13]. In order to reason the need for such hybrid agents an explanation of the use of agents in general is needed, as set forth below.

3.1. The Use of Agents

An agent is a piece of software that has the capacity to autonomously conduct its work [5]. The reason for use of agents in this application is to provide the necessary conversation-like synchronous processing during each of the phases. Agents are inherently communicative [9], whereas WSs are passive until invoked.

Autonomy is one of the main features of agents. The ability of each of the agents to exist on its own is inherently important for a loosely coupled WS architecture; the autonomous agents help accomplish this feat. In addition, agents can suggest to buy, sell or hold a stock or suggest accordingly if there is certain trend in the market (i.e. federal reserve raises the interest rate); these are the characteristics of an intelligent agent. Such agents can also learn about the trend of traders (i.e. in which sectors stocks are traded often); these statistical data may have some intrinsic value to the traders themselves.

The agents in our model have to be able to communicate through some mechanism- in the WS context the choice would be SOAP. [5] suggests that one key advantage of having the ability to communicate this way for social interaction is that agents can form a domain agent society. This means that if an agent cannot answer a question it will find an agent who can. Such resources can ultimately help the trader make quicker decisions, i.e. Position Indicator Agent (PIA) in Figure 3 can suggest a buy, sell or hold position depending on the historical and current prices.

3.2. The Need for Hybrid Agents

A simple approach to understanding the need for hybrid agents is that an agent with one plan or philosophy will accomplish less than if it were to constitute a combination of two or more philosophies within a singular agent [13]. This was proven in [16] when hybrid agents were created out of the strengths of both the collaborative (deliberative) and reactive paradigms. Such agents conform to a set of rules which lead to higher benefits as opposed to standard agents. [17] states hybrid agents not only combine aspects of both reactive and deliberate agents but also make use of the best features of both architectures. Although the definition in [13] gives more adherences to the concept of hybrid agents during the implementation of STM the definition used by [17] is used to further extend the model.

4. FINANCIAL SERVICES APPLICATION DOMAIN MODEL: STOCK TRADING

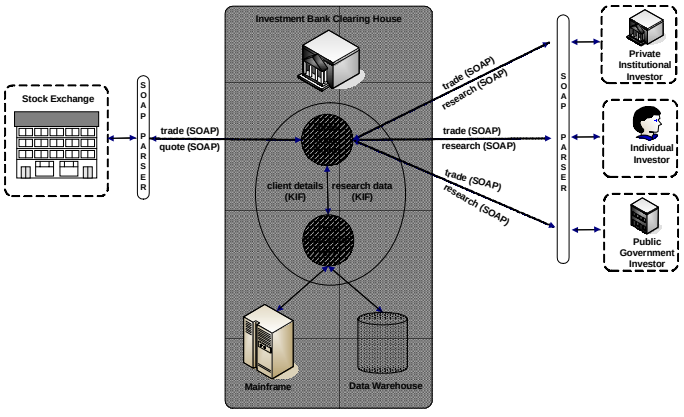
The STM is based on a scenario of trading stocks in an open financial market. STM stipulates a stock trade to any one given entity; in the real world the trading of stocks involves two essential fundamental steps: evaluation of historical data of the company to be traded and the execution of trade. STM emphasizes the importance of multi-party synchronous dialogue that must be carried out before a trade can be finalized. And such dialogue can only be carried out with the help of agents, hybrid agents to be more precise.

4.1. General Description

As in [12] the goal of this application is to realize and advance the potential of agent-based applications by constructing an open, distributed network of platforms hosting diverse agents and WSs. The motivating scenario behind this application is a potential stock trader who would like to conduct research by gathering historical data and to perform stock trades. Its goal is to make this process available to the masses and not confined to only an elite breed of investors who own expensive proprietary trading systems, such as Bloomberg Trading and NASDAQ Level II; in another words, this application domain not only tries to level the playing field for any one potential investor but also conforms to loosely coupled open trading systems. The intricacies of this application are heavily dependent on the synchronous request-response SOAP framework.

Figure 3 presents a high level depiction of the stock trade process. A potential trader either requests research information or wants to execute a trade. This request is first read by the Stock Trade Agent (STA); depending on the incoming information the STA performs a number of activities- fetches quotes from the stock exchange, executes trades on behalf of the client, and completes the following tasks with the help of Data Processor Agent (DPA): delivers raw historical data, performs OLAP (On Line Analytical Processing) queries with given parameters by utilizing existing data warehouse, and fetches client details from mainframes or other data repositories.

Figure 2. Financial services- stock trade model



4.2. Hybrid Agent: Stock Trade Agent

The STA in Figure 2 is an example of a hybrid agent. It has the ability to make request/response claims by interacting with WS via SOAP, as well as the ability to communicate with DPA via KIF; DPA may have either a local or distributed architecture. A scenario is now used to point out these intricacies. If a private institutional investor wants to trade equities it sends in a request to STA. STA will determine the nature of the invocation and if the request is to make a trade it will immediately send the request to one of its affiliated brokers in the Stock Exchange. STA could have also completed any one number of events before a trade was executed; for example, it could have supplied the investor with real time quotes, or client details or research data by utilizing DPA. The ability to complete these tasks would be daunting if STA was not hybrid; in other words, if STA lacked the ability to communicate in multiple languages.

1.1.1 STA utilizing SOAP message to complete a task

The code implementation below has been built using a Java tool, Java Agent Development Framework (JADE) [10] [8]. It shows STA in action- parsing SOAP message to XML, traversing through the XML document to locate the type of service requested and finally executing that service. In the above example, STA conducts a stock trade for an investor by sending the trade information to the Stock Exchange and relaying the confirmation of the trade back to the client investor.

```
package  com.investmentbank.stocktrade;

import jade.core.Agent;
import  jade.core.behaviours.*;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import  jade.domain.DFService;
import jade.domain.FIPAException;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;

import org.apache.xerces.parsers.DOMParser;
import  org.apache.xerces.dom.*;

import  org.w3c.dom.*;

import  java.util.*;


public class StockTradeAgent extends Agent
{
    protected void setup()
    {
        // Register the new stock trading service in the central registry
        DFAgentDescription dfAgentDesc = new DFAgentDescription();
         dfAgentDesc.setName( getAID() );
        ServiceDescription serviceDesc = new ServiceDescription();
         serviceDesc.setType( “stock-trading” );
         serviceDesc.setName( “JADE-stock-trading” );
         dfAgentDesc.addServices( serviceDesc );

        try
        {
            DFService.register( this, dfAgentDesc );
        }
         catch ( FIPAException fe )
         {
             fe.printStackTrace();
         }

       // Parse incoming SOAP message to XML (skipped for the sake of brevity)
        String soapMessage = SOAPMessageToXML.soapParser().toString();

         /* Traverse through the XML document using DOM
             (skipped some parts for the sake of brevity) */
        DOMParser parser = new DOMParser();
         parser.parse( soapMessage );

         String requestTypeNode =
              (((DocumentImpl)parser.getDocument()).getFirstChild()).toString();

         int accountNumber =
             Integer.parseInt((((DocumentImpl)parser.getDocument(
                                                 )).getSecondChild()).toString());

         String tickerSymbol =
             (((DocumentImpl)parser.getDocument()).getThirdChild()).toString();

          // If the investor wants to trade stocks
         if( requestTypeNode.equals( “tradestockrequest” )
         {
             String tradeType =
                (((DocumentImpl)parser.getDocument()).getFourthChild()).toString();

            String position =
                (((DocumentImpl)parser.getDocument()).getFifthChild()).toString();

             int shareAmount =
                 Integer.parseInt((((DocumentImpl)parser.getDocument(
                                                 )).getSixthChild()).toString());

              // Add the behavior for trade queries from investors
            addBehaviour( new TradeRequest(
                             accountNumber, tickerSymbol,
                                     tradeType, position, shareAmount ) );
         }
          // If the investor wants to conduct research
         else if( requestTypeNode.equals( “researchrequest” )
         {
             /* Needed variables would be traversed through
          the XML document (as above)  */

              // Add the behavior for research request from investors
             addBehaviour( new ResearchRequest( /* parameters here */ ) );
         }
    }

    // Agent clean-up functions
    protected void takeDown()
    {
        // Deregistering from the central registry
```

```
        try
        {
            DFService.deregister( this );
        }
        catch ( FIPAException fe )
        {
            fe.printStackTrace();
        }
    }

    /* Inner class which passes investor information to Stock Exchange and
       returns the result back to the investor */
    private class TradeRequest extends CyclicBehaviour
    {
        private int accountNumber;
        private String tickerSymbol;
        private String tradeType;
        private String position;
        private int shareAmount;

        private TradeRequest( int accountNumber, String tickerSymbol,
String tradeType, String position, int shareAmount )
        {
            this.accountNumber = accountNumber;
            this.tickerSymbol = tickerSymbol;
            this.tradeType = tradeType;
            this.position = position;
            this.shareAmount = shareAmount;
        }
        public void action()
        {
            MessageTemplate mt =
              MessageTemplate.MatchPerformative( ACLMessage.CFP );
            ACLMessage toStockExchange = msg.createReply();

            toStockExchange.setPerformative(ACLMessage.PROPOSE);

            /* The content would contain an XML document with these attributes:
                tickerSymbol, tradeType, position, shareamount */
            toStockExchange.setContent(
                String.valueOf( /* XML document here */ ));

            // Message sent to Stock Exchange(SE); SE sends back a SOAP
message
            String soapMessage = myAgent.send( toStockExchange );

            // The parsing of SOAP message is similar to above
            ...

            /* SOAP is parsed to XML doc; once traversed through the XML
               document the message would be sent back to the investor */

            ACLMessage toInvestor = msg.createReply();

            toInvestor.setPerformative(ACLMessage.PROPOSE);
            toInvestor.setContent(String.valueOf( /* XML document here */));
            myAgent.send(toInvestor);
        }
    }

    private class ResearchRequest extends CyclicBehaviour
    {
        /* The functions here would be similar to TradeRequest class;
           only exception would be this agent would communicate
           with the Data Processor Agent */
    }
}
```

4.3. Data Processor Agent

DPA has two responsibilities; it supplies client detail information and fetches research data. Once it is invoked by STA it can span through legacy systems, i.e. mainframe, to locate critical data. For example, if the investor wants to buy equity by utilizing leverage DPA would fetch current client account balance and any past due amounts. Another example is if STA requires any historical data (the clients or the investors may ask for this information if they are to conduct their own research) DPA would fetch this from the existing warehouse, see Figure 3.

4.4. Hybrid Agent: Position Indicator Agent

The PIA is another hybrid agent which utilizes a similar concept adopted by STA. The purpose of PIA is to inform the investors of market signals and trends. In order to arrive at such signals and trends it uses trading models to process the existing data from a data warehouse. In the process it may use OLAP (to query data warehouse) and fetch current quote for a company in question via SOAP (by connecting to a Stock Exchange). The result then will be passed to the caller, STA, via KIF. Thus, these multi-faceted tasks by one single agent translate to hybridism in action.

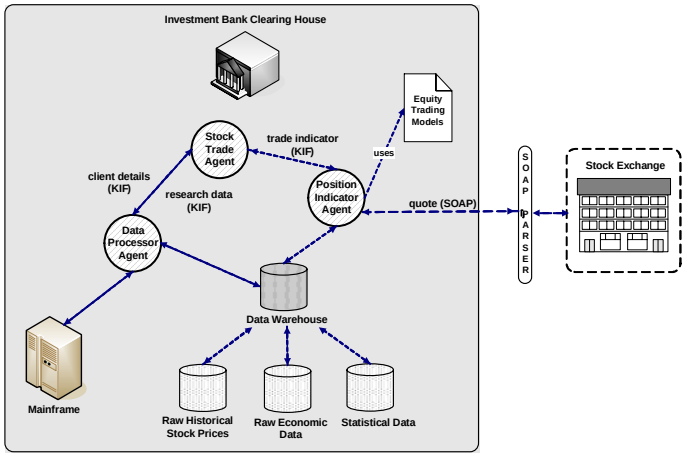
4.5. SOAP as ACL between Hybrid Agents and WS

Since WS in the financial applications framework is wrapped around hybrid agents SOAP has been chosen as the ACL for communication. Listing 1 to 4 below are SOAP messages during request/response between WS and STA.

Listing 1: WS SOAP request for a stock trade

```
<?xmlversion="1.0"?>
<SOAP-ENV:Envelope
```

Figure 3. Detailed view of hybrid agents



```
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:xsi="http://
www.w3.org/1999/XMLSchema-instance">
  <SOAP-ENV:Body>
    <m:tradeStockRequest xmlns:m="http://www.investmentbank.com/">
      <accountnumber xsi:type="xsd:int">98589025</accountnumber>
      <tickersymbol xsi:type="xsd:string">intc</tickersymbol>
      <tradetype xsi:type="xsd:string">market</tradetype>
      <position xsi:type="xsd:string">buy</position>
      <shareamount xsi:type="xsd:int">1000</shareamount>
    </m:tradeStockRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 2: STA SOAP response for a stock trade

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:xsi="http://
www.w3.org/1999/XMLSchema-instance">
  <SOAP-ENV:Body>
    <m:tradeStockResponse xmlns:m="http://www.investmentbank.com/">
      <tradeconfirm xsi:type="xsd:string">bought</tradeconfirm>
      <price xsi:type="xsd:int">48</price>
      <shareamount xsi:type="xsd:int">1000</shareamount>
      <totalamount xsi:type="xsd:int">48000</totalamount>
    </m:tradeStockResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 3: WS SOAP request for position indicator

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:xsi="http://
www.w3.org/1999/XMLSchema-instance">
  <SOAP-ENV:Body>
    <m:getPosition xmlns:m="http://www.investmentbank.com/">
      <tickersymbol xsi:type="xsd:string">intc</tickersymbol>
    </m:getPosition>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 4: STA SOAP response for position indicator

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:xsi="http://
www.w3.org/1999/XMLSchema-instance">
  <SOAP-ENV:Body>
    <m:getPositionResponse xmlns:m="http://www.investmentbank.com/">
      <currentposition xsi:type="xsd:string">sell</currentposition>
    </m:getPositionResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

5. CONCLUSION AND FUTURE WORK

In this paper we have presented a novel framework illustrating the integration of hybrid agents within a WS. The framework consists of two paradigms- one is agent-WS architecture [6] and the other is the use of hybrid agents to further extend the previous architecture. The notion of hybrid agents is relegated to a set of rules which state that the benefits accrued from having the combination of philosophies within a singular agent is greater than the gains obtained from the same agent based entirely on a singular philosophy [13].

WSs by themselves are passive; however, the integration of agents makes them not only dynamic but have these added benefits- improving the architecture by clearly redefining business processes, keeping local knowledge, the ability to ensue beliefs and perceptions from its current environment. Since agent is a pre-existing concept its vastly available tools (i.e. application of formal methods- engineering concepts) can be used to speed up the newly attained concept of WS since building WS is not an ad-hoc process. Therefore, combining these existing benefits extends the WS capability by making the application more loosely coupled, dynamic and intelligent. Another application domain model where multi-agents can mediate and compose WS in heterogeneous environment (possibly by using other ACL or mediation architectures) would be a consideration for future work.

6. REFERENCES

[1] BOOTH, D.; Haas, H.; McCabe, F.; Newcomer, E.; Champion, M.; Ferris, C.; and Orchard, D. 2004. Web Services Architecture. World Wide Web Consortium (W3C).

[2] DALTON, C. Model Checking Multi-Agent Web Services. Centre for Intelligent Sys and their Applications (CISA)

[3] http://logic.stanford.edu/kif/dpans.html

[4] FININ, T.; Fritzson, R.; McKay, D.; McEntire, R.; KQML- A Language and Protocol for Knowledge and Information Exchange University of Maryland, UMBC; Valley Forge Engineering Center, Unisys Corporation,

[5] FOU, J.; Web Services and Mobile Intelligent Agents: Combining Intelligence with Mobility; Web Services Architect October 10, 2001

[6] PRADHAN, S.; Lu, H.; Using SOAP Messaging to Coordinate Business Transactions in Multi-Agent Financial Web Services; Volume 1, books@ocg.at, iiWAS05, pg. 109-120, September 19-21, 2005

[7] http://www.w3schools.com/soap

[8] http://agents.felk.cvut.cz/teaching/ui2/JADEtutorial_v2.5.doc

[9] HUHNS, M.; Agents on the Web: Agents as Web Services; University of South Carolina; IEEE Internet Computing July-August 2002; IEEE Computer Society.

[10] http://jade.cselt.it/

[11] W3C Web Service Architecture Working Group; Web Service Architecture Recommendation.

[12] WILLMOTT, S.; Dale, J.; Burg, B.; Charlton, P.; O'Brien, P.; Agentcities: A Worldwide Open Agent Network. In: The Agentlink Newsletter 8, November 13-15, 2001

[13] NWANA, H. S.; Intelligent Systems Research; Knowledge Engineering Review, Vol. 11, No. 3, pp. 1-40, September 1996

[14] DIGNUM, F.; Web Services and Software Agents, Invited Presentation, iiWAS2003

[15] GENESERETH, M. R.; Knowledge Interchange Format, draft proposed American National Standard (dpANS) NCITS.T2/98-004

[16] MAES, P. (1991b), "Situated Agents Can Have Goals", In Maes, P. (ed) (1991a), Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, London: The MIT press, 49-70.

[17] NAMEE B.M.; Cunningham, P.; A proposal For an Agent Architecture for Proactive Persistent Non Player Characters; Department of Computer Science, Trinity College, Dublin 2, Ireland