Bridging Multi Agent Systems and Web Services: towards interoperability between Software Agents and Semantic Web Services

Omair Shafiq, Ying Ding, Dieter Fensel
Digital Enterprise Research Institute (DERI)
ICT Technology Park, Technikerstrasse 21a,
University of Innsbruck (UIBK),
6020 Innsbruck, Austria.
E-mail: {omair.shafiq,ying.ding,dieter.fensel}@deri.org

Abstract

The Semantic Web Services have been emerging to enable dynamic service discovery, composition, invocation, execution monitoring. On the other hand, Software Agents are envisioned as autonomous, proactive software entities that act on behalf its users according to a given agenda of goals. Software Agents have been envisioned as potential user of Semantic Web Services in order to interact with semantic descriptions of SWS to autonomously discover, select, compose, invoke and execute the services based on user requirements. However, there exists a communication gap among the both. The major reason is that Software Agents are not compatible with widely accepted standards of Web Services. This paper provides a solution to make Multi Agent Systems compatible with existing Web Services standards without changing their existing specifications and implementations, with an assumption that it will be helpful for enabling further interoperability between Software Agents and Semantic Web Services. AgentWeb Gateway is an initiative for dynamic and seamless interoperation of Multi Agent Systems and Web Services. We present abstract architecture and detailed design of the proposed system, abstract algorithms for required protocol transformations, evaluation of the system and analysis of results.

1. Introduction

The next generation of Web as Semantic Web Services envisions Web Services to be dynamically discovered, composed, invoked and executed. Software agent is a computer system which is capable of flexible autonomous action in a dynamic, unpredictable and open environment [9]. Agent technologies are a natural extension of current component based approaches [7], and have the potential to greatly impact the lives and work of all of us and, accordingly, this area is one of the most dynamic and exciting in computer science today. Hence, software agents being autonomous and proactive entities can be used to realize this vision by enabling interoperation between Software Agents and Semantic Web Services [30]. However, the current specification and implementation of Multi Agent Systems are not compatible with existing web standards. That is why, currently software agents and multi agent systems technologies cannot be used and communicate with Semantic Web Services. As a first step to bridge this communication gap, we introduce a middleware namely AgentWeb Gateway in between both the systems (Multi Agent Systems and Web Services) that enables software agents to dynamically and seamlessly discover, publish and invoke the web services in multi agent systems and vice versa. This will make the Software Agents and Multi Agent Systems compatible with existing Web Services standards. It will help is extending it further to enable the interoperability of Software Agents with Semantic Web Services.

A lot of work has been done by many other research communities to fulfill this communication gap, which enabled for the first time the major specification governing body of Software Agents and Multi Agent Systems, IEEE standards committee FIPA (Foundation of Intelligent Physical Agents) to show its interest for adapting Semantic Web and Web Services standards in its evolving specifications as a sub-group named Agents and Web Services Interoperability Working Group (AWSI) [28]. However, there are a number of limitations that exists in context of accuracy and functionality.

We have proposed AgentWeb Gateway [17] for interoperation of Software Agents and existing Web Services standards. It enables service discovery, service description transformation and service

invocation among software agents and web services without disturbing the existing specifications of both. Major challenges are involved in this integration is that both the technologies use different service registries, service description languages and communication protocols.

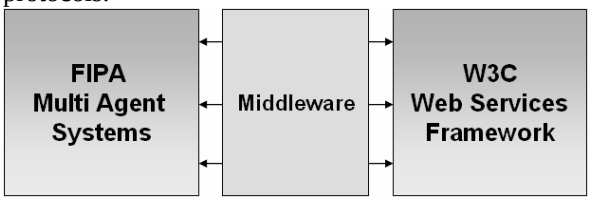

Figure 1: Middleware requirement for required integration

AgentWeb Gateway middleware provides solution for both the challenges by providing appropriate transformation mechanisms. The importance of this approach is that it enables integration of Software Agents and Web services without changing their existing specifications at the cost of time taken for translations which is negligible as compared to a transaction.

The paper further presents the detailed design of proposed system i.e. AgentWeb Gateway middleware, algorithms used for required transformation, evaluation of system with examples, observations on results, and finally draws conclusion and future work.

2. AgentWeb Gateway – Detailed Design

AgentWeb Gateway is going to provide the first step towards interoperation of Software Agents and Semantic Web Services. This first tep will enable Software Agents to discover, compose, invoke and monitor Web Services. Software Agents and Multi Agent Systems specifications are governed by FIPA (Foundation of Intelligent Physical Agents) and specifications of Web Services are governed by W3C, hence there is a lot of difference among specifications of both technologies and hence Software Agents and Web Service cannot communicate with each other.

We provide AgentWeb Gateway that acts as middleware between Multi Agent System and Web Services Framework and without changing existing specifications of both technologies.

It provides Service Discovery transformation, Service Description transformation and Communication Protocol transformation. Which means that using AgentWeb Gateway, without changing any specification of FIPA and W3C (agents and web services)

1. Software Agents can discover Web Services in Web Service registry (UDDI)
2. Software Agents can publish their services in Web Service registry (UDDI)
3. Software Agents can invoke Web Services
4. Web Service clients can discover Software Agents in Directory Facilitator (DF) of Agent Platform
5. Web Services can be published in Directory Facilitator (DF) of Agent Platform
6. Web Service clients can invoke Software Agents

This section describes the detailed design of proposed system in which the most important technical challenges are solved, i.e. without changing any

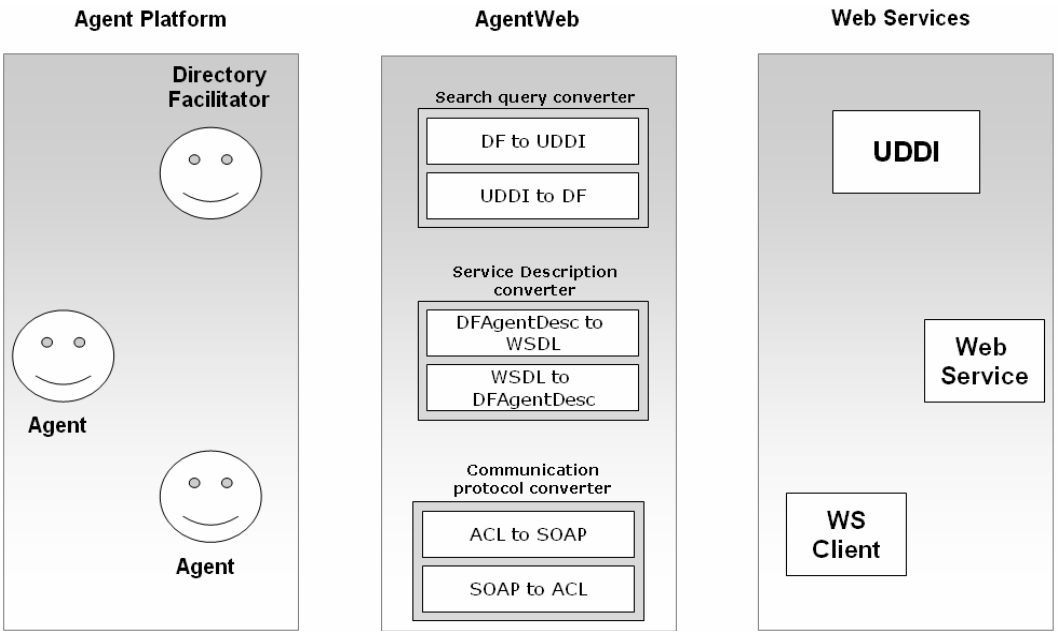

Figure 2: Detailed design of AgentWeb Gateway middleware

specification and implementation of Grid and Agents by enabling two-way Service Discovery, Service Publishing and Service invocation among Software Agents and Web-Services.

2.1. Service Discovery converter

This section presents the details of first component of AgentWeb Gateway which is called as Service Discovery Converter. This component enables service discovery among Software Agents and Web services i.e. Software Agents can do service discovery in Web Services registry as Universal Description Discovery and Integration (UDDI) and Web Service clients can do service discovery in Multi Agent Systems service registry as Directory Facilitator (DF).

2.1.1. DF to UDDI search query converter:

Whenever a Software Agent searches some service, it performs lookup for the Agent in Directory Facilitator (DF) of Multi Agent System by sending required DF-Agent-Description. If DF does not have the required agent registered, it redirects its search to the middleware by sending an ACL (Agent Communication Language) message. As soon as the middleware input interface receives message, it passes it to DF-Agent-Description analyzer. It extracts out three major portions of information i.e. information about Agent that provides the required service, required service description and inputs and outputs of the services.

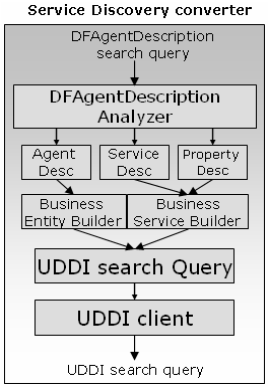

Figure 3: Software Agent searching for required service in UDDI

The information about Agent providing required services is far waded to Business Entity builder where it is mapped to Business Entity for UDDI search query. Information about required service descriptions and its properties are forwarded to Business service builder where it is mapped to name and description of required service and its inputs and outputs. The generated business entity and business service is forwarded to UDDI search query builder.

A search is performed in UDDI and if required service is found, a message is returned by the UDDI as SOAP based search query response to the middleware which is converted back to a valid ACL based search response message and sends back to DF. The DF further forwards the message to Software Agent that requested for search. In this way, the middleware has helped a Software Agent to search for the services in UDDI with an illusion that it is searching Agent services in DF of Agent Platform. Whole description can be visualized from figure 3.

2.1.2. UDDI to DF search query converter:

Whenever SOAP based Web service client needs some service, it performs lookup for the service in UDDI, if the UDDI doesn't have the required service, it redirects its search to the middleware by sending a simple SOAP based UDDI search request message. As soon as the middleware input interface receives message, it passes it to UDDI search query analyzer. It extracts out information about business entity and business service. Information about business entity is sent to Agent description builder there business entity is mapped over information about Agent providing required service. Information about business service is forwarded to service description builder and property builder where service name and type is used for building service description and inputs and output parameters are used for building property.

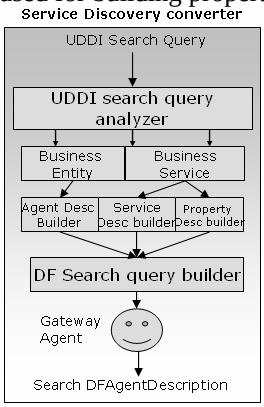

Figure 4: WS client searching for required service in Agent Platform

The generated Agent description, service description and property are forwarded to DF search query builder where DF-Agent-Description is generated and forwarded to DF of search. Directory Facilitator of the Agent Platform performs a search. If required service is found, a message is returned by the DF of that remote Agent Platform to the agent at our middleware which transforms ACL based DF search response back to SOAP based UDDI search response and sends to the UDDI lookup service which further forwards message

to the Web Service client requested for the search as shown in figure 4. In this way, the middleware helps the Web Service client to search for the services at Agent Platform. The SOAP response message contains the address of the middleware which means that Web Service client is given an illusion that the required service is available as Web Service at middleware.

2.2. Service Description converter

This section presents the details of second component of AgentWeb Gateway which is called as Service Description Converter. This component enables service publishing among Software Agents and Web services i.e. Software Agents can publish services in Web Services registry as Universal Description Discovery and Integration (UDDI) and Web Services can be published in Multi Agent Systems service registry as Directory Facilitator (DF).

2.2.1. WSDL to DF-Agent-Description converter

When a Software Agent comes to know about the existence and address of the required service and now the agent is required to consume the service. In order to consume the service, Software Agent is needed to know about the Ontology, AgentAction Schema, Predicate Schema and Concept Schema etc.

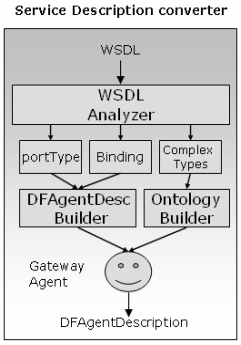

Figure 5: WSDL to DF-Agent-Description conversion: Agent understanding WSDL

On the other hand Middleware has the address for Services Description Language (WSDL) file. WSDL Analyzer, a component as obvious from name gets the WSDL file of required Web Service and analyzes portTypes, SOAP bindings for service and extracts out useful information from it. This extracted information is then passed further. For all complex types in the WSDL, Ontology (concept schema) is generated. Information about portType and binding is forwarded to DF-Agent-Description builder where the required the final 'Directory Facilitator Agent Description' is generated and given to Gateway Agent which further send an ACL based publish request to Directory Facilitator.

Transformation is performed from WSDL of Web service into a form (DF-Agent-Description) that Software Agents can understand. In this way we have made the Web Service description published in Agent Platform in order to make it understandable for Software Agents. Figure 5 explains whole scenario.

2.2.2. DF-Agent-Description to WSDL converter

This section explains that how a Software Agent publishes its services in Web Services registry UDDI. Information about services provided by an agent is stored as "Directory Facilitator Agent Description" (DF-Agent-Description) ontology in Directory Facilitator which is transformed by Service Description converter into WSDL. The whole transformation process is given below:

First of all DF-Agent-Description ontology is analyzed and description about Agent i.e. AgentID is taken out and is mapped to Service-End-Point of WSDL to be built. There are one or more Service-Descriptions available in this ontology which has information about the services of the agent. Name of each service is mapped to name of operation in portType of WSDL.

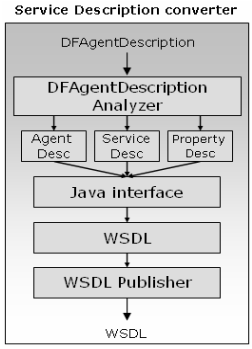

Figure 6: DF-Agent-Description to WSDL conversion: Agent publishing its services in UDDI

Each Service-Description of DF-Agent-Description has Property objects which indicate inputs and outputs of the corresponding services of the agent. Each Property object is checked. If it belongs to Predicate Schema (Predicate schema indicates propositions of an Agent) of Agent's ontology, it is treated as output of the corresponding operation of portType in WSDL. If the Property object belongs to Action schema of ontology (Action schema indicates the activities that can be carried out by an agent), it is then treated as input argument of the corresponding operation. After getting all the information about operation names, inputs and output, a java interface code is generated.

The java code is passed to Java to WSDL converter in order to translate the interface code into a WSDL to make it understandable for Web Service clients. WSDL file is generated and sent to Web Service client

and may be used for preparation of SOAP requests. Same WSDL file can be published in UDDI which will make the Agent, publish it services in Web Services world to make it understandable for Web Service clients. In this way a Software Agents gets its services published in UDDI by transformation of its DF-Agent-Description ontology into WSDL by Service Description converter of Agent Web Gateway.

2.3. Communication Protocol converter

This section presents the details of third component of AgentWeb Gateway which is called as Communication Protocol Converter. This component enables service invocation among Software Agents and Web services i.e. Software Agents can invoke Web Services and Web Service clients can invoke Software Agents in Multi Agent Systems.

2.3.1. ACL to SOAP converter

In previous sections, middleware has helped the Software Agent to search and understand services. Now the Software Agent is ready to consume the service. Software Agents gets the DF-Agent-Description ontology (which was generated in previous step) with an illusion that Gateway Agent is providing the required services. After getting ontology (DF-Agent-Description) from Agent, Software Agent sends an ACL request message having input parameters to the middleware.

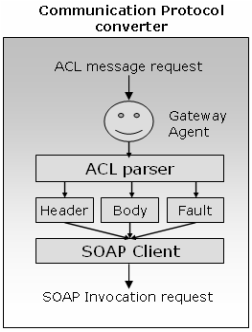

Figure 7: ACL to SOAP conversion: Software Agent invoking a service

Input interface receives the message and passes it to ACL2SOAP protocol converter. This converter extracts out the input parameters from ACL request message and creates an equivalent SOAP message. The SOAP client at middleware is directed to send the generated SOAP request message is sent to the Web Service at remote Web Server providing required services.

The Service after receiving SOAP request message processes the input parameters and then returns the output in the form of an SOAP response message to the SOAP client at middleware which upon receiving the SOAP response message passes it to SOAP2ACL protocol converter which extracts outputs from SOAP message and generates a ACL response message as shown in figure 7. The generated ACL message is then sent to the Software Agent. In this way, the middleware helps the Software Agent search, understand and consume Web Services.

2.3.2. SOAP to ACL converter

Up till now, the middleware has helped the Web Service client to search and understand the services provided by Agents. Now the Web Service client is ready to consume the services provided by the Agent. This time Web Service client communicates with the middleware with an illusion that it is the required Web Service.

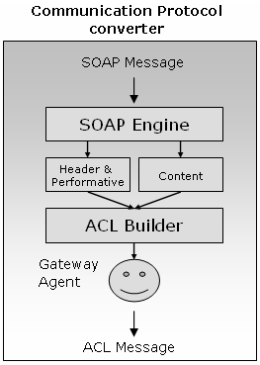

Figure 8: SOAP to ACL conversion: WS client consuming services provided by Agent

The client generates a SOAP request message (according to the service description which it got in WSDL) having input parameters. This SOAP request message is sent to the middleware. Input interface receives the message and passes it to SOAP2ACL protocol converter. This converter extracts out the input parameters from SOAP input message and creates an equivalent ACL message. The Agent at middleware is directed to send the generated ACL request message is sent to the Agent at remote platform providing required services.

The Agent after receiving ACL request message processes the input parameters and then returns the output in the form of an ACL response message to the Agent at middleware. The Agent at middleware upon receiving the ACL response message passes it to ACL2SOAP protocol converter which extracts outputs from ACL message and generates a SOAP response message. The SOAP response message is finally sent to the Web Service client as shown in figure 8. In this way, the middleware helps the Web Service client search, understand and consume services provided by Software Agents.

3. Testing and Evaluation

This section presents some useful scenarios for evaluation of the algorithms presented in previous section.

3.1. Evaluation of Service Discovery transformation

This section presents a scenario for evaluation of the algorithms for service discovery transformation. Here we show how a Web Service client performs service discovery in DF of Agent Platform. The request initiated by Web service client is UDDI search query which is as follows:

```
<businessEntity
 businessKey="677cfa1a-2717-4620-be39-6631bb74b6e1"
 operator="test " authorizedName=" Omair Shafiq: 86">
 <discoveryURLs>
   <discoveryURL useType="businessEntity">
http://uddi.rte.microsoft.com/discovery?businessKey=677cfa1a-2717-4620-be39-
6631bb74b6e1
   </discoveryURL>
 </discoveryURLs>
 <name xml:lang="en">CalculatorXmlWS</name>
<description xml:lang="en">Testing for AgentWeb Gateway by M. Omair Shafiq
</description>
 <businessServices>
   <businessService
   serviceKey="d8091de4-0a4a-4061-9979-5d19131aece5"
   businessKey="677cfa1a-2717-4620-be39-6631bb74b6e1">
   <name xml:lang="en">Math  Service</name>
   <description xml:lang="en">
    Math Service
   </description>
   <bindingTemplates>
    <bindingTemplate
       bindingKey="942595d7-0311-48b7-9c65-995748a3a8af"
       serviceKey="d8091de4-0a4a-4061-9979-5d19131aece5">
      <accessPoint URLType="http">
   http://202.83.166.177:8080/axis/Calculator.jws        </accessPoint>
       <tModelInstanceDetails>
        <tModelInstanceInfo
          tModelKey="uuid:42fab02f-300a-4315-aa4a-f97242ff6953">
         <instanceDetails>
          <overviewDoc>
            <overviewURL>
                http://202.83.166.177:8080/axis/Calculator.jws
           </overviewURL>
          </overviewDoc>
         </instanceDetails>
        </tModelInstanceInfo>
       </tModelInstanceDetails>
    </bindingTemplate>
   </bindingTemplates>
  </businessService>
 </businessServices>
</businessEntity>
```

For the above mentioned generated UDDI search query, following DF search query was produced by service discovery converter of AgentWeb Gateway.

```
( REQUEST
:sender ( agent-identifier :name Creator:77166138202@cern1-
7 )
:receiver  (set ( :agent-identifier DF:77166138202@cern1-7))
:content "( ( search-service ( :service-description : name Math
Service) ))"
:ontology Directory-Facilitator )
```

The ACL message generated above is DF search query which is sent to DF for service discovery.

3.2. Evaluation of Service Description transformation algorithms

This section presents a scenario for evaluation of the algorithms for service description transformation. We take a Web Services named ‘Calculator’ that contains one operation ‘add’ which requires two primitive integer types of arguments and has returns type of integer as well. Web Service Description Language (WSDL) (given below) of the web service is in plain text and is human readable.

```
<?xml version="1.0" encoding="UTF-8" ?>
   <wsdl:definitions targetNamespace="http://localhost:8080/axis/Calculator.jws"
<types>
    <xsd:schema
       targetNamespace="http://www.ecerami.com/schema"
       xmlns="http://www.w3.org/2001/XMLSchema">
       <xsd:complexType name="argument">
         <xsd:sequence>
           <xsd:element name="i1" type="xsd:int"/>
           <xsd:element name="i2" type="xsd:int"/>
         </xsd:sequence>
       </xsd:complexType>
    </xsd:schema>
  </types>
   <wsdl:message name="addResponse">
      <wsdl:part name="addReturn" type="xsd:int" />
   </wsdl:message>
   <wsdl:message name="addRequest">
      <wsdl:part name="i1" type="xsd:argument" />
   </wsdl:message>
   <wsdl:portType name="Calculator">
      <wsdl:operation name="add">
         <wsdl:input message="impl:addRequest" name="addRequest" />
         <wsdl:output message="impl:addResponse" name="addResponse" />
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="CalculatorSoapBinding" type="impl:Calculator">
      <wsdlsoap:binding                                                    style="rpc"
transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="add">
          <wsdlsoap:operation soapAction="" />
            <wsdl:input name="addRequest">
                <wsdlsoap:body …/>
            </wsdl:input>
            <wsdl:output name="addResponse">
                <wsdlsoap:body              …namespace="http…/axis/Calculator.jws"
use="encoded" />
            </wsdl:output>
          </wsdl:operation>
      </wsdl:binding>
   <wsdl:service name="CalculatorService">
      <wsdl:port binding="impl:CalculatorSoapBinding" name="Calculator">
         <wsdlsoap:address location="http://localhost:8080/axis/Calculator.jws" />
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

In case of interoperability among Agents and Web Services, WSDL of calculator web services is to be published in Directory Facilitator of Agent Platform and hence WSDL is transformed by service description transformation component of AgentWeb Gateway into Directory Facilitator Agent Description (DFAgentDescription). DFAgentDescription is serialized in binary format and is not human readable. Information about DFAgentDescription about object having values is shown below:

DFAgentDescription
- AgentID = ‘CalculatorAgent:reverse-ip@machine-name’
- Ontologies = ‘CAOntology’
- Protocols = ‘’
- Languages = ‘’
- Lease time= ‘default’
- Scope = ‘default’

Service-Description
- Name = ‘add’
- Type = ‘Math’
- Ontologies = ‘addOntology’
- Protocols = ‘’
- Languages = ‘’
- Ownership = ‘CalculatorAgent’

Property
- Name = ‘addRequest’
- Value = ‘AddRequestActionSchema’

- Name = ‘addReturn’
- Value = ‘AddResponsePredicateSchema’

addOntology has following information:

addOntology

Concept Schema
- Name = ‘argument’

AgentAction Schema
- Name = i1
- Schema = Concept (argument)

Predicate Schema
- Name = addReturn
- Schema = Primitive (Integer)

argument (concept schema)
- name = ‘i1’
- type = Primitive (Integer)

- name = ‘i2’
- type = Primitive (Integer)

In order to publish Web Services Description Language (WSDL) of Web Service in Directory Facilitator, it has been converted into Directory Facilitator Agent Description (DFAgentDescription) as given above according to algorithm in section 7.1.

Similarly, a transformation would be required from Directory Facilitator Agent Description (DFAgentDescription) to Web Services Description Language (WSDL) according to the algorithm given in section 7.2.

3.3. Evaluation of Communication Protocol transformation algorithms

This section completes the above mentioned scenario, i.e. after service description transformation, communication protocol transformation is required for service invocation. Consider a WS/SOAP client want to get services provided by an Agent that provides services of add, subtract etc. The WS/SOAP client would send request in according to its SOAP format as follows:

SOAP Request

```
POST /InStock HTTP/1.1
Host: http://202.83.166.177:8080/axis/Calculator.jws
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">
<soap:Body
xmlns:m="http://202.83.166.177:8080/axis/Calculator.jws">
   <m:add>
     <m:i1>2</m:i2>
     <m:i1>3</m:i2>
   </m:add>
  </soap:Body>
</soap:Envelope>
```

Using communication protocol converter of AgentWeb Gateway, the SOAP message will be transformed into ACL request according to the algorithm presented in section 8.1. The transformed ACL request is given below:

Transformed ACL Request

```
(request
   :sender (agent-identifier
     :name Gateway:78166138202@Cern1-7
     :addresses (sequence http://202.83.166.187:7776/acc))
   :receiver (set (agent-identifier
     :name MathAgent@78166138202@Cern1-7
     :addresses (sequence http://202.83.166.187:9999/acc)))
   :content
   "(action (addAgentAction
     :properties (set
        (property i1 2)
        (property i2 3)))")
```

The transformed ACL message will be forwarded to the actual agent (MathAgent) providing the required add service. The MathAgent would response accordingly in ACL which will be received by Gateway Agent. The ACL response is given below:

ACL Response

```
(request
   :sender (agent-identifier
     :name MathAgent:78166138202@Cern1-7
     :addresses (sequence http://202.83.166.187:7776/acc))
   :receiver (set (agent-identifier
     :name Gateway@78166138202@Cern1-7
     :addresses (sequence http://202.83.166.187:9999/acc)))
   :content
   "(action (addAgentAction
     :properties (set
        (property addResult 5)))")
```

Using communication protocol converter of AgentWeb Gateway, the ACL response message would be converted into SOAP response message according to the algorithm presented in section 8.2. The transformed SOAP response is given below:

SOAP Response

```
HTTP/1.1 200 OK
Content-Type: application/soap; charset=utf-8
Content-Length: nnn<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">
  <soap:Body
xmlns:m="http://202.83.166.177:8080/axis/Calculator.jws">
   <m:add>
     <m:addResult>5</m:addResult>
   </m:add>
  </soap:Body>
```

</soap:Envelope>

4. Results and Analysis

This section presents results of required transformations i.e. Service Description transformation and Communication Protocol transformation.

4.1. Service discovery transformation results

The service discovery transformation is carried out with 100% accuracy as it is based on keyword based search. However the delay imposed in the conversion depends on number of business services that are present in a business entity of a UDDI search query.

When a Web Services client searches for some Agent from Directory Facilitator of Multi Agent System, time required for the transformation of UDDI search query depends on the number of BusinessServices entries under the BusinessEntity of UDDI search query. Same is the case for vice versa, i.e. when an Agent wants to search for some web services in UDDI, time required for transformation of DF search query into UDDI search query depends on number of services specified under the description of an Agent.

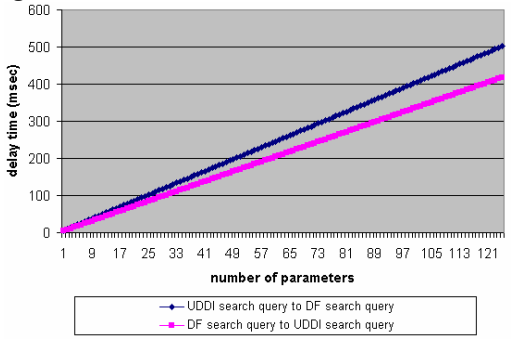

Figure 9: Performance analysis of service discovery transformations

Figure 9 below shows the analysis of delay that occurs while transforming the DF search query into UDDI search query and vice versa in order to enable Software Agents discover Web Services from UDDI and vice versa. The line of increment in delay of transformation shows linear behavior. For a given number of parameters to be searched in the query, DF search query to UDDI search query conversion takes lesser time than that of UDDI search query to DF search query due to the fact that Reason for this is UDDI search query requires more time to be parsed as it is basically and XML document, whereas DF search query is in the form of object and binary based.

4.2. Service description transformation results

Accuracy depends on the provided information of Web Service in WSDL and Software Agent in DFAgentDescription. If it is completely valid then 100% results can be obtained. In case of a Software Agent publishing services in UDDI (DFAgentDescription to WSDL conversion), time required for transformation for Service description depends upon the complexity of ontology and number of service-description in DFAgentDescription.

In case of Web Services publishing its services in Directory Facilitator (WSDL to DFAgentDescription conversion), time required for transformation for service description depends on Complex-Types and number of operations in portType of WSDL.

Figure 10 shows our analysis of the delay that occurs in the process of transformation on either side. In case of DFAgentDescription to WSDL conversion for an Agent to publish its services in UDDI, It was observed that as number of services of an agent increases the time taken for transformation also increases. Same is the case for WSDL to DFAgentDescription when a Web Service is to be published in Directory Facilitator, delay in transformation increases with increase in number of operations in portType of WSDL.

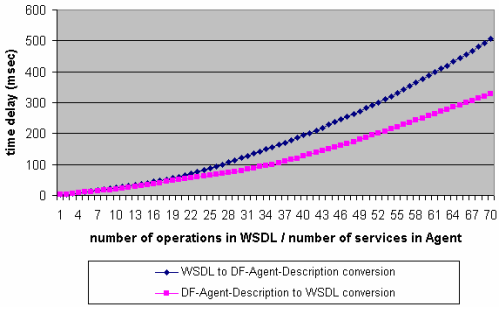

Figure 10: Performance analysis of service description transformations

Next observation is that line of increment in delay of transformation shows exponential behavior. Reason for exponential behavior of increment is as there is increment of one operation in portType of WSDL; it would have some input message, output message, elements and complex types (optional). Same is the case in with DFAgentDescription.

Graph of WSDL to DFAgentDescription transformation has rapid increase than in case of DFAgentDescription to WSDL transformation. Reason for this is WSDL file requires more time in parsing as is based on file with plain text than that of

DFAgentDescription which is in the form of object and binary based.

4.3. Communication protocol transformation results

Both in SOAP request/response message and ACL message, only one operation or AgentAction/Predicate respectively can be targeted in a single call. Accuracy of SOAP to ACL and ACL to SOAP transformation depends upon the accuracy of message. A valid message would be transformed into it's vice versa with 100% accuracy.

In case of a Software Agent invokes a Web Service, time required for communication protocol transformation (ACL to SOAP conversion) depends upon the complexity of schema of Property objects in Service-Description of DF-Agent-Description of an Agent. If there are primitive schemas only, then transformation process would take almost similar time as expected. In case of concept schema involved, additional time would be required for transformation of concept schema into complex type.

In case of a Web Service client invokes a Software Agent, time required for communication protocol transformation (SOAP to ACL conversion) depends upon the complexity of input and output parameters. If inputs and outputs are primitive data-types, transformation process would take almost similar time as expected. If there are complex data-types involved, additional time would be required for transformation of complex data-type into ontology (concept schema).

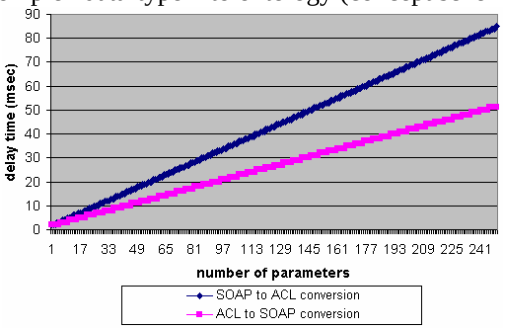

Figure 11: Performance analysis of communication protocol transformation

In figure 11, we have analyzed the delay occurs in the process of transformation on either side. In case of ACL to SOAP conversion for an Agent to invoke web service, it was observed that as number of parameters SOAP message increases, the time taken for transformation also increases. Same is the case for ACL to SOAP when a Web Service client is to invoke an Agent, delay in transformation increases with increase in number of Property elements of content of ACL message. Next observation is that line of increment in delay of transformation shows linear behavior. Exponential behavior can only be shown in case of complex data-types are used. For a given number of parameters, ACL to SOAP conversion takes little less time as compared to SOAP to ACL conversion. Reason for this is SOAP requires more time parsing as it is based on plain text than that of ACL message which is in the form of object and binary based.

5. Conclusions and Future Work

In order to fulfill the needs of emerging distributed applications with higher complexity, Software Agents and Multi Agent System are envisioned to be interoperable with emerging Semantic Web Services. But firstly, the Multi Agent Systems have to be compliant to the widely accepted standards of Web Services. This paper provides an initial step to do so by bridging the communication gap among Multi Agent System and Web Services without changing their existing specifications and implementations by providing AgentWeb Gateway that acts as a middleware between the both technologies. It is facilitated by carrying out two way dynamic and seamless transformations for service discovery, service description and communication protocols. This paper provides detailed design of the system along with detailed functional components of each of its components, algorithms for required transformations, evaluation of the system with examples and critical analysis of the results.

After enabling Multi Agent Systems compatible with existing Web Services standards, next step is to extend this further to Semantic Web Services, i.e. to enable Software Agents understand the semantic descriptions of SWS or semantically interact with SWS. It will bring the Semantic Web to next level of automation where goal-oriented Software Agents on behalf of their users proactively and dynamically discover, select, compose, mediate and invoke Web Services based on user's requirements.

Acknowledgements

Authors would like to thank for partial funding support from Austrian Government under FIT-IT GRISINO project, Higher Education Commission of Pakistan (HEC) Islamabad Pakistan and Communication Technologies (Comtec) Sendai Japan.

References

[1] S. Tuecke, ANL; K. Czajkowski, USC/ISI; I. Foster, ANL; J. Frey, IBM; S. Graham, IBM; C. Kesselman, USC/ISI; T. Maquire, IBM; T. Sandholm, ANL; D. Snelling, Fujitsu Labs; P. Vanderbilt, NASA, GWD-R "Open Grid Services Infrastructure (OGSI) Version 1.0".

[2] I. Foster, D. Snelling, "Web Service Resource Framework – WSRF", http://www.globus.org/wsrf/faq.asp, March 2004

[3] I. Foster, Nicholas R. Jennings, Carl Kesselman, "Brain Meets Brawn: Why Grid and Agents Need Each Other", Proc. Autonomous Agents and Multi Agent Systems (AAMAS) July 2004.

[4] The Web Services Agent Integration Project AgentCities.NET http://wsai.sourceforge.net

[5] H. Kuno and A. Sahai, "My Agent Wants to Talk to Your Service: Personalizing Web Services through Agents" 1st International Workshop on "Challenges in Open Agent Systems, Bologna, Italy,July 2002.

[6] Agentcities: Building a Global Next-Generation Service Environment – J. Dale, S. Willmott and B. Burg 10th June, 2002.

[7] M. Luck, P. McBurney, C. Preist, "Agent Technology: Enabling Next Generation Computing - A Roadmap for Agent Based Computing", January 2003, AgentLink II.

[8] I. Foster, and C. Kesselman (eds.). The Grid: Blueprint for a New Computing Infrastructure (2nd Edition). Morgan Kaufmann, 2004.

[9] M. Wooldridge, Agent-based software engineering. IEE Proc. Software Engineering, 144. 26-37. 1997.

[10] N. R. Jennings, K. Sycra, M. Wooldbridge, "A Roadmap of Agent Research and Development", Autonomous Agents and Multi-Agent Systems, pp.275-306, Kluwer Academic Publisher, Boston,(1998).

[11] M. S. Raisinghani, "Electronic Commerce at the Dawn of Third Millenium", Idea Group Publishing, (2000).

[12] Global Grid Forum http://www.gridforum.org/

[13] I. Foster, C. Kesselman, J. Nick and S. Tueske, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Globus,Project, www.globus.org/research/papers/ogsa.pdf (2002).

[14] K. Mori, "Autonomous Decentralized Systems: Concept, Data Field Architecture and Future Trends", Proc. of the first International Symposium on ADS (ISADS93), IEEE, Kawasaki, Japan, pp.28-34, 1993.

[15] R. Wolski, J. Brevik, J. Plank and T. Bryan, Grid Resource Allocation and Control Using Computational Economies. Berman, Wiley and Sons, 2003, 747-772.

[16] C. Walton and A. Barker, "An Agent-based e-Science Experiment Builder", 1st International Workshop on Semantic Intelligent Middleware for the Web and the Grid, Valencia, Spain, August 2004.

[17] M. Wooldridge and N. R. Jennings, Software Engineering with Agents: Pitfalls and Pratfalls. IEEE Internet Computing, 3 (3). 20-27. 1999.

[18] H. F. Ahmad, K. Iqbal, A. Ali, H. Suguri, "Autonomous Distributed Service System: Basic Concepts and Evaluation", Proc. The 2nd International Workshop on Grid and Cooperative Computing, GCC 2003, pp. 432-439, Shanghai, China.

[19] C. Goble, D. De Roure, N. R. Shadbolt and A. Fernandes. Enhancing Services and Applications with Knowledge and Semantics. The Grid: Blueprint for a New Computing Infrastructure (2nd Edition), Morgan-Kaufmann, 2004.

[20] D. Levine and M. Wirt, Interactivity with Scalability: Infrastructure for Multiplayer Games. The Grid: Blueprint for a New Computing Infrastructure, Kaufmann, 2004.

[21] H. F. Ahmad, K. Mori, "Autonomous Information Fading and Service-Guided Navigation Techniques for Mobile Agents", Proceeding of IEEE Computer Society, SMC99 conference pp. II-83-II-II-87, (1999).

[22] H. Tsunemitsu, H. F. Ahmad, Helene, A. , K. Mori, "Autonomous Decentralization Technology for Service Integration of Different Service Providers", 12th SICE Symposium on Decentralized Autonomous Systems, pp. 373-378 (2000).

[23] H. F. Ahmad, A. Ali, H. Suguri, Z. A. Khan, M. Rehman, "Decentralized Multi Agent System: Basic Thoughts", 11th Assurance System Symposium, Sendai, Japan 2004.

[24] A. Ghafoor, M. Rehman, Z. A. Khan, A. Ali, H. F. Ahmad, H. Suguri, "SAGE: Next Generation MAS" pp. 139-145, Vol 1. Navada, USA, 2004.

[25] S. Bashir, M. Rehman, H. F. Ahmad, A. Ali, H. Suguri. "Distributed and Scalable Message Transport Service for High Performance Multiagent Systems", INCC 2004 Pakistan. pp. 152-157

[26] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, "W3C specifications Web Services Description Language (WSDL) 1.1"

[27] Foundation for Intelligent Physical Agents, FIPA Agent Management Specifications 2002, SC00023J, Geneva, Switzerland.

[28] IEEE-FIPA Agents and Web Services Interoperability (AWSI) Working Group. Available at http://www.fipa.org/subgroups/AWSI-WG.html

[29] M. O. Shafiq, H. F. Ahmad, H. Suguri and A. Ali, "Autonomous Semantic Grid: Principles of Autonomous Decentralized Systems for Grid Computing", IEICE & IEEE Joint Journal, Special issue on Autonomous Decentralized Systems (ADS), Transactions on Information and Systems E88-D(12):2640-2650, December 2005.

[30] D. Roman, U. Keller, H. Lausen, J. Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel: Web Service Modeling Ontology, Applied Ontology, 1(1): 77 - 106, 2005.