An Agent-Based Web Services Market

Ali Durmus and Nadia Erdogan

Istanbul Technical University
Istanbul, Turkey
alidur@gmail.com,erdogan@itu.edu.tr

Abstract. Agent Based Web Services Market (AWSM) is a framework for agents to present and sell their capabilities as web services. Agents take certain roles and cooperate to constitute agent societies execute their functionality as web services. The roles they possess determine which web service they can provide. Roles in an agent society are a collection of services and can be taken and be left dynamically by agents at runtime. Service provider agents in AWSM register their services as web services. Client that needs web services can apply AWSM to find a web service according to services information, service level parameter and cost criteria. The system organizes a tender to determine the best matching web service and provider agent. The client then can call the web service that is provided by the awarded agent. Service level and performance of this call is registered by system to assess the awarded agent's performance and commitment and to later use this information in future tenders.

1 Introduction

Currently, there are several web services that carry out different tasks in the internet. They either work alone or cooperate with other web services to fulfill their tasks. They can be called by other applications; they can refer to other web services as well. Web services are sold by companies whose main aim is to profit by providing service in the market. They create and present services. They may as well buy other companies' services and use them to provide their own services. Bids may be carried out to determine the best suitable service providers. Companies in markets are very dynamic in real life. They may change their structure and business fields. An agent based approach to the implementation of a framework that meets the needs of a web services market fits well in this dynamic environment. Web services are provided by agents which carry the roles appropriate for those services.

We describe the design and implementation issues of a new agent-based web services market. Agents carry roles which are composed of services that are published as web services. A client looking for a service calls a particular web service of the system, providing it's criteria for the request. The system puts out to tender the request and determines the suitable agents that provide that service and are currently available on the market. The details of the implementation will be presented in following sections.

1.1 Service Oriented Architectures

Service Oriented Computing is a paradigm based on services for application development [1]. Service Oriented Architectures (SOA) have very widespread usages.

R. Lee & N. Ishii (Eds.): Soft. Eng. Research, Manage. & Appli. 2009, SCI 253, pp. 145–157.
springerlink.com

They are used in individual services like book reservations, as well as in complex services such as hotel and flight reservation. Services can carry out simple requests or complex business process functionalities. SOA aggregates small services to form large and complex applications. SOA focus on small service functionalities that do their task and have interfaces that show how to interact with other services. In heterogeneous network environments, a service is encapsulated and its standard interface enables it to interact with other programs or applications.

- **Technology independence:** Services can be called in a standard way by applications located in environments of different information technology.
- **Loose coupling:** Providers and consumer of services do not need to know the internal structure or context of each other.
- **Reusability:** Since services focus on individual functions, tasks which will be carried out are divided into very small services which can also be used in other implementations as well.

Service oriented programs are designed to carry out functions that can be called in different environments or between different environments. Although SOA solves interaction issues between different environments and divides larger problems into small ones, it is not as good at questions of coordination, orchestration, cooperation, and adaptation, which are superiority of agent systems. Thus, advantages of service oriented architectures and agent oriented architectures can complement each other effectively. This approach has been adopted in some recent work in literature [2, 3].

2 Motivations and Background

Companies that sell services need a web services market to publish, to receive requests for and to sell their services. Both representation and implementation of the dynamic structure of companies and of the market is a big challenge. For example, consider modeling a travel agency company. The company sells flight tickets and has hired an employee with the sales representative role to provide this service. So, here we see that a service is associated with a role. Next, assume that the company makes agreements with some hotels to make reservations for them. As the company does not have the resources to hire a new employee, the former employee is now in charge of this task as well, leading to a change in the sales representative role to include hotel reservations service as well. This case shows how role definitions may need to be modified at runtime. As the company's sales go up, a new employee with the customer care role is hired to increase customer satisfaction, so as to profit. Thus a new role, customer care role, is injected into the system. Now, assume that conditions lead to the dismissal of the new employee and the first employee is asked to take the customer care role too. This is an example of how an entity may acquire more than one role dynamically at runtime. The contrary may also be true, resulting in the loss of certain roles carried during execution. Several companies similar in structure to this travel agency company may create and take part in a market to sell their services and they may even enter and leave it from time to time, thus requiring a dynamic execution environment.

As the above example shows, there are several open issues to be solved. Assigning services roles, acquiring and leaving roles at runtime, and changing role definitions at runtime are some of them.

Web services and classical programming techniques do not provide solutions to these problems. A web service is a very static entity which cannot change its structure and it is not good at cooperation and coordination.

The need for dynamism, cooperation and coordination leads us to use web service and agents together. As stated in [4], agents can enhance the capabilities of a web service architecture in the following aspects:

- A web service knows only about itself while agents are often self-aware and gain awareness of other agents and their capabilities as interactions among the agents occur.
- Web services, unlike agents, are not designed to use and reconcile ontologies, while agents can make extensive use of ontologies.
- Agents are inherently communicative, whereas web services are passive until invoked.
- A web service, as currently defined and used, is not autonomous. Autonomy is a characteristic of agents, and it is also a characteristic of many envisioned Internet based applications.
- Agents are cooperative, and by forming teams and coalitions they can provide higher-level and more comprehensive services. Current standards for web services do not provide for composing functionalities.

There are examples on using web services together with agents in the literature. Rykowski, Wojciech [3] propose a Virtual Web Service (VWS) system which makes statically provided web services more dynamic. VWS users consume web services which are provided by one or more agents. The system includes two types of agents; private agents and public agents.

Wang, Wang, Deng, Zhao, Yung, Gao [5] use agents and web services together in Web-service-agents-based Securities Trading Simulation System (STSS). Agents are wrapped as Web Services communicating and interacting with each other in an open environment. The system makes use of advantages of web services in communication and makes use of advantages of agent in autonomy and intelligence.

Richards, Splunter, Brazier, Sabou [6] propose to use agents in automatic web services composition. Web services are defined semantically by using DAML-S and Agent Factory uses these descriptions in its design process to derive a Web service configuration.

Vall, Ramparany, Vercouter[7] use agent and web service together in pervasive computing environment with smart device to compose web service dynamically. They use semantic service description to abstractly describe services' functionality. Agents use these descriptions to form services.

In the works cited above, the use of agents makes individual web services more dynamic and intelligent. In our work, we need not only dynamic, autonomous and intelligent web services, but also dynamic companies (agent societies) to provide those services. Entities (agent societies) in system should be autonomous and dynamic in

structure. Agents (people) can apply agent society for some positions (roles). The positions (set of roles) provide the agent with the ability to perform certain services. An agent society can accept or reject an agent's application. An agent society owns a set of roles that are composed of services and these may be modified at runtime.

We propose to use role-based agents to meet the above stated requirements of a web services market and companies taking part in that market. The concept of a "role" associated with an agent enhances the dynamic behavior of the agent, and also extends the dynamic nature of agent societies which represent companies/organizations. Dynamic changes in the internal structure of companies can now be easily modeled through assignment and manipulation of the roles carried by agents. Agent societies can change their role definitions dynamically at runtime. The behavior of an agent can be altered by making it acquire or leave roles dynamically at runtime, thus easily adapting to new environmental conditions.

3 Agent Web Services Market Framework

Agent Web Service Market (AWSM) is a framework where companies/organizations can provide web services and customers can search for and purchase them. A role

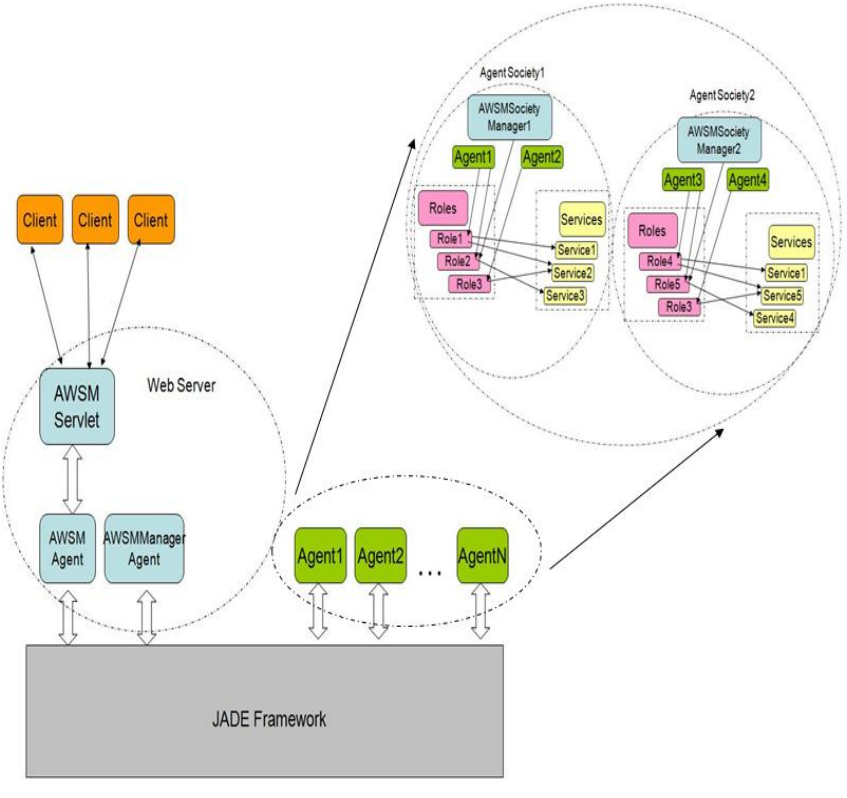

Fig. 1. AWSM Framework Architecture

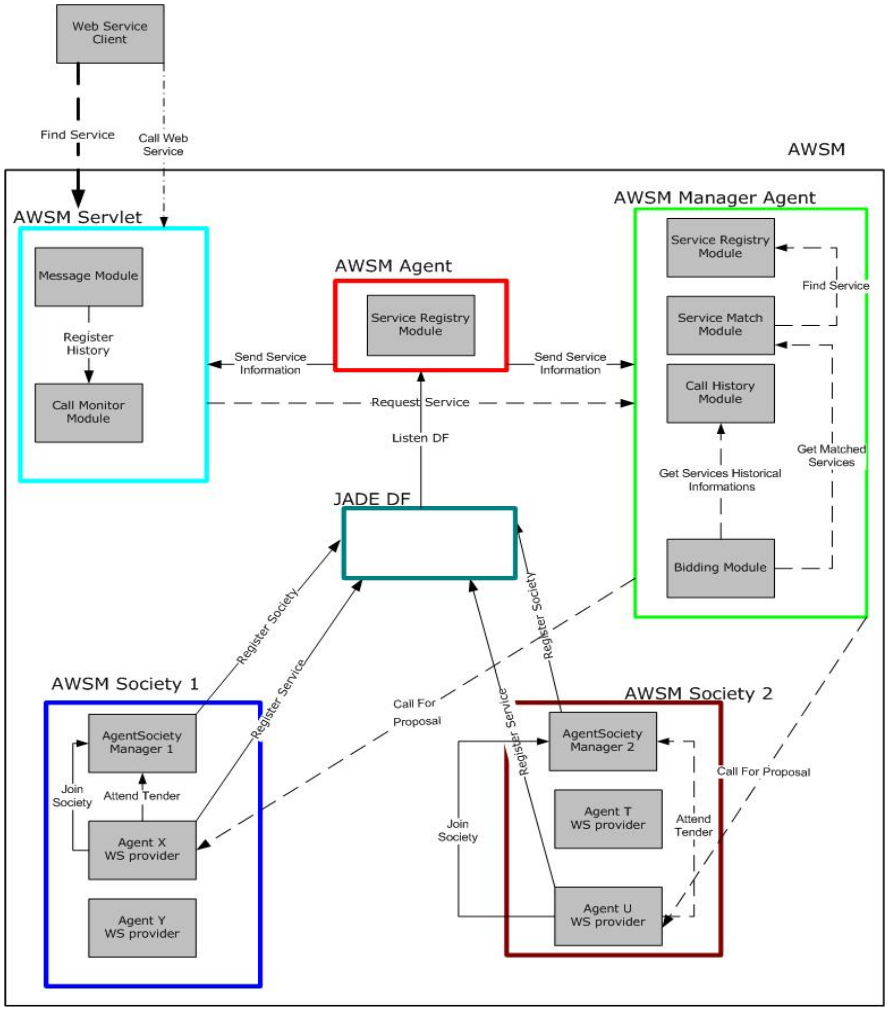

Fig. 2. AWSM Internal Structure

based agent system supports the framework. Service providers are organizations with well defined behaviors that cooperate/interact to fulfill the goal of the organization. In AWSM, each of these behaviors is represented by a role. These roles are defined by the organization, which also specifies how and in which context roles interact. Roles are played by agents. Thus, an agent may participate in an organization by acquiring a certain role in that organization. An agent can also play multiple roles in an organization and different roles in other organizations as well. There exists a strong correlation between roles and services; actually an agent that plays a certain role fulfills the activities expected of that role through well-defined web services. An agent may acquire several roles (provide several services) or lose some of its roles (cease providing particular services). The agent together with its roles is seen and manipulated as a single

entity and agents that cooperate to fulfill the goals of an organization form an Agent Society that is composed of agents, roles and services. Agents can dynamically enter and leave agent societies, thus adapting to conditions of real-life environments Each agent society has an AgentSocietyManager, to which agents consult in order to register their services as web services, to take and leave roles, or to attend biddings. Role definitions are given in XML files.

Figure 1 depicts the architecture of the framework. An AWSM Servlet resolves communication issues with clients. The AWSM Agent creates web service definitions for agents' services, while the AWSM Manager agent is responsible of finding requested web services, arranging bids, and logging histories of service calls. A client looking for a particular web service addresses the AWSM Manager Agent, providing parameters which include descriptive information of the requested service, cost conditions and service level information. AWSM is built upon the JADE agent framework [8]. The internal structure of AWSM is depicted in Figure 2.

3.1 AWSM Servlet

The AWSM Servlet running in a web server is the interface between clients that request web services and organizations that provide those services. Client requests expressed as SOAP [9] messages are transferred to corresponding agents by the AWSM Servlet. The modules (Figure 2.) the AWSM Servlet consists of and their functionality is stated below:

3.1.1 Message Module

- Receive /Reply HTTP/SOAP request messages
- Transmit requests and related information to an AWSM agent
- Transform request results into a SOAP messages
- Prepare HTTP/SOAP replies and send them to requesters

3.1.2 Call Monitor Module

This module monitors service calls. It sends service calls information to AWSM Manager Agent. Service calls in system, except for findService, contain key values which describe services and call instance. Call statistics are gathered by using these key values and forwarded to AWSM Manager Agent.

3.2 AWSM Agent

The AWSM agent is the gateway between agents and the web. It converts agent services into web services by creating their descriptions in Web Service Definition Language (WSDL) [10] and sends relevant information about thus converted web services to the AWSM Manager agent. The AWSM agent also transmits service request to agents. The AWSM Agent is responsible of the following tasks:

- Forward the information received from AWSM Servlet to related agents.
- Convert an agent's service proposal into a WSDL description and publish it.
- Create AWSM Manager agent at system startup.
- Keep the AWSM Manager agent informed about agents that provide services.

3.3 AWSM Manager Agent

The AWSM Manager agent handles service discovery requests initiated by web clients. This agent, after being created by the AWSM agent, registers *findservice* to the system as a web service it provides. This web service is called by clients who want to locate services that match their request criteria. The AWSM Manager agent evaluates requests it receives, arranges and finalizes bids to find appropriate services that meet the requests and sends results to clients. The bidding process will be described in detail in the following sections. The AWSM Manager agent consists of following modules (Figure 2.):

3.3.1 Service Registry Module

This module is responsible for registering and removing web services. It keeps information about services, such as the identity of the service provider agent, service name, service description, and service wsdl path. Other modules interact with service registry module to get service information. To register and remove agents and services the calls listed below are used.

registerService(serviceName,agentId,serviceInformation)
deregisterService(serviceName,agentId)

3.3.2 Service Match Module

This module is responsible for finding suitable candidate services which match requested service criteria. It uses serviceName, serviceDescription, keywords and previous similar service search. The following calls are used to find services that match request criteria.

matchByServiceName(serviceName)
Finds services which possess the same name.
matchByServiceDescription(serviceDescription)
Finds services which have similar description.
matchByKeyword(keywords)
Finds services which are described by similar keywords.
matchBySimilarSearch(serviceName,serviceDescription,keywords)

All service requests and results are registered in the system. This information is later used to shorten the matching process when service requests with similar criteria are received. The last three methods calculate similarity by calculating the percentage of words that are similar with input words.

3.3.3 Call History Module

This module is responsible for gathering and storing service call history information on performance details such as availability rate and response time of previous service calls that have completed. This module interacts with AWSM Servlet to get service call details. The agent's activity is monitored to assess its performance, i.e. how satisfactory has been the service provided, and to use its performance in new tenders. The assessment is based on the agent's average response time and the average availability

rate for previous calls. The fulfillment rate of previous commitments of the agent is also considered. The historical performance of a service provider agent is calculated by the call given below, each assessment contributing equally.

calculateHistoricalPerformance(serviceId)

3.3.4 Bidding Module

This module is responsible for service bidding. After a client's request for a service is receieved, bidding module starts a bidding process. It interacts with the other three modules to gather information necessary to organize a bid for the requested service. After locating candidate service provider agents, it announces a tender to determine the best fit for the request. The bidding process will be described in detail in Section 5.

3.4 Web Service Agent

Agents that take part in agent societies (members of an organization) provide web services, attend bids and fulfill service requests they have accepted in the system. They carry roles which determine their ability to provide certain services. A role may support one or more services. After receiving a new bid proposal request, an agent decides whether to attend the bid or not by consulting the AWSMSocietyManager. Since several agents in an agent society may carry similar roles, which means they may provide the same service, it may be the case that more than one agent in the agent society will receive the same bid proposal request. As it may not be desirable for two agents in the same society to attend the same tender, AWSMSocietyManager makes a decision on which agent participates the bid. Also, the AWSMSocietyManager may conclude that it is not useful for the society to attend new tenders based on the current work load and reaches its decision based on the above considerations. The agent carries out the web service if it is awarded the bid.

3.5 Client

Clients are outer components that request and consume web services. A client may be any application that is capable of calling web services. Clients refer to the *findservice* web service provided by the AWSM Manager, providing request criteria, to locate a suitable web service provider agent. If one that meets requested conditions is determined, *awardId* which specifies service provider agent is returned and the client can proceed with the service call over *awardId*.

4 Services, Roles and Agents in AWSM

The main contribution of AWSM is the role based agent model that supports the system. Agent roles are closely associated with the services they provide. The dynamic nature of agents and their ability to take or leave roles during their lifetime fits very well into the concept of real world services market. The following sections elaborate on the relations between agents, roles and services.

4.1 Service-Role-Agent Relation

In AWSM, a role is a triple that consists of services, protocols and permissions. The details of a service are the determined by the problem definition. System design is based on services at atomic level. A protocol defines the nature of interactions between roles and agents. Permissions describe roles' privileges to access resources. Permissions and privileges are represented by XML documents.

$S= \{s_1, s_2, s_3, \ldots, s_n\}$, where S is the set of all services in the system.

Services carry out actions to fulfill the responsibilities of a role. Services can be registered as a web service by agents. A single or a set of services may belong to a role. An agent may carry a single role (r_i) or a multiple of roles.

Role= {Services, Protocols, Permissions}

$R= \{r_1, r_2, r_3, \ldots, r_n\}$, where R is the set of all roles in the system and r_i is an individual role which can consist be fulfilled by one or more services. Thus, as an example, three roles, r_1, r_2, and r_3 can have the following composition:

$r_1 = \{s_1, s_2\}$

$r_2 = \{s_1, s_4\}$

$r_3 = \{s_5\}$

Agents that belong to a certain organization form an agent society. Each agent society is managed by an AWSMSocietyManager agent.

Agent Society: One or more agents form an agent society that is considered to be compatible with an organization or a company in real life. Agents that compose a society have certain responsibilities they carry out. Agents in a society can gain roles with the permission of AWSMSocietyManager. After acquiring a role, an agent can register services that its role supports as web services. Agents can attend web service bids with the approval of AWSMSocietyManager. AWSMSocietyManager records agents' previous performance in tenders. AWSMSocietyManager also requests historical call performance values of its agents from the AWSM Manager Agent occasionally.

Agents in an Agent Society: Agents are included in an agent society in order to provide certain services and they are assigned roles which give them the rights and ability to fulfill these services. Agents need the approval of AWSMSocietyManager before they can participate in a society.

Roles in Agent Society: The AWSMSocietyManager agent is responsible of role administration.

4.2 Service Definition and Registration

As stated above, a role can consist of more than one service. Services can be added or deleted from a role definition at runtime. Web service definition can be extracted from service classes that are added or deleted from role definitions at runtime. Java annotation mechanism is used to define services. Annotation is a mechanism for associating a meta-tag with program elements and allowing the compiler or the VM to extract program behaviors from these annotated elements and generate interdependent codes when necessary.

Table 1. Annnotation Classes and Their Descriptions in AWSM.

Name	Description
WebService	To define web service and its' properties such as service name, keywords etc.
WebServiceOperation	To define web service operations and its' properties such as operation name and description
WebServiceParameter	To define operation input parameters.
WebServiceResult	To define operation output properties.

The annotations defined in the system are listed in Table 1. Annotations to define services and operations are as the following:

```
//Web service annotation to define service and its properties
@WebService(serviceName="InvoiceService",serviceDescription="invoice detail",
keywords="invoice detail, invoice information")
public class InvoiceService extends Service {
                    public String serviceName="InvoiceService";
                    public String getServiceName() {
                    return serviceName;
}
//Operation annotation to define operation and its properties
                    @WebServiceOperation(operationName="invoiceDetail",operationDescription
"invoice detail", keywords="invoice detail,invoice information")
                    //Service Result annotation to describe operation output
                    @WebServiceResult(resultName="invoiceDetailResult")
                    //Service Parameter annotation to define operation input
                    public String invoiceDetail( @WebServiceParameter( parameterName =
"invoiceId") String invoiceId){
                              return new String("InvoiceId:"+invoiceId+" detail");
}
}
```

An agent gets a role after its application for that role is accepted. Next, it forms a service definition XML document according to service classes. RoleServiceManager class converts the class definitions into XML service definitions by using the annotations described above and through other programming methods. The agent sends this definition to Directory Facilitator (DF). AWSM Agent that listens on DF receives the service definition and converts it into WSDL and stores it as a web server. After this procedure completes, AWSM Agent sends the service definition to AWSM Manager Agent, which has the responsibility of locating/matching web services according to the requests of web service clients. AWSM Manager Agent registers the service and stores detailed information about the agent that provides service.

5 The Bidding Process in AWSM

The AWSM Manager announces bids to determine services that meet call request criteria specified by clients. Firstly, a list of candidate services is generated following the steps described in the following section. Next, the procedure for the bidding process is executed to find the most suitable web service and its provider agent.

The aim of the bidding process is to determine the best fit service and provider agent. The criteria that are considered are as follows:

- Do candidate services fit the requested service?
  - Service Name match is used
  - Service Description match is used
  - Keywords match is used
- How much does the candidate provider satisfy requested criteria (quality of services, commission rates, etc.)
  - Requested criteria are used in bidding

Even though those criteria would be sufficient to determine agents that meet the requested criteria, they cannot evaluate how the winner agent fulfills commitments. To overcome insufficiency, historical performance of service provider agents is taken into account as well in bidding process to evaluate fulfillment of commitments.

5.1 Service Request Criteria

When applying for a web service, a client is required to specify the set of parameters listed below so that the most suitable service registered in the framework can be determined.

serviceName: The name of service the client is requesting
serviceDescription: The description of service the client is requesting. AWSM Manager agent uses this information to locate candidate services in it's service repository , in case a match with the requested service name can not be possible.
keywords: Keywords that will help to recognize appropriate web services
commisionRate: The commission rate that the client is willing to pay for the service call. A maximum value (commisionRate max) should be specified.
fixedCommisionValue: The fixed commission value that client is willing to pay for the service call. If the commission value calculated by using commisionRate exceeds the specified fixedCommisionValue, the client agrees to pay the fixedCommisionValue, not the calculated commission value. fixedCommisionValue max value should be specified.

The following are the parameters that determine the service level quality:
invocationCount: The number of times the service will be invoked monthly. Both of invocationCountmax and invocationCount min values need to be specified.
availabilityRate: Required availability rate of service. availabilityRatemin value should be specified.
responseTime: Required response time of service. responseTimemax value should be specified.
averageResponseTime: Required average response time of service. $\text{AverageResponseTime}_{max}$ value should be specified.

5.2 Determining Service Provider Candidates

AWSM applies three methods to determine a list of candidate service provider agents:

c_a: Match service with the parameter *serviceName*
c_b:Search service repository according to the *serviceDescription* and *keywords* and find matches.
c_c: Return services that were selected on a previous similar service request

With the execution of each method, candidate agents are determined and C, the set of candidate service provider agents, is computed.

$C= \{c_a, c_b, c_c\}$

5.3 Bidding Process

When the AWSM Agent receives a request with the parameters stated above, it initializes a bidding process following the procedure described below:

1. Determine a set of candidate service providers that match the request criteria,
2. Get performance history data of candidates and calculate their performance values,
3. Start bidding by calling for proposals from candidates
4. Receive bids from bidders,
5. Evaluate bids by using their offers and historical performance values. Choose the best n (system parameters) bids. If there are no suitable bids, end the bidding process with failure.
6. If there are n bids, repeat the bidding process k times, each time starting from step 3. k is the system parameter that shows how many times new proposals for bids are requested.
7. Send the winner agent an "awarded" message with a newly generated *awardId*. *awardId* will be used with web service calls by the client.
8. Send a result message to the web service client with related information.

5.4 Service Calls by Client

Clients use *awardId* to be logged and be authorized when calling the awarded agent's web services. It is not allowed to call services with an invalid *awardId*. The call and its performance data such as service response time are logged in AWSM to be used in later tenders. The performance of agent's service should be consistent with the proposal it has made to win the tender. If it difference is above a predefined limit, it will produce a negative effect on the agent's historical performance in future tenders.

6 Conclusion

This paper describes the design and implementation issues of a new web services market framework supported by role based agents. The framework provides dynamic web services and allows its clients to find favorable web services through competitive bidding.

Integrating the agent model of computation with web services results in an environment where web services are more dynamic and autonomous. Cooperation, coordination and configuration of services can be accomplished easily and the burden is not reflected to consumers.

Using role-based agents contributes to system modeling, flexibility, and reusability. With the presented framework, agent societies (organizations) gain the ability to provide services and easily modify their internal structures dynamically via agents that can take or leave roles at runtime. A prototype of the framework has been completed on JADE platform. Currently work on the implementation of real world services and performance measurements of the system are in progress.

References

1. Papazoglou, M.P.: Service-Oriented Computing: Concepts, Characteristics and Directions. In: Proceedings of Fourth International Conference on Web Information Systems Engineering, WISE 2003 (2003)
2. Blake, M.B.: Forming Agents for Workflow-Oriented Process Orchestration. In: Workshop on Electronic Commerce, Agents, and Semantic Web Services in conjunction with the International Conference on Electronic Commerce, ICE 2003 (October 2003)
3. Rykowski, J., Cellary, C.: Virtual Web Services-Application of Software Agents to Personalization of Web Services. In: ICEC 2004, Sixth International Conference on Electronic Commerce (2004)
4. Huhns, M.N.: Agents as Web Services. IEEE Internet Computing 6(4), 93–95 (2002)
5. Wang, Y., Wang, H., Deng, J., Zhao, X., Yung, K., Gao, S.: Web-Service-Agents-Based Securities Trading Simulation System. In: PACIS 2005 Proceedings. Paper 32 (2005)
6. Brazier, F.M.T., Richards, D., Sabou, M., Van Splunter, S.: Composing Web Service Using An Agent Factory. In: Proceedings of AAMAS Workshop on Web Services and Agent-Based Engineering (WSABE), Melbourne, Australia, pp. 57–66 (2003)
7. Valle, M., Ramparany, F., Vercouter, L.: A Multi-Agent System for Dynamic Service Composition in Ambient Intelligence Environments. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) PERVASIVE 2005. LNCS, vol. 3468. Springer, Heidelberg (2005)
8. `http://jade.tilab.com`
9. `http://www.w3.org/tr/soap/`
10. `http://www.w3.org/tr/wsdl`