A Web Services Market Framework with Role Based Agents

Ali Durmus
Computer Engineering Department
Electrical-Electronics Faculty
Istanbul Technical University, Istanbul, Turkey
adurmus@etiya.com

Nadia Erdogan
Computer Engineering Department
Electrical-Electronics Faculty
Istanbul Technical University, Istanbul, Turkey
nerdogan@itu.edu.tr

***Abstract*—As the internet becomes widespread, many companies have started to present their services through the internet to reach more customers. They have increasingly been using web services in their applications to cope with the complexity and heterogeneity of the internet. Companies not only sell, but they also buy services over the internet. We present a new web services market framework implemented through role based agents to provide a suitable and efficient infrastructure for presenting, selling and buying web services.**

This paper describes the design and implementation details of a framework named Agent Web Service Market (AWSM) and a new role based agent model. In AWSM, companies are represented by agent societies which are composed of agents that possess roles to reflect companies' dynamic structure. A role is a collection of services which can be taken and left dynamically by agents at runtime. The structure of a role can also change during execution time. Depending on their roles, agents gain the ability to provide services, which they can register as web services. A client in search of a particular web service can look up for web services provided by agents on the. The client requesting the web service also provides service information, service level parameters and cost criteria. The framework puts out a tender to determine a web service that meets those criteria.

***Keywords*— Roles, Role-based Agents, Multiagent Systems, SOA, Web Services, Service Bidding**

I. INTRODUCTION

Currently, there are several web services that carry out different tasks on 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 on 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 on the markets are very dynamic in real life. Those that provide service can enter and leave the market. They may change their structure and business fields. They may hire people who take responsibilities (roles) and provide services accordingly or they may dismiss employee, in which case, the company should continue to provide services which were fulfilled by dismissed employee. A dynamic web services market is required to fulfill the needs of companies and the market. Web services and classical programming techniques are very static software entities; they are neither aware of the environment nor autonomous. They can't change their structure according to changing environmental conditions. What is needed is a way of handling the dynamic nature of companies and the market in such a manner that these changes are not reflected to services consumers.

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. Although an agent oriented implementation provides the means for more dynamic, autonomous and cooperative web services, changes in companies' inner structure (changing position of employee, hiring/dismissal of employee) can not be handled easily. AWSM adopts a role based approach to overcome these difficulties. Web services are provided by agents which have acquired the roles appropriate for those services. Dynamic role management at runtime reflects well the dynamism in the structure of companies.

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. Agent societies represent companies in real life. Agents that are members of an agent society 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 a tender. 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. The advantages that are considered during the design of the framework are briefly stated below.

A. Service Oriented Architectures

Service Oriented Architectures (SOA) have very widespread usages. They are used in individual services like book reservations, as well as in complex services such as hotel and flight reservation. Service Oriented Computing is a paradigm based on services for application development [1].

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.

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].

B. Roles

Role is person's defined and expected behavior patterns which should be performed by his/her position privileges in an environment [4]. Individuals represent specific positions that give them the right and responsibility in organizations. Roles can be defined as collections of abilities and expected behaviors gained by having those positions. Role, in context of software, is object behaviors, a conceptualization of a constraint set with a subset of object interactions [5]. Role is also set of privileges of access and operations. Enhancing agents with roles adds agents new behaviors, new coordination and cooperation abilities and allows multi agents systems to be flexible and reusable. A role represents an agent's goal and agents may acquire and lose roles as their goals change.

II. MOTIVATIONS AND BACKGROUND

Companies that sell services need a web services market to publish, receive requests for and 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 also, 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 [6], 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 [7] 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 [8] 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.

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 represents companies. 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 (companies) 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.

III. AGENT WEB SERVICE MARKET FRAMEWORK

Agent Web Service Market (AWSM) is a framework where services are present and sold to clients. The agent system that supports the framework is role based. Roles are composed of services. An agent may acquire several roles (provide several services) or lose some of its roles (cease providing particular services). The agent and its roles are seen and manipulated as one entity and agents with related roles form an Agent Society. 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.

Fig.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. Agents that provide related services are assembled into Agent Societies that are composed of agents, roles and services. AWSM is built upon the JADE agent framework [9].

IV. ROLES, SERVICES AND AGENTS

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 same role can be in different structures in different companies. The structure of a role may also dynamically change. The following sections elaborate on the relations between agents, roles and services.

A. *Service-Role-Agent Relation*

As stated above, a role represent an objects' services, a conceptualization of a constraint set with a subset of object interactions. Services define role ability and behaviors. A role can be composed of one or more services.

In AWSM, a role is a tuple that consists of services, attributes, protocols, permissions.

Role= {Services, Attributes, Protocols, Permissions}

The details of a service are the determined by the problem definition. Services are not only web services but services that an agent should provide and are extracted by using service oriented architectures principal. For example, flight ticket service which will be web service while a organization of an auction is an internal service in the system. System design is based on services at atomic level. Each service specified in the system is not bound to only a single role. Since some roles can involve common services to fulfill, the same service can be assigned to different roles. For example, Flight Ticket Sell Service can be assigned both Sales Representative Role and Sales Manager Role in travel agency companies.

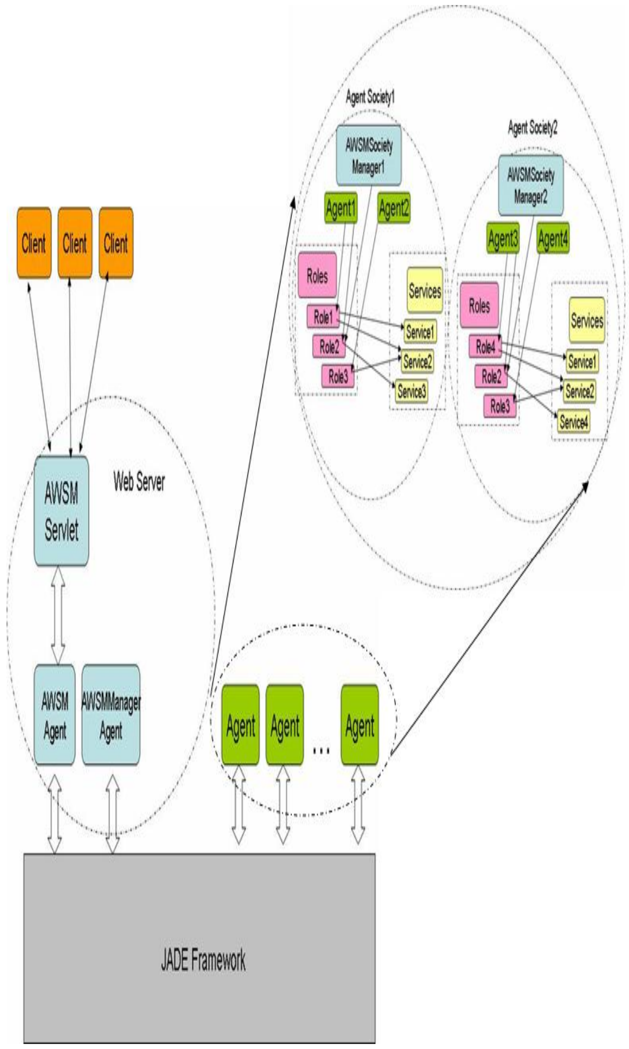


Figure 1. AWSM Framework Architecture

S= {s_1, s_2, s_3, 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, each consisting of a set of services.

R= {r_1, r_2, r_3,r_n}, where R is the set of all roles in the system.

r_1 can be {s_1, s_2}

r_2 can be {s_1, s_4}

r_3 can be {s_5}

Attributes defines properties of roles. Roles can have many attributes such as name, a description.

A protocol defines the nature of interactions between roles and agents. Interactions are among distinct features of the agent systems. An agent behaves according to its roles, in the context of the protocols defined. These can be auction protocols which are used to implement auctions or communication protocols. There can be more than one auction protocols to reflect different auction strategies in the system.

Permissions describe a role's rights and privileges to access resources. A role enhances the agent that carries it with certain abilities. Constraints on these abilities are defined by permissions. For example, attending a tender may be permission and is defined role xml files.

Groups of agents which cooperate and coordinate form agent societies. Each agent society is managed by an AWSMSocietyManager agent. A set of roles are assigned to AWSMSociety.

AWSMSociety= {Roles, Agents}.

1) 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. Roles can be taken by a maximum predefined number of agents. This maximum value is set in role definition XML file by the roleMaxTakes attribute. Some roles are taken by AWSMSocietyManager agent at system start. These roles are determined by the roleManagerDefault value in role definition XML file.

AWSMSocietyManager loads role XML files at the initialization of the system. It can also add new role definitions and change existing role definitions at runtime. AWSMSocietyManager registers it's society as a JADE DF(Directory Facilitator) agent. Incoming agents with a request to join a society consult DF asking for open roles from AWSMSocietyManager agent. After receiving and evaluating possible roles, an agent applies for a role, sends its qualifications with application too. AWSMSocietyManager evaluates applications and makes decisions, either accepting or rejecting an applications. An agent can acquire roles only with the permission of AWSMSocietyManager Agent.

AWSMSocietyManager keeps records of an agent's previous performance in tenders. AWSMSocietyManager also requests historical call performance values of its agents from the AWSM Manager Agent occasionally.

2) Agent 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 the ability to fulfill these services. Agents have to have required criteria to be accepted in an agent society. Agents need the approval of AWSMSocietyManager before they can participate in a society. An agent can acquire new roles or leave existing roles dynamically at runtime as changing conditions impose such modifications. An agent can also carry one or more roles. Agents can register their services as web services according to the specifications of the roles. If agents receive bid request, they checks their attendTender permissions to see if they are allowed to attend a tender or not. If agents have permission to attend a tender, they cooperate with AWSM Manager agent to proceed with tender operations. Agents determine their strategy in a tender according to the protocols they possess. Protocols may differ according to the type of auction method implemented. If agents are awarded a tender, they have to fulfill service call made by clients.

3) Roles in Agent Society: Roles are assigned to agent societies. A set of roles in agent society corresponds to available positions in a company. As in real life, every position can be represented via one or more roles. The AWSMSocietyManager agent is responsible of role administration. Agent societies have a predefined set of roles which are administrated by AWSMSocietyManager agent. AWSMSocietyManager agent can change service role assignments at runtime, thus reflecting the dynamic nature of the execution environment.

B. An Example Depicting Service-Role-Agent Relation

Now, we give an example to show how a real life company can be modeled to function under AWSM. Let Unicorn be an airline flight ticket booking and selling company. Unicorn can reserve and sell different airlines tickets. Customers can call Unicorn to ask flight information or post sale support. Unicorn is a small company with four employees. In order to have Unicorn to work and sell service at the AWSM, we can model Unicorn according to our service based role system in the following manner.

Firstly, a set of services S, listed below, are defined according to Unicorn's business description and the company's internal tasks. S is the set that includes all of the services provided in the system S= {s_1, s_2, s_3, s_4, s_5, s_6, s_7}

- s_1= Flight Ticket Sell Service
- s_2= Flight Ticket Reservation Service
- s_3= Project Management Service
- s_4= Human Resource Service

- s_5= Payment Service
- s_6= Billing Service
- s_7= Help Desk Service

Next, the roles are determined. R is the set that includes all of the roles present, R= $\{r_1, r_2, r_3\}$

- r_1= Manager Role
- r_2= Sale Manager Role
- r_3= Sale Representative Role

Roles and services are matched by using the problem definition. One role can consist of one or more services. We make the following assumptions:

Permissions of the systems (P) specifies

- the number of agents that can carry a certain role.
- roles which can attend tenders
- manager default role

Attributes of the system (A) : Role Name

Protocol (Pr): Simple auction protocol

Thus, roles in the system are described by the following tuples:

$r_1= \{\{s_3, s_4\}, A, Pr, P\}$

$r_2= \{\{s_1, s_2, s_3\}, A, Pr, P\}$

$r_3= \{\{s_1, s_2, s_5, s_6, s_7\}, A, Pr, P\}$

Assuming that one manager, one sales group manager and two sales representatives work at Unicorn, the corresponding agents in the system would be:

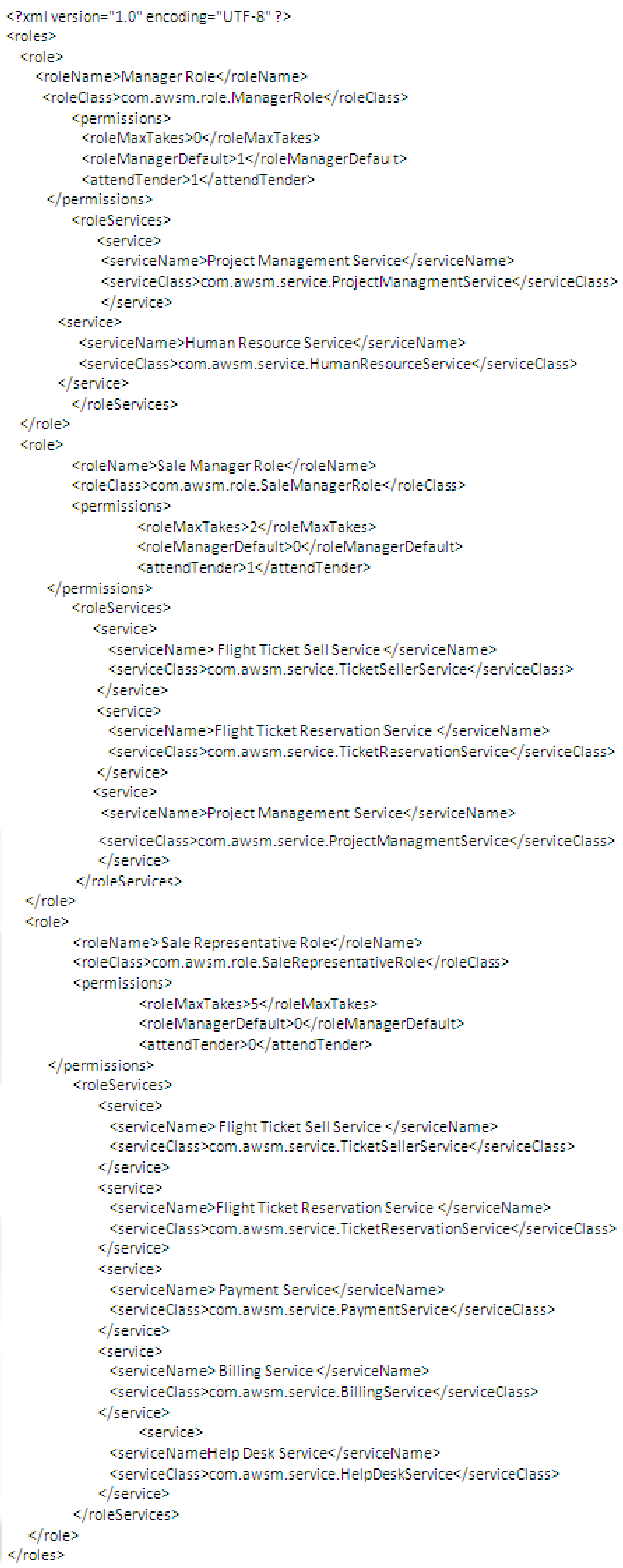
- $Agent_1$: AWSMSocietyManager
- $Agent_2$: Sale Group Manager
- $Agent_3$: Sale Representative
- $Agent_4$: Sale Representative

With agents assigned roles as listed below:

- $Agent_1= \{r_1\}$
- $Agent_2= \{r_2\}$
- $Agent_3= \{r_1, r_2\}$
- $Agent_4= \{r_1, r_2\}$

In this example, some of the services may be registered as web services, such as s_1, s_2, s_7. Those services can be sold to clients, which can be applications, over the internet.

According to above definitions role xml files shown in Figure 2. will be constructed.



```
<?xml version="1.0" encoding="UTF-8" ?>
<roles>
  <role>
    <roleName>Manager Role</roleName>
     <roleClass>com.awsm.role.ManagerRole</roleClass>
         <permissions>
           <roleMaxTakes>0</roleMaxTakes>
           <roleManagerDefault>1</roleManagerDefault>
           <attendTender>1</attendTender>
      </permissions>
         <roleServices>
             <service>
              <serviceName>Project Management Service</serviceName>
              <serviceClass>com.awsm.service.ProjectManagmentService</serviceClass>
              </service>
       <service>
          <serviceName>Human Resource Service</serviceName>
          <serviceClass>com.awsm.service.HumanResourceService</serviceClass>
       </service>
         </roleServices>
  </role>
  <role>
         <roleName>Sale Manager Role</roleName>
         <roleClass>com.awsm.role.SaleManagerRole</roleClass>
         <permissions>
                    <roleMaxTakes>2</roleMaxTakes>
                    <roleManagerDefault>0</roleManagerDefault>
                    <attendTender>1</attendTender>
      </permissions>
         <roleServices>
            <service>
              <serviceName> Flight Ticket Sell Service </serviceName>
              <serviceClass>com.awsm.service.TicketSellerService</serviceClass>
             </service>
             <service>
              <serviceName>Flight Ticket Reservation Service </serviceName>
              <serviceClass>com.awsm.service.TicketReservationService</serviceClass>
             </service>
            <service>
             <serviceName>Project Management Service</serviceName>
             <serviceClass>com.awsm.service.ProjectManagmentService</serviceClass>
             </service>
          </roleServices>
   </role>
   <role>
         <roleName> Sale Representative Role</roleName>
         <roleClass>com.awsm.role.SaleRepresentativeRole</roleClass>
         <permissions>
                    <roleMaxTakes>5</roleMaxTakes>
                    <roleManagerDefault>0</roleManagerDefault>
                    <attendTender>0</attendTender>
      </permissions>
         <roleServices>
             <service>
               <serviceName> Flight Ticket Sell Service </serviceName>
               <serviceClass>com.awsm.service.TicketSellerService</serviceClass>
             </service>
             <service>
               <serviceName>Flight Ticket Reservation Service </serviceName>
               <serviceClass>com.awsm.service.TicketReservationService</serviceClass>
             </service>
             <service>
               <serviceName> Payment Service</serviceName>
               <serviceClass>com.awsm.service.PaymentService</serviceClass>
             </service>
             <service>
               <serviceName> Billing Service </serviceName>
               <serviceClass>com.awsm.service.BillingService</serviceClass>
             </service>
                    <service>
               <serviceNameHelp Desk Service</serviceName>
               <serviceClass>com.awsm.service.HelpDeskService</serviceClass>
             </service>
         </roleServices>
   </role>
</roles>
```

Figure 2. Role-Service Definition

C. ROLE XML

In our framework, role definitions are described in the form of XML files. New XMLdocument definitions are created. After determining roles, services and their relation and permissions in the design phase, these values can be defined by using XML templates. Role name, role class, role related services and permission are defined in XML files.

AWSMSocietyManager agents load XML file definitions during initialization to determine the roles in the society. AWSMSocietyManager agent can change role-service definitions in runtime by using this document.

Using XML files to define role-service relations enhances the flexibility of the framework. Definitions can be changed very easily and also users can read and understand definitions easily, too.

D. Agents and Agent Society in Bidding Process

When an agent receives a bid request, it checks its *attendTender* permissions to decide how it should act. If *attendTender* values permit the agent to attend a tender, it applies to the AgentSocietyManager of its agent society to get permission to participate in the bid. It also gets a range of cost values which will be used in tenders. If it needs to exceed its range, it has to reapply the AgentSocietyManager again for new range limits. An agent determines its bidding strategies according to its roles.

AgentSocietyManager allows agents to attend tenders according to previously defined descriptions. If more than one agent request permission, it selects candidate agents considering their previous tender performance and historical call performance. It also determines the range of costs values for agents in biddings.

V. CONCULUSION AND FUTURE WORK

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.

Using agent and web services together makes web services more dynamic and autonomous. Web services cooperation, coordination and configuration can be accomplished easily and the burden is not reflected to consumers.

Using role-based agents contributes to system flexibility, and reusability. Agent societies (companies) gain the ability to provide services and easily modify their internal structures dynamically via agents that can take or leave roles at runtime.

Role are composed of services and the role-service relation can also be changed at runtime which enhances the adaptability of the system to changing market or company conditions. A prototype of the framework has been completed on JADE platform and currently work on the implementation of real world services and performance measurements of the system is in progress.

VI. REFERENCES

[1] M. P. Papazoglou, ”Service-Oriented computing: concepts, characteristics and directions” Proceedings of Fourth International Conference on Web Information Systems Engineering (WISE’03), 2003.

[2] M. B. Blake, ”Forming agents for workflow-oriented process orchestration” Workshop on Electronic Commerce, Agents, and Semantic Web Services in conjunction with the International Conference on Electronic Commerce (ICEC2003), October 2003.

[3] J. Rykowski, C. Cellary ”Virtual web wervices-application of software agents to personalization of web services” ICEC’04, Sixth International Conference on Electronic Commerce, 2004

[4] D.I. Hawkins, R.J. Best, K. A. Coney, Consumer behavior, business publications, Inc., Plano, Texas, 1983

[5] G. Genilloud, A. Wegmann, “A foundation for the concept of role in object modeling”, Proceedings of the 4th International Enterprise Distributed Object Computing Conference(EDOC2000), Makuhari, Japan, September, 2000, pp. 76-85.

[6] M.N. Huhns, “Agents as web services” IEEE Internet Computing (6:4), 2002, pp. 93-95.

[7] Y. Wang, H. Wang, J. Deng, X. Zhao, K. Yung, S. Gao, "Web-service-agents-based securities trading simulation system" PACIS 2005 Proceedings. Paper 32.

[8] F.M.T. Brazier, D. Richards, M. Sabou, S. van Splunter. “Composing web service using an agent factory” Proceedings of AAMAS Workshop on Web Services and Agent-Based Engineering(WSABE). Melbourne, Australia, 2003, pp. 57-66

[9] http://jade.tilab.com