Negotiation Platform for Personalised Advertising

Luís Ventura de Sousa[1]
Malheiro, Benedita[1]
Foss, Jerry[2]

1st January 2012

1 – "Instituto Superior de Engenharia do Porto", Department of Electrical Engineering, Rua Dr. António Bernardino de Almeida, 431, P-4200 PORTO
{1900503, mbm}@isep.ipp.pt

2 – "Birmingham City University", Millennium Point, Birmingham, B4 7XG
jeremy.foss@bcu.ac.uk

Abstract

This paper describes a multi-agent brokerage platform for near real time advertising personalisation organised in three layers: user interface, agency and marketplace. The personalisation is based on the classification of viewer profiles and advertisements (ads). The goal is to provide viewers with a personalised advertising alignment during programme intervals.
The enterprise interface agents upload new ads and negotiation profiles to producer agents and new user and negotiation profiles to distributor agents. The agency layer is composed of agents that represent ad producer and media distributor enterprises as well as the market regulator. The enterprise agents offer data upload and download operations as Web Services and register the specification of these interfaces at an UDDI registry for future discovery. The market agent supports the registration and deregistration of enterprise delegate agents at the marketplace.
This paper addresses the marketplace layer, an agent-based negotiation platform *per se*, where delegates of the relevant advertising agencies and programme distributors negotiate to create the advertising alignment that best fits a viewer profile and the advertising campaigns available.
The whole brokerage platform is being developed in JADE, a multi-agent development platform. The delegate agents download the negotiation profile and upload the negotiation results from / to the corresponding enterprise agent. In the meanwhile, they negotiate using the Iterated Contract Net protocol. All tools and technologies used are open source.

Keywords: *Multi-agent computing, brokerage, automated negotiation, Iterated Contract Net, Web Services, UDDI, JADE.*

1 Introduction

Context-aware computing is currently a major research topic driven by the availability, popularity, seamless network access and processing power of mobile devices. Mobile device applications exploit existing on board sensor data (GPS, compass, accelerometer, gyroscope, pressure, temperature or proximity sensors) and nearby devices to dynamically adapt to the location, available resources, user interests, *etc*. Furthermore, they allow service providers to build and maintain user profiles based on the user interactions and on the user context data collected. Personalisation, user recommendation or location-based services are examples of popular context-aware computing systems.

The work herein described expects content distributors to build and maintain viewer profiles in order to provide viewers with a personalised advertising alignment during programme intervals. The goal is, thus, to develop a multi-agent brokerage platform for near real time advertising personalisation [1]. It is a component of a larger system intended for the automated near real time content personalisation. The architecture for this broader scenario – networked video personalised placement – is depicted in Figure 1.

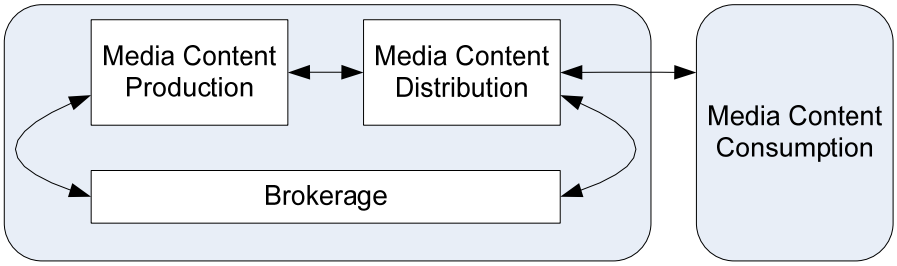

Figure 1: Personalised placement overview

This multi-tier architecture is constituted of four main tiers: the content production tier, the content distribution tier, the content consumption tier and the artefact brokerage tier. The key players are the producers, the distributors and the viewers.

End-user clients (PC clients, set-top boxes) need to support object processing, *e.g.*, decoding and rendering needs to be supported in advanced codecs. All current video distribution formats are feasible, including Satellite Television (TV), Cable TV, Cable Internet Protocol TV (IPTV), Telco IPTV, WebTV and Digital Terrestrial TV. The video head-end will be unaffected by the requirement to host the source content stream.

The work described in this paper is concerned with the dynamic selection of the objects to be inserted in the viewer play-out stream.

2 Background

The brokerage tier is responsible for the dynamic selection of the objects to be inserted in the viewer play-out stream. This is achieved through automated agent-based negotiation involving the video content producers and distributors. This functionality is exposed to the involved parties as a Software-as-a-Service (SaaS) component [2].

Here, a service-centric model is proposed to provide producers and distributors an automated negotiation service based on Web Service interfaces, Service-Oriented Architecture (SOA) and SaaS approaches. Such combination is, according to [3], an interesting attempt to combine the strengths of SOA, Web Services, agent-based systems and instant messaging technologies. The idea of developing a multi-agent negotiation system integrated in a service-oriented architecture is feasible and meaningful for e-commerce oriented intelligent trading applications [4].

The producers and distributors of media content are modelled by autonomous intelligent agents. These so-called enterprise agents must, on one hand, be entirely controlled by their real world counterparts to ensure the privacy of the company strategic knowledge and, on the other hand, be fully compatible and interoperable with the remaining components of the framework. The latter is achieved by the adoption of a Web Service interface guaranteeing interoperability and allowing the creation of loosely coupled enterprise agents that can enter and leave freely the proposed transaction environment. The resulting SOA relies on Universal Description, Discovery and Integration (UDDI) service registries to hold the descriptions of existing agent services. On one hand, producers and distributors can publish, update and remove their service descriptions – metadata descriptions of the objects they hold or seek to insert in the viewer stream. On the other hand, any agent can discover, download and interact with any service (agent) automatically.

All video objects are MPEG-4 instances annotated in an MPEG-7 based multimedia ontology[1]. This applies both to the source video objects (the viewer-selected video streams) and to the external video objects (automatically selected and inserted by this framework).

2.1 Scenario

The scenario used for testing the platform is the personalisation of advertising content. In this context, content producers are advertising agencies eager to place their ads in the video streams of target viewers. Content distributors are video on demand providers that intend to offer viewers a personalised advertising experience. To achieve this goal, distributors have to build and maintain the viewer profiles. Viewer profiles and ads have been segmented into a set of predefined classes to ensure a straightforward matching. Viewer profiling as well as other context-aware inputs are not addressed in this paper.

[1] MPEG-4 and MPEG-7 are multimedia standards of the Moving Picture Experts Group (MPEG).

When a viewer selects a programme (video stream) from a distributor registered at the platform, he triggers the automated brokerage mechanism. The distributor representative queries the UDDI registry for producers with ads matching the viewer profile and sends out invitations to the discovered producers to participate in a dedicated negotiation. If the invitations are accepted, a collection of trading delegate agents are launched at the marketplace and the automated negotiation involving the viewer distributor delegate and the producer delegates begins. This procedure is repeated until the accumulated list of ads spans the planed interval timeslot. As a result, when the programme interval occurs, the distributor has a personalised advertising alignment ready to send to the viewer.

3 Brokerage Platform

The brokerage platform is a competitive Multi-Agent System (MAS) where enterprise agents (representing producer and distributor enterprises) and the market regulator meet in order to trade media components according to the negotiation profiles of the agents and the rules of the market. The resulting MAS is structured in three layers presented in Figure 2.

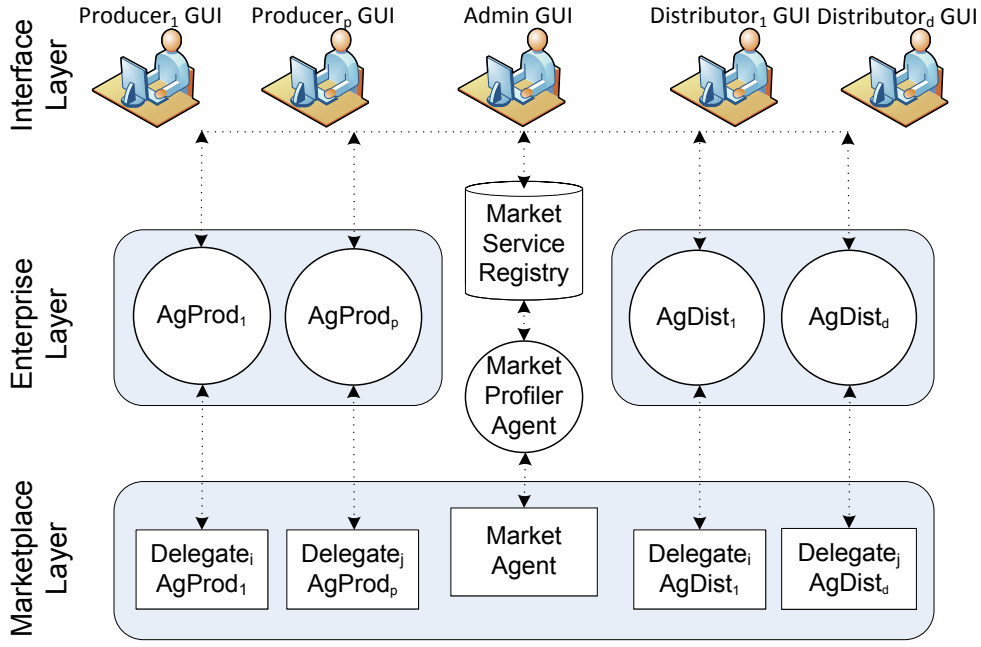

Figure 2: Brokerage platform architecture

The top layer is composed by the enterprise user interface agents, the middle layer is made of agents that model and represent each enterprise in the platform and the bottom layer represents the actual marketplace where the automated negotiation occurs. The top layer is the platform interface and holds a collection of enterprise user interface agents, one per each registered enterprise. The middle layer contains

coarse grain agents representing content producers and distributors – the enterprise agents – and the market profiler agent. The bottom layer is composed of the market agent and finer grain agents that are delegate enterprise agents. The brokerage platform includes, thus, the following agent categories:

Enterprise interface agents that allow enterprises to join and interact with the platform. They are responsible for taking the inputs, spawning or reconfiguring the enterprise agent and reporting back the results. A producer uses its producer interface agent to upload new ad features and market behaviours and to download the obtained results. A distributor defines through its distributor interface agent the current viewer profiles, the available advertising timeslots and market behaviours and retrieves the negotiation outcomes;

Enterprise agents that represent producers (*AgProd*) and distributors (*AgDist*) within the platform are coarse grain agents. They participate, upon invitation, in specific negotiations by launching delegate agents at the marketplace. The enterprise agents expose through Web Service interfaces the services required to interact with the other layers;

Market profiler agent that is responsible for defining the type of negotiation. It is controlled by the platform administrator and offers a Web Service interface;

Market agent that is the coordinator of the marketplace;

Market delegate agents that are small grain agents responsible for trading individual ads or timeslots on behalf of enterprise agents. Their ephemeral life terminates upon success or failure in the negotiation round for which they were invited to participate.

The brokerage platform MAS is being implemented using the Java Agent Development Framework (JADE) [5] [6], the Web Service Integration Gateway (WSIG) add-on [7], the Web Service Dynamic Client (WSDC) add-on [8] and the UDDI4J API to interact with Web Services. The UDDI service registry used is jUDDI [9], an open source Java implementation of the Universal Description, Discovery, and Integration (UDDI) specification for Web Services. The WSIG and jUDDI technologies are supported by the Apache Tomcat application server [10] and Axis 2/Java API for Web Services [11]. To maintain the jUDDI database, we use additionally the MySQL database server [12].

The enterprise agents offer Web Service interfaces that expose a collection of agent actions as Web Service operations to the marketplace. These actions were defined in a specific ontology created with the Protégé Ontology Editor [13] and imported into the platform using JADE Bean Generator plug-in for Protégé – the *NegPub* ontology. This ontology specifies a collection of agent actions (*RegisterAgent*, *GetProducerProfile*, *GetDistributorProfile* and *SetResults*) and data types (*Profile*, *ProducerProfile*, *DistributorProfile* and *Results*). Ontology mapper classes were defined to ensure that each enterprise agent exposes only the operations relevant to its functionality: producers expose *GetProducerProfile* and *SetResults* operations,

distributors *GetDistributorProfile* and *SetResults* operations and the market agent the *RegisterAgent* and *DeregisterAgent* operations.
The Web Service interfaces of the enterprise agents are deployed at the WSIG and registered at jUDDI. This approach allows the automated search, identification and invitation to the marketplace of the potential negotiation partners (advertising agencies) by programme distributors. Upon acceptance, the corresponding delegates are launched at the marketplace layer.
The interaction between the enterprise and the delegate agents on the second and third layers, respectively, rely on the Web Service Dynamic Client (WSDC) mechanism, *e.g.*, the delegate agents get the negotiation profile and report the negotiation results through WSDC.
The work detailed in this paper is concerned with the marketplace layer.

3.1 Marketplace

Since the interface and enterprise layers are currently under development, we have developed a temporary brokerage platform Graphical User Interface (GUI) where it is possible to create directly new producer and distributor agents – see Figure 3.

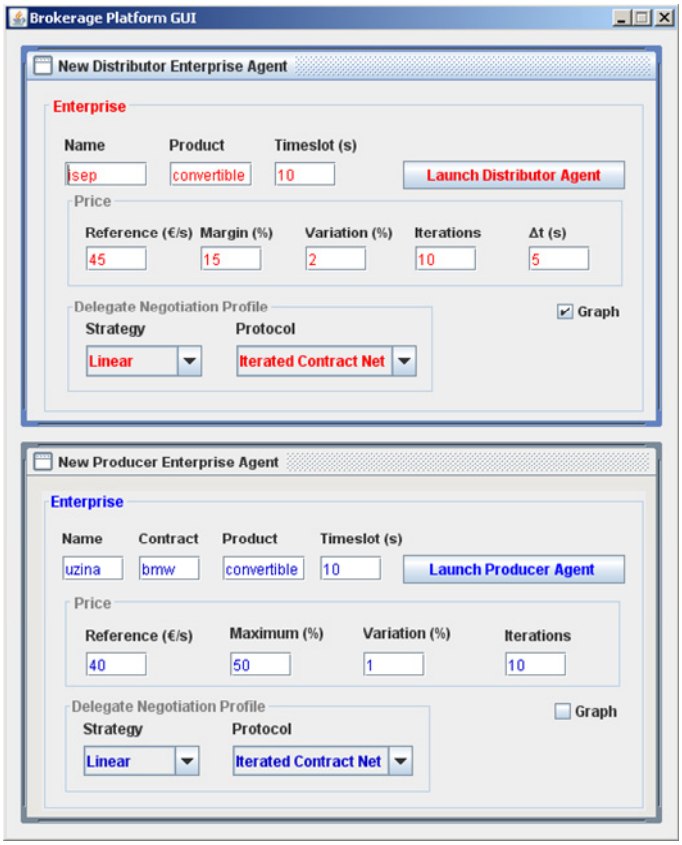

Figure 3: Brokerage platform GUI

The creation of producer and distributor agents requires the specification of the enterprise agent name, the ad properties (product category, price-related features and, in the case of a producer, the ad contract reference) and the negotiation behaviour (protocol and strategy). The negotiation strategy will be used to make, in the case of a producer delegate, new bids/proposals and, in the case of a distributor delegate, for evaluating the proposals received so far. Figure 4 shows the GUI of the distributor and producer agents.

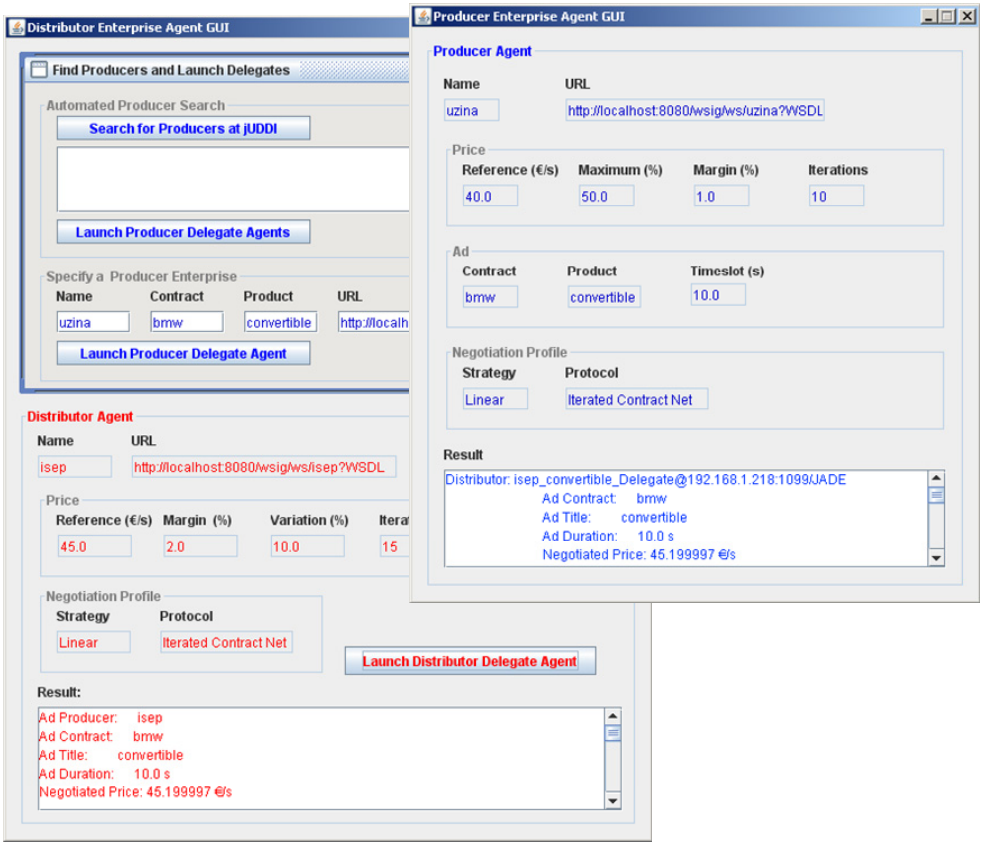

Figure 4: Distributor and producer agents GUI

For each enterprise agent launched, a new service interface is deployed at WSIG and the corresponding Web Service is registered at jUDDI. Figure 5 shows the list of Web Services registered at the WSIG after the creation of the marketplace agent *market*, *isep*, a distributor agent, and *uzina*, a producer agent, as well as the details of each Web Service interface.

The *uzina* and *isep* agents expose, respectively, the *GetProducerProfile* and *GetDistributorProfile* operations, used by the delegate agents to download their negotiation profile, and the *SetResults* operation, used by the delegate agents to report the negotiation outcomes. The *market* agent exposes the *RegisterAgent*

operation used to register the delegate agents in the marketplace. The different types of agents expose diverse operations due to the application of specific ontology mappers: producers use the *NegPubOntoMapProducer*, distributors the *NegPubOntoMapDistributor* and the market agent the *NegPubOntoMapMarket*. Finally, it presents for each agent the UDDI service key created as well as the link for the Web Service Description Language (WSDL) file automatically created by the WSIG.

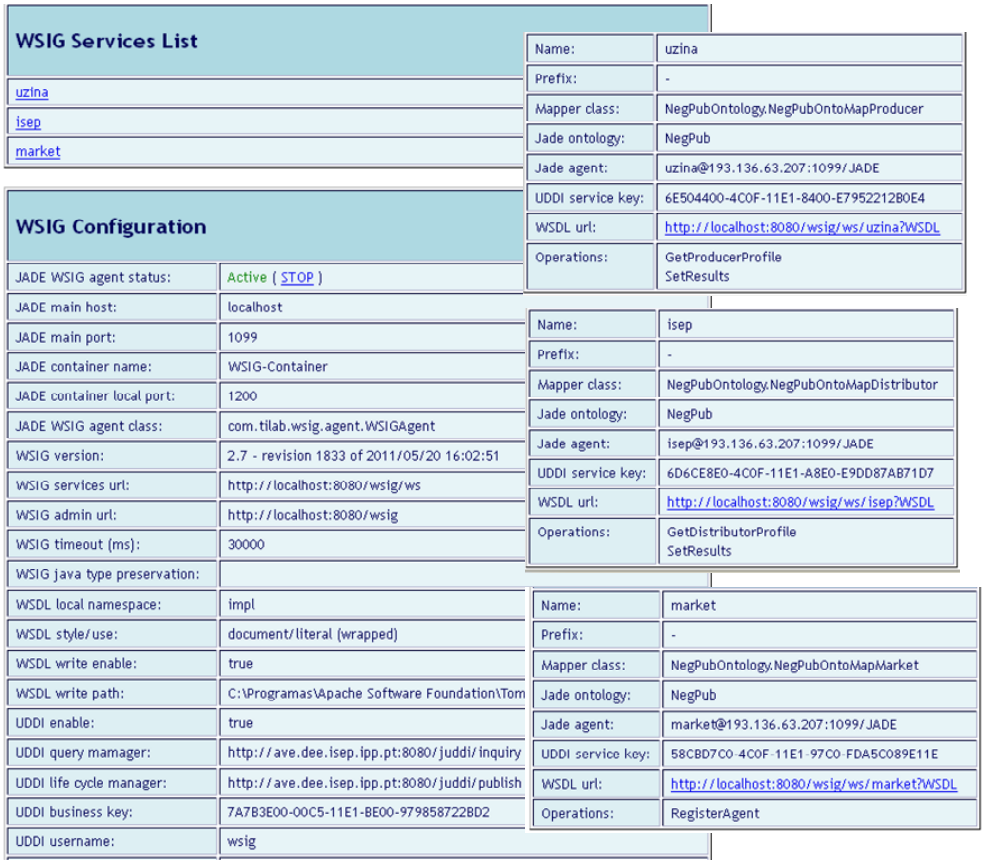

Figure 5: WSIG's service list

On the jUDDI side, it is possible to obtain the features of a business. Figure 6 shows the WSIG services registered at jUDDI: *isep*, *market* and *uzina*.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body xmlns="urn:uddi-org:api_v2">
    <serviceList generic="2.0" operator="jUDDI.org">
      <serviceInfos>
        <serviceInfo businessKey="7A7B3E00-00C5-11E1-BE00-979858722BD2"
                     serviceKey="6D6CE8E0-4C0F-11E1-A8E0-E9DD87AB71D7">
          <name>WSIG's businessService for isep</name>
        </serviceInfo>
        <serviceInfo businessKey="7A7B3E00-00C5-11E1-BE00-979858722BD2"
                     serviceKey="58CBD7C0-4C0F-11E1-97C0-FDA5C089E11E">
          <name>WSIG's businessService for market</name>
        </serviceInfo>
        <serviceInfo businessKey="7A7B3E00-00C5-11E1-BE00-979858722BD2"
                     serviceKey="6E504400-4C0F-11E1-8400-E7952212B0E4">
          <name>WSIG's businessService for uzina</name>
        </serviceInfo>
      </serviceInfos>
    </serviceList>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 6: WSIG's service descriptions at jUDDI

Figure 7 shows the *isep* agent service description stored to jUDDI.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body xmlns="urn:uddi-org:api_v2">
    <serviceDetail generic="2.0" operator="jUDDI.org">
      <businessService businessKey="7A7B3E00-00C5-11E1-BE00-979858722BD2"
                       serviceKey="6D6CE8E0-4C0F-11E1-A8E0-E9DD87AB71D7">
        <name>WSIG's businessService for isep</name>
        <bindingTemplates>
          <bindingTemplate bindingKey="6D77BE50-4C0F-11E1-BE50-EB165205F663"
                           serviceKey="6D6CE8E0-4C0F-11E1-A8E0-E9DD87AB71D7">
            <accessPoint URLType="http">http://localhost:8080/wsig/ws</accessPoint>
            <tModelInstanceDetails>
              <tModelInstanceInfo tModelKey="uuid:6D66F570-4C0F-11E1-B570-ED9EE68159CC"/>
            </tModelInstanceDetails>
          </bindingTemplate>
        </bindingTemplates>
        <categoryBag>
          <keyedReference keyName="fipaServiceName" keyValue="isep"
                          tModelKey="UUID:A035A07C-F362-44dd-8F95-E2B134BF43B4"/>
          <keyedReference keyName="GetDistributorProfile" keyValue="GetDistributorProfile"
                          tModelKey="UUID:A035A07C-F362-44dd-8F95-E2B134BF43B4"/>
          <keyedReference keyName="fipaServiceName" keyValue="isep"
                          tModelKey="UUID:A035A07C-F362-44dd-8F95-E2B134BF43B4"/>
          <keyedReference keyName="SetResults" keyValue="SetResults"
                          tModelKey="UUID:A035A07C-F362-44dd-8F95-E2B134BF43B4"/>
        </categoryBag>
      </businessService>
    </serviceDetail>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 7: WSIG's *isep* service description at jUDDI

After the creation of the enterprise agents, trading can start. The distributor agents query the UDDI registry for producers with ads matching the viewer profile and invite the discovered producers to participate in a dedicated negotiation. If these invitations are accepted and the involved agents support a common negotiation protocol, a collection of delegate agents are launched at the marketplace and the automated negotiation between delegates begins. In our case, *isep* and *uzina* launch two delegate agents in the marketplace: the *isep_convertible_Delegate* and the *uzina_bmw_convertible_Delegate* – see Figure 8.

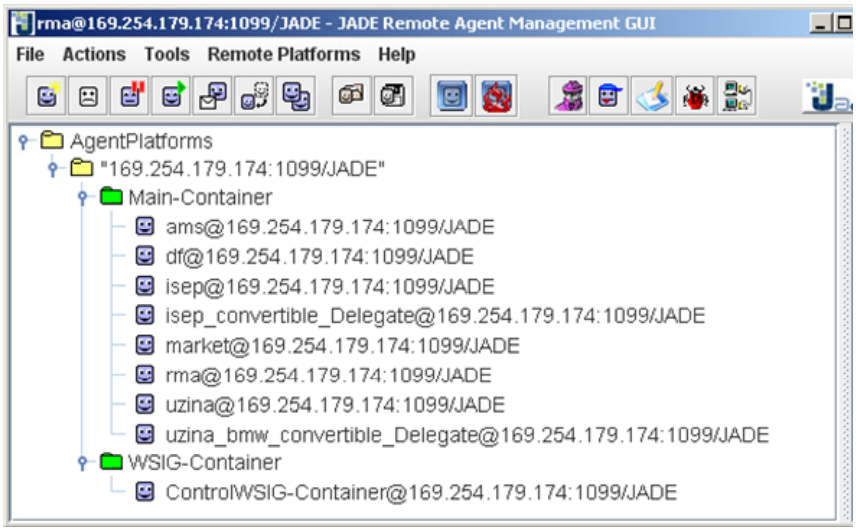

Figure 8: JADE platform GUI

In this example, distributor and producer delegates use the Iterated Contract Net (ICNET) negotiation protocol [14] and adopt a linear negotiation strategy regarding a single ad feature – price. Although the negotiation strategy is identical, the input parameters downloaded via the *GetDistributerProfile* and *GetProducerProfile* operations differ.

The negotiation follows the ICNET protocol, where the distributor delegate issues calls for proposals to all relevant producer delegates and collects the received proposals for a given number of iterations. In the end, the distributor delegate accepts the best proposal (higher price) and reports the result using the Web Service interface of the distributor agent, *i.e.*, invokes the *SetResults* operation of the distributor agent. This cycle is repeated until the whole interval timeslot is filled with an alignment of ads. At this point, the distributor delegate terminates execution. The producer delegates report their results back by invoking the *SetResults* operation of the corresponding producer agents and terminate.

Although the ICNET protocol adopts the Foundation for Intelligent Physical Agents (FIPA) Agent Communication Language (ACL) standard [15], it does not specify the actual content of a message. The content of the ICNET messages exchanged between delegate agents uses a dedicated multimedia negotiation

ontology that was also defined with the Protégé Ontology Editor and imported into the platform using JADE Bean Generator plug-in for Protégé. This ontology represents all relevant negotiation actions and features. The *negMultimediaComp* ontology defines a single action – *Negotiate* – and several negotiation features: *category*, *price*, *available_timeslot*, *contract* and *component*, a structure to hold multimedia objects composed of *type*, *category* and *timeslot*. Table 1 illustrates the use of this ontology during the ICNET negotiation by showing the content of three FIPA ACL standard messages: a call for proposals (CFP) issued by the *isep_convertible_Delegate*, a propose message send by the *uzina_bmw_convertible_Delegate* and a final accept-proposal of the *isep_convertible_Delegate*.

Table 1: Content of ICNET messages

Performative	**Content**
CFP	(*Negotiate* :*category* convertible :*price* 45.0 :*available_timeslot* 10.0)
PROPOSE	(*Negotiate* :*contract* bmw :*price* 45.199997 :*available_timeslot* 0.0 :*component* (*MultimediaComponent* :*type* ad :*category* convertible :*timeslot* 10.0))
ACCEPT-PROPOSAL	(*Negotiate* :*category* convertible :*price* 45.199997 :*available_timeslot* 0.0)

Finally, the distributor delegates have a GUI where the outcomes of the negotiation rounds are displayed – see Figure 9.

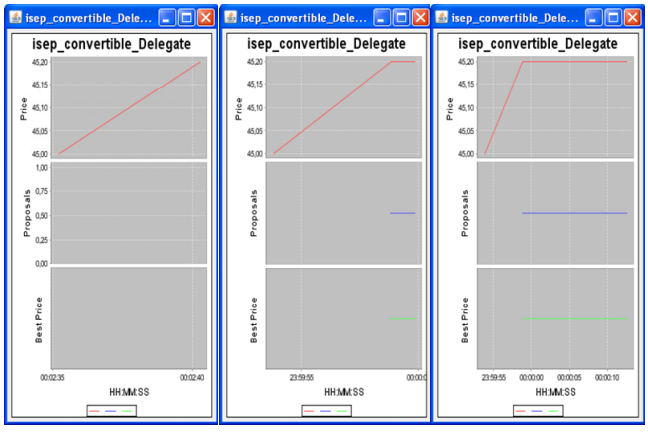

Figure 9: Distributor agent *isep* GUI

4 Conclusions

In this paper we situate this research work, describe an implementation scenario, the layered platform architecture and, then, focus on the marketplace layer development and on its interaction with the enterprise layer.

4.1 Achievements

A brokerage platform GUI has been implemented for specifying and launching distributor and producer agents. These agents offer Web Service interfaces to interact both with user interface (interface layer) and market delegate (marketplace layer) agents. Furthermore, they register their Web Service specifications on a UDDI service registry to allow the discovery and consumption of the exposed services by others.

Distributor agents wishing to create a personalised ad alignment for an upcoming viewer interval, query the UDDI to discover and invite relevant producers to the market. Upon success, a distributor delegate and one or more producer delegates are created at the marketplace and the actual negotiation occurs according to the negotiation profiles downloaded from the enterprise agents. Once a negotiation finishes, the delegates report back their negotiation outcomes and terminate.

Two ontologies were created to support the negotiation process: the first is used between enterprise and delegate agents, *i.e.*, in the inter-layer communication, and the second between delegate agents within the marketplace layer.

The setup of this test-bed has involved a relevant effort in the selection, deployment and use of several different technologies exclusively supported by open source API and tools.

4.2 Future Work

The interface layer between the real world producers and distributors and the platform will continue to be developed. The enterprise interface agents will interact with the enterprise layer through Web Service interfaces, allowing enterprises to reconfigure their enterprise agents, define new inputs and collect the negotiation results.

The enterprise layer agents will expose additional Web Service interfaces to support the interaction with the interface layer.

Support to other negotiation protocols and strategies will be added to the marketplace layer.

To reinforce the availability and scalability of the platform, the migration to the cloud computing paradigm will be attempted in an effort to offer the platform as a Software-as-a-Service (SaaS) component to the involved parties, *i.e.*, the real world enterprises.

Finally, extensive testing with appropriate datasets will be carried out.

References

[1] Malheiro, B., Foss, J., Burguillo, J. C., Peleteiro, A. and Mikic, F. A. (2011), "Dynamic Personalisation of Media Content", Proceedings of the Sixth International Workshop on Semantic Media Adaptation and Personalization (SMAP 2011), ISBN 978-0-7695-4524-0, pp 21-26, Vigo, Spain.

[2] Malheiro, B. and Foss, J. (2010), "A Proposal for Media Component Brokerage", Proceedings of the Fourth International European Conference on the Use of Modern Information and Communication Technologies (ECUMICT 2010), Ed. Lieven de Strycker, Nevelland v.z.w., ISBN 978-9-08-082555-5, pp 389-403, Gent, Belgium.

[3] Ma, Y., Xia Li, H. and Sun, P. (2007), "A lightweight agent fabric for service autonomy", Proceedings of the Conference on Service-oriented Computing: Agents, Semantics, and Engineering (SOCASE'07), Huang et al. (Eds.), LNCS 4504, Springer-Verlag, pp. 63-77.

[4] Cao, M., Chi, R. and Liu, Y. (2009), "Developing a Multi-Agent Automated Negotiation Service Based on Service-Oriented Architecture", Service Science, Vol. 1, No. 1, pp 31-42.

[5] Bellifemine, F., Caire, G. and Greenwood, D. (2007), "Developing Multi-Agent Systems with JADE", Wiley.

[6] Telecom Italia. JADE (2012). Available at http://jade.tilab.com/ [Accessed in January 2012].

[7] JADE Board (2011), "JADE Web Services Integration Gateway (WSIG) Guide", Telecom Italia, Tech. Rep.

[8] Scagliotti, E. and Caire, G. (2010), Telecom Italia, "Web Services Dynamic Client (WSDC) Guide", Telecom Italia, Tech. Rep.

[9] The Apache Software Foundation (2012). Apache jUDDI. Available at http://juddi.apache.org/ [Accessed in January 2012].

[10] The Apache Software Foundation (2012). Apache Tomcat. Available at http://tomcat.apache.org/ [Accessed in January 2012].

[11] The Apache Software Foundation (2012). Apache Axis2/Java. Available at http://axis.apache.org/axis2/java/core/ [Accessed in January 2012].

[12] Oracle Corporation. MySQL (2012). Available at http://www.mysql.com/ [Accessed in January 2012].

[13] Stanford Center for Biomedical Informatics Research (2012). Protégé. Available at http://onlinelibrary.wiley.com/book/10.1002/9780470058411 [Accessed in January 2012].

[14] FIPA (2012). Iterated Contract Net Specification. Available at http://www.fipa.org/specs/fipa00030/ [Accessed in January 2012].

[15] FIPA (2012). Agent Communication Language. Available at http://www.fipa.org/repository/aclspecs.html [Accessed in January 2012].