Arquitectura multiagente para un sistema e-learning centrado en la enseñanza de idiomas

Liliana E. Machuca V.
Universidad del Valle
Colombia
liliana_emv@hotmail.com

Paola J. Rodríguez C.
Universidad del Valle
Colombia
paolajr@eisc.univalle.edu.co

ABSTRACT

In this paper we present the description of a multiagent architecture for an e-learning system (SE-MAS) focused on language teaching. Its purpose is to provide intelligence and adaptivity, taking into account the needs of students and their performance in the platform.

RESUMEN

Este artículo presenta la descripción de una arquitectura multiagente para un Sistema e-learning (SE-MAS) orientado a la enseñanza de idiomas. Su finalidad, es proveer inteligencia y adaptatividad, teniendo en cuenta las necesidades de los estudiantes y su desenvolvimiento en la plataforma.

KEYWORDS

Arquitectura de Software, e-learning , Sistemas Multiagente, enseñanza, idiomas.

1. INTRODUCCIÓN

En la actualidad, los sistemas e-learning se han orientado principalmente, a la gestión del material asociado a un curso y su fácil navegación, dejando de lado aspectos importantes para los procesos de enseñanza aprendizaje, como las características de los estudiantes y las necesidades específicas de las diferentes áreas del saber. Por ello, en la enseñanza y el aprendizaje de lenguas a través de internet se observan que la mayoría de instituciones educativas opta por sistemas multipropósito comerciales (p.e. Blackboard, WebCT) o de uso libre (p.e. Moodle, A-Tutor, Claroline), combinados a veces con otras herramientas para ofrecer cursos de lenguas en línea un poco más acoplados a las necesidades de este tipo de saberes. Ejemplo de esto es ILLP (Independant Language Learning Program) de la Universidad de Manchester [1] o, incluso, con materiales multimedia off-line como el caso de la Universitá Oberta de Catalunya. En esta misma línea, el proyecto COVCELL (Cohort-Oriented Virtual Campus for Effective Language Learning), financiado por la Unión Europea, busca crear y adaptar un entorno virtual para el aprendizaje de lenguas dentro de Moodle; Intentando incorporar herramientas para el desarrollo de tareas de aprendizaje colaborativas, como chat, video chat, pizarra compartida, audio/video conferencia, video uno a uno y grabación de audio para evaluación [2].

En Colombia, se registran algunas experiencias relativamente nuevas de integración de este tipo de sistemas para la formación en idiomas. La plataforma de mayor uso es Moodle como se ve en la Escuela de Idiomas de la Universidad de Antioquia, en el Programa Alex de la Universidad Nacional de Colombia y en el Departamento de Idiomas de la Universidad Tecnológica del Chocó. En la Universidad del Valle también se han implementando algunos cursos de ESP (English for Specific Purposes) y del programa de desarrollo profesional docente en inglés de la Escuela de Ciencias del Lenguaje en el Campus Virtual montado en Moodle [3].

En cuanto a entornos virtuales de uso particular, se pueden mencionar: la plataforma UdeC English Online de la Universidad de Concepción en Chile [4], cuyo diseño e implementación se enmarcan en una perspectiva interaccionista de la adquisición de segundas lenguas y que ofrece gran variedad de herramientas y de recursos; iWill (Intelligent Web-based Interactive Language Learning), un ambiente web para el aprendizaje de idiomas con acceso a diversos recursos de interacción (foro, club de lectura, análisis de concordancias textuales); el Campus Virtual FLE de la Universidad de León en España, que integra también Moodle para los cursos en línea. Muchos de estos entornos, llamados a menudo "campus virtuales" pues presentan información institucional, ofrecen páginas web con información y contenidos (documentos, actividades y ejercicios), espacios de interacción como blogs, etc; sin embargo, se trata de diversos sistemas de e-learning que no ofrecen la posibilidad de inscribirse en cursos para disponer del acompañamiento de un tutor o de un profesor.

Aunque existen herramientas generales de gestión de cursos en línea, que ofrecen variados sistemas de comunicación y de estudio, la mayoría carece de recursos que respondan a las necesidades específicas de la enseñanza y el aprendizaje de idiomas, como son: altos niveles de interactividad y socialización, acompañamiento constante y evaluación inmediata del desempeño lingüístico de los aprendices. El seguimiento y la evaluación dependen casi exclusivamente del profesor o de pares, que no siempre están presentes en el proceso de realización de las tareas lingüísticas.

De acuerdo al contexto anterior, la definición de una arquitectura basada en agentes de software, que modele la solución a los inconvenientes presentados en los sistemas de e-learning para la enseñanza de idiomas, debe resolver las siguientes preguntas:

Machuca, L., Rodríguez, P. (2009). Arquitectura Multiagente para un Sistema e-learning centrado en la enseñanza de idiomas. En J. Sánchez (Ed.): Nuevas Ideas en Informática Educativa, Volumen 5, pp. 83 – 91, Santiago de Chile.

- ¿Cuál es el tipo de adaptatividad necesaria para un sistema de e-learning centrado en la enseñanza de idiomas?
- ¿Cuál debe ser la arquitectura interna de los agentes software que den soporte al sistema e-learning?
- ¿Cómo deben ser las interfaces de percepción y actuación de los agentes definidos para el sistema multiagente?

La respuesta a cada uno de ests interrogantes se resuelve en las secciones 2, 3 y 4 respectivamente.

2. MÓDELO DE ADAPTATIVIDAD

Para la definición del tipo de adaptatividad a incorporar, se parte del modelo pedagógico propuesto en [5], tomando la producción textual como eje primario del sistema a desarrollar. Adicionalmente, se estudiaron las taxonomías de tecnologías hipermedia adaptativa descritas en [6] y presumidas en la Figura 1.

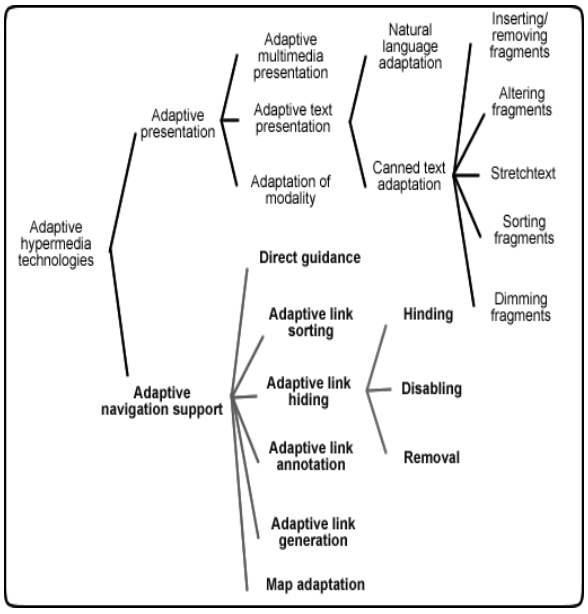

Figura 1. ***Taxonomía de tecnologías hipermedia adaptativas***

2.1. Técnicas de adaptatividad seleccionadas

Después de un estudio concienzudo de la anterior taxonomía y con la asesoría de las autoras de modelo pedagógico seleccionado se optó por las técnicas Adaptive link annotation y Direct Guidance.

Con Adaptive link annotation, se le permite al estudiante, diferenciar su nivel de conocimiento, es decir, lo que debe saber, lo que está aprendiendo y lo que desconoce. Para esto, un agente software monitorea y actualiza la interfaz del usuario, identificando cada nivel con un color, el cual, resalta el momento que vive el estudiante en cada producción textual realizada. Este tipo de adaptatividad apoya el enfoque constructivista que enmarca el modelo pedagógico, al incitar al estudiante en la construcción de su aprendizaje desde el plano informativo de su progreso en el curso. Adicionalmente, promueve un acompañamiento tácito ya que los colores poseen un significado valioso, que orienta el progreso del estudiante e indica nivel de adelanto o demora con respecto al tiempo de desarrollo del curso. Consecuentemente, Adaptive link annotation, permite ahorrar tiempo y esfuerzo adicional del docente, ya que es directamente el sistema multiagente, el que se encarga de determinar el estadio del estudiante.

Direct Guidance, busca guiar al estudiante hacia el siguiente ejercicio o problema a resolver de acuerdo a la calidad de la solución que ha dado a un ejercicio o problema previo. Para llevar a cabo esta técnica, el profesor debe crear el ejercicio complementario y definir rangos de valoración que habilitaran el ejercicio complementario. El sistema multiagente monitorea estos rangos para activar dichos ejercicios e informa al docente la efectividad de los mismos, empleando una valoración del usuario (sistema de calificación por parte del estudiante) y un monitoreo automático apoyado en redes bayesianas que permite relacionar un peso dado a un ejercicio con la cantidad de estudiantes que lo resuelven satisfactoriamente. En la figura 2, se muestra el esquema general de adaptación que deberá soportar la arquitectura a proponer:

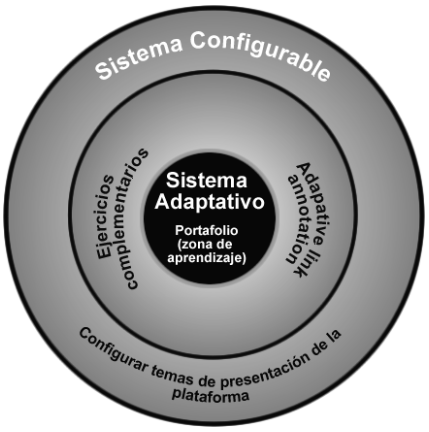

Figura 2. ***Esquema general del Modelo de Adaptatividad***

3. ARQUITECTURA PROPUESTA

A continuación se describe el conjunto de vistas que conforman la arquitectura propuesta.

3.1. Vista conceptual

En el esquema que se presenta a continuación, se identifican las secciones o módulos en los que se ha organizado el sistema. Cada sección se ha agrupado teniendo en cuenta los patrones de diseño alta cohesión y bajo acoplamiento, esto ha permitido que se generen las áreas que se visualizan en la imagen de acuerdo a las funcionalidades comunes presentadas por ellas.

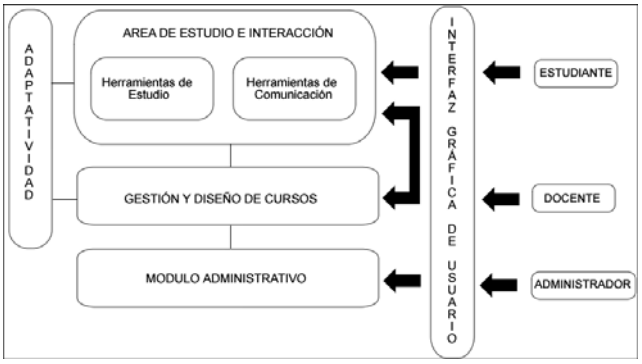

Figura 3. ***Estructura Conceptual del Sistema***

De la anterior estructura se identifican los paquetes de: Administración, Curso, Herramientas, Adaptatividad y Interfaz gráfica:

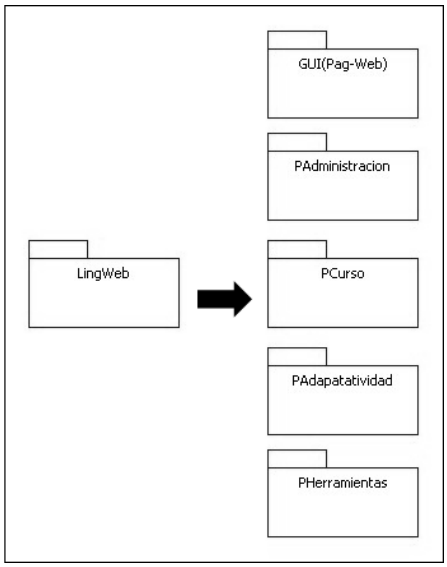

Figura 4. ***Vista conceptual***

El paquete *PAdministración* cubre la gestión de cursos, la gestión de usuarios y la gestión de la configuración y administración de la plataforma. El paquete *PCurso* está relacionado con la gestión de cursos, matrícula, gestión de unidades, secuencias, actividades y ejercicios, así como la gestión de material. El paquete *PHerramientas* hace alusión a la gestión de las herramientas de estudio y comunicación de la plataforma. El paquete *PAdaptatividad* encierra todo lo relacionado con los agentes de software diseñados para proveer adaptatividad en la navegación y acompañamiento en la plataforma. Finalmente, el paquete GUI(Pag-Web) contiene las interfaces de usuario y lógica de interfaz (debido al uso de Tecnologías AJAX) del sistema.

3.2. Vista Lógica

La figura 5, presenta el sistema distribuido en capas, permitiendo visualizar los patrones de diseño controlador y fachada para permitir el control de los eventos de la interfaz del usuario a la lógica, y el acceso de la lógica a los medios de almacenamiento respectivamente.

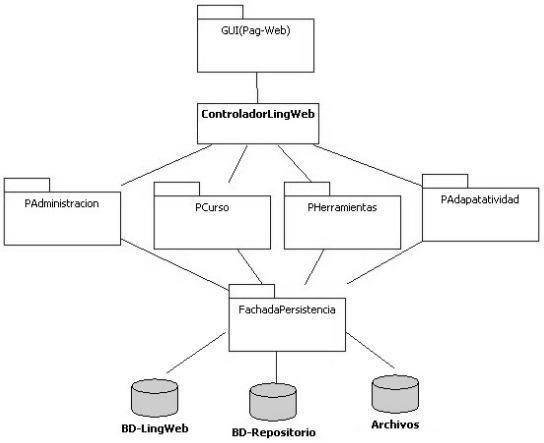

Figura 5. ***Estructura general por capas***

Profundizando en la arquitectura multiagente que está asociada con el paquete de adaptatividad, a continuación se presentan los modelos más representativos soportados en la metodología Ingenias y el lenguaje de Modelado de Agentes (AML) [7][8].

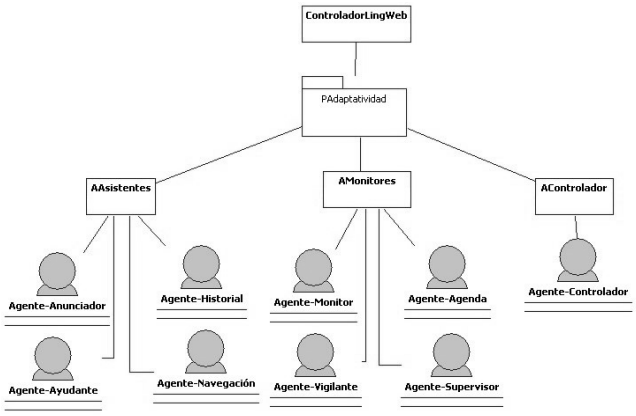

Figura 6. ***Estructura del Paquete Adaptatividad***

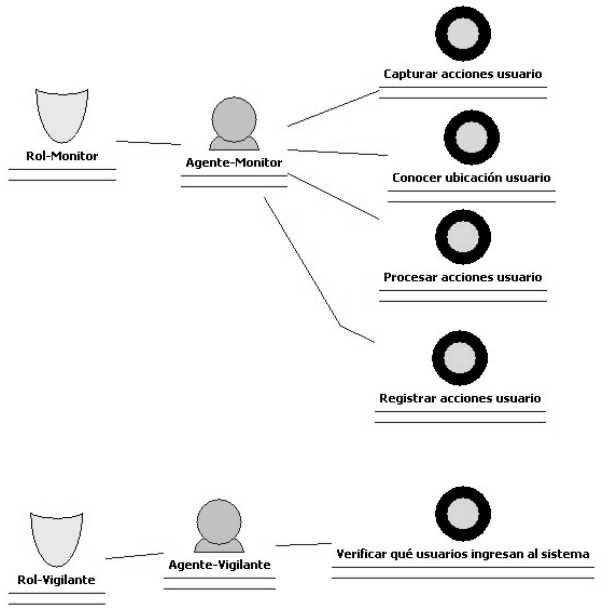

Figura 7. ***Detalle del Modelo de Agentes Monitor y Vigilante***

Para dar claridad al paquete de adaptatividad, la Tabla 1, presenta los agentes definidos para el sistema y sus responsabilidades.

Agente	Responsabilidad	Descripción
Monitor	• Capturar las acciones del usuario • Conocer ubicación del usuario • Procesar acciones de usuario • Registrar acciones usuario	Monitorea los movimientos de los usuarios para saber qué está haciendo en determinado momento, en qué lugar se encuentra y de acuerdo a ello poder dar soporte, ya sea de ayuda o a la navegación dentro de un curso.
Vigilante	• Verificar qué usuarios ingresan al sistema	Encargado de informar al agente controlador sobre qué usuarios están ingresando al sistema llevando una lista de los mismos
Anunciador	• Enviar mensaje de llegada de usuario • Enviar anuncio	Está al tanto de los mensajes que se deben enviar a los usuarios, ya sea cuando un usuario ha ingresado al sistema o cuando tenga pendientes por hacer
Historial	• Registrar contenido a estudiar • Registrar Unidades, secuencias, actividades y ejercicios estudiados • Registrar Unidades, secuencias, actividades y ejercicios que se están estudiando • Reportar estado del historial	Su función es llevar el registro del progreso del estudiante. Este le ayudará al agente de navegación para que pueda colorear cada área con el correspondiente estado en el que se encuentre el usuario
Agenda	• Monitorear tareas programadas	Está constantemente revisando las actividades pendientes que pueda tener un usuario estudiante. Al encontrar pendientes, le informa al agente anuncios para que éste último le informe de dichos eventos al usuario.
Supervisor	• Monitorear porcentajes de ejercicios • Monitorear que actividades y ejercicios estén completos antes de la fecha de inicio	Tienen como objetivo controlar que el docente asigne los porcentajes a las pruebas correctamente, como también indicarle que tiene alguna actividad incompleta que está próxima a publicar.
Ayudante	• Capturar preguntas • Analizar preguntas • Responder preguntas • Procesar acciones de usuario • Desplegar sugerencia • Registrar tema de ayuda	Es el encargado de dar el acompañamiento al usuario en el desenvolvimiento de la plataforma. Estará disponible para que se le pregunte temas de ayuda sobre el uso de la plataforma, así como para dar sugerencias del desenvolvimiento en algún lugar específico.
Controlador	• Validar usuarios por curso	Con este agente se busca identificar los cursos a los que pertenece el usuario que va ingresando.

Navegación	• Consultar historial de aprendizaje • Asignar colores a los niveles de aprendizaje • Monitorear ejercicios calificados • Comparar rangos de calificación • Sugerir ejercicio complementario	Tiene como función dar soporte a la navegación de un estudiante. Tiene dos roles, uno que se orienta al progreso del estudiante y otro relacionado con el ejercicio complementario. Su utilidad se manifiesta en el acompañamiento que se le da al estudiante al indicarle el estado de su progreso de aprendizaje a través de colores diferentes. También le sugiere realizar un ejercicio complementario cuando el resultado de calificación no es favorable con su rendimiento, esta opción depende si el profesor ha definido un ejercicio complementario

Tabla 1. ***Agentes y sus responsabilidades***

Igualmente, se definieron algunas interfaces de comunicación entre los paquetes que conforman la arquitectura, como se muestra en la figura 8.

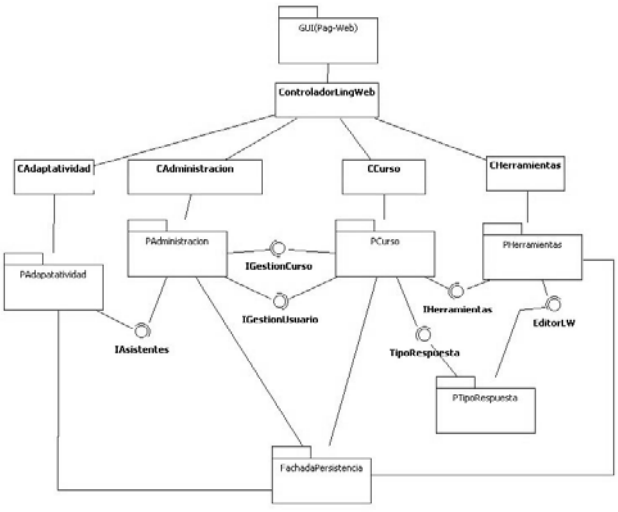

Figura 8. ***Interfaces vistas desde la estructura general***

Las interfaces de comunicación definidas son: IAsistentes para la creación de los diferentes agentes asistentes que son requeridos cuando un usuario inicia sesión; IGestionCurso define los métodos que permiten la creación y edición de un curso; IGestionUsuario se establece para proveer los diferentes métodos de acceso a la gestión de la información relacionada con los usuarios del sistema; IHerramientas definida para la creación de los diferentes tipos de herramientas a utilizar, ya sea en la creación de un curso, de un ejercicio o cuando un estudiante es registrado en un curso; IEditorLW definida para tener acceso al editor especializado; ITipoRespuesta establecida para interconectar con el subsistema relacionado con la producción textual que define los tipos de respuesta que pueden ser trabajados en la plataforma.

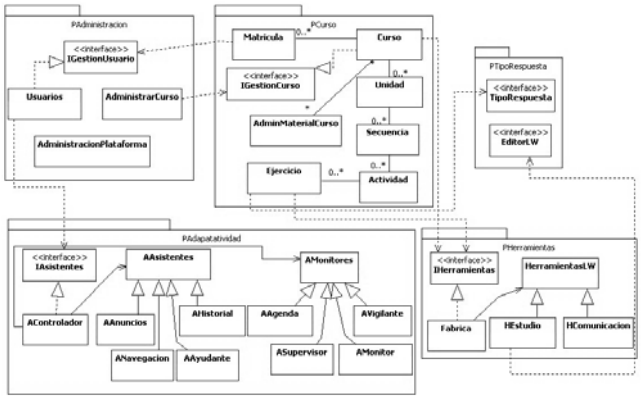

Figura 9. ***Vista de Interfaces entre paquetes de la lógica de programación***

3.3. Vista Física

Dado, que el sistema es de tipo e-learning y se plantea una alta demanda de cursos soportados, se propone trabajar con un balanceador de carga, dos servidores web, dos servidores de aplicaciones y dos servidores de bases de datos, a fin de lograr un mejor desempeño y tolerancia a fallos. Ver figura 10.

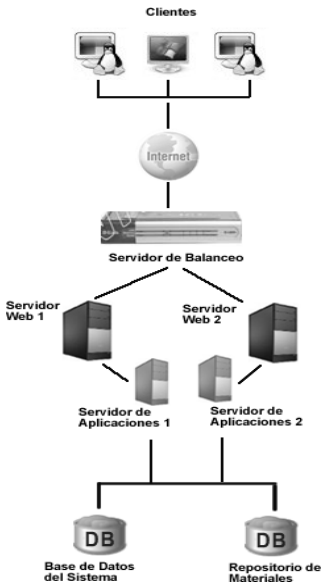

Figura 10. Vista Física del Sistema

El balanceador de carga, distribuirá las peticiones a los dos servidores web dependiendo de unos criterios ya establecidos

en su configuración. Los servidores web a su vez comunicaran las peticiones a su servidor de aplicaciones asociado. Los servidores de aplicaciones enviaran la transacción deseada a uno de los dos servidores de bases de datos dependiendo de si es la base de datos del Sistema o la base de datos del Repositorio de Materiales.

4. ALGUNOS ASPECTOS DE IMPLEMENTACIÓN

A fin de mostrar cómo el diseño anterior es plasmado en el producto software, se documenta la implementación del prototipo funcional de uno de los agentes planteados en la arquitectura. Con esto se busca visualizar en una pequeña dimensión, cómo se llevaría a cabo la integración entre los diferentes componentes involucrados en el sistema. Para tal fin, se seleccionó al agente anuncios, dado que éste se relaciona con otros agentes, la base de datos y con la interfaz Web.

La implementación de los agentes se estructura por paquetes organizados de acuerdo a su funcionalidad, incluso, se ha creado un paquete denominado *Comun*, al cual pertenecen las clases base utilizadas por todo el sistema multiagente. Estos paquetes son:

- SEMAS.Agenda: Contiene las clases relacionadas con el agente Agenda. La clase para la creación del agente y sus comportamientos.
- SEMAS.Anuncios: Es el paquete que contiene las clases y comportamientos del agente Anuncios
- SEMAS.Comun: Contiene las clases que son utilizadas por los demás agentes, como es, el acceso a la BD, y el formato para el envío de los mensajes entre agentes.
- SEMAS.Controlador: Es el paquete que contiene las clases y comportamientos del agente controlador
- SEMAS.Monitor: Es el paquete que contiene las clases y comportamientos del agente Monitor
- SEMAS.Supervisor: Es el paquete que contiene las clases y comportamientos del agente Supervisor

Los agentes que le envían mensajes al agente anuncios, toman la información de un archivo de texto[1], que contiene datos como curso, ejercicios, estudiantes, fechas, etc. que son utilizados por el agente para construir el mensaje. Una vez estos son creados, el agente ya sea Agenda, Supervisor, o Monitor envían el mensaje al agente anuncios, éste los espera y procesa para almacenarlos en la Base de Datos y hacerlos disponibles para la interfaz a través de un archivo XML.

Cuando el Agente Anuncios es ejecutado, se registra en el DF (Facilitador de Directorio de Jade[2]) y adiciona un primer comportamiento de inicialización del agente, en el que se revisa si es necesario eliminar mensajes que ya por su tiempo no deban ser publicados y así poder actualizar el archivo XML para que la interfaz muestre los mensajes. Este comportamiento, a su vez, adiciona el comportamiento de recibir eventos, el cual permanece escuchando su llegan mensajes de otros agentes para ser registrados en la base de datos y generar el archivo *anuncios.xml*.

El código de los agentes (Agenda, Controlador, Monitor y Supervisor) que se han implementado, se limita a su creación y a la adición de los comportamientos requeridos por ellos para poder enviar un mensaje al Agente Anuncios. Esta implementación no se preocupa por saber el cómo han ocurrido los eventos que le informan al agente, sólo toman los datos de un archivo para crear el mensaje.

El prototipo incluyó solo aquellos agentes que están relacionados con el Agente Anuncios, esto se puede visualizar en el modelo de servicios del sistema multiagente. ver figura 11.

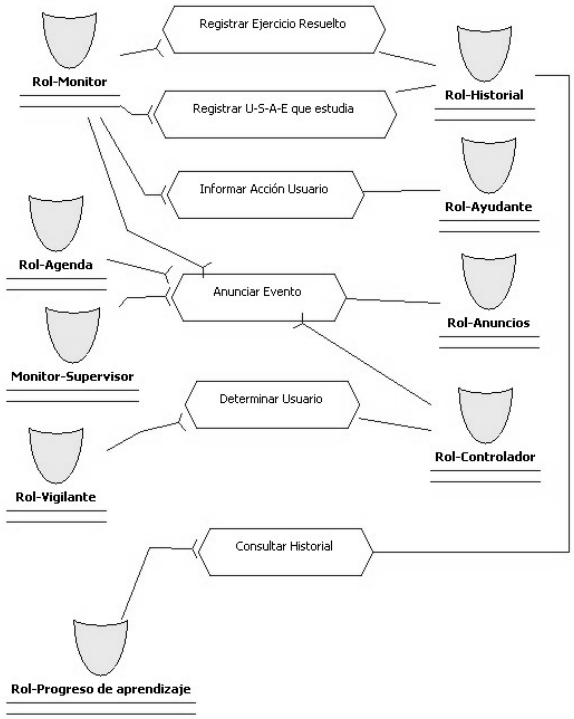

Figura 11. ***Modelo de Servicios del Sistema Multiagente***

La interfaz gráfica arranca los agentes para que ellos se ejecuten desde la Web. Esto se hace, una vez se carga en el navegador la página Web que contiene los anuncios.
El agente anuncios se comunica con la interfaz gráfica por medio de un archivo xml, el cual es creado y/o actualizado cada vez que un mensaje nuevo es reportado.
La página web donde se reflejan los anuncios, ha sido desarrollada utilizando Ajax. Con ésta tecnología se pretende crear una zona en la que a través de pestañas el usuario pueda ver clasificados sus mensajes. Los mensajes se han

[1]Cuando el sistema esté totalmente funcional, esta información será tomada de la Base de Datos del Sistema, o del monitoreo realizado a las interacciones del usuario.
[2] Java Agent Development Framework.

organizado en *usuarios* (muestra los usuarios que están conectados en ese momento en el curso), *pendientes* (tanto para docentes como estudiantes, en esta zona se visualizan los mensajes que tienen que ver con las actividades y ejercicios pendientes por realizar) y *general* (en ésta se muestran aquellos anuncios que le indican al usuarios si tiene un ejercicio calificado o resuelto). Cada vez que el usuario selecciona alguna de las pestañas, la página a través de ajax visualiza los mensajes que el archivo xml contiene y que son propios para esa sección.

4.1. Interfaces de comunicación genéricas entre cada una de las tecnologías

El agente Anuncios, utiliza el paso de mensajes asíncrono de JADE y el Lenguaje de Comunicación de Agentes (ACL) para la creación de dichos mensajes; de esta forma logra comunicarse con los demás agentes [9] [10].

Es importante resaltar que el agente Anuncios utiliza el comportamiento concurrente BRecibirEvento, que está a la espera de cualquier mensaje enviado por los demás agentes y una vez recibe los mensajes, éstos son procesados a través del método procesarEvento que se le definió al agente. Se ha decidido implementar el comportamiento con un CyclicBehaviour, ya que permite que la acción de recibir eventos de otros agentes se ejecute varias veces. Esta clase de comportamiento, se mantiene activa tanto tiempo como esté activo el agente.

A continuación se presenta un fragmento del código del comportamiento:

```
public class BRecibirEvento extends CyclicBehaviour
{
 Evento contenido;
 private int step = 0;
 MessageTemplate mt1;
 ACLMessage mesg;
 int tiempoBloqueo = 10000;
 /**
 * Ejecuta el comportamiento del agente
 */
 public void action() {
//Define la plantilla del tipo de mensajes que
puede recibir - INFORM -

mt1=MessageTemplate.MatchPerformative(ACLMessage.IN
FORM);
 mesg = myAgent.receive(mt1);
 if (mesg != null) {
 //Obteniendo el contenido del mensaje
 try {
 contenido = (Evento) mesg.getContentObject();
 } catch (jade.lang.acl.UnreadableException ex) {
 System.out.println("Excepción al obtener el
contenido, agente " + myAgent.getAID().getName());
 }
//Solicitando el procesamiento del evento recibido
 System.out.println("AGENTE ANUNCIOS RECIBE UN
EVENTO DE TIPO " + contenido.obtenerTipoEvento());
 ((AAnuncios)
myAgent).procesarEvento(contenido.obtenerTipoEvento
(), contenido);
 mesg = null;
       }
    }
}
```

Por otra parte, el agente Anuncios y los otros agentes cuando lo requieran, se pueden comunicar con la base de datos a través de la fachada BD que se encuentra en el paquete Comun. En ella se provee los diferentes métodos que permiten la conexión, consulta, actualización en la Base de Datos. Para efectos del prototipo no se ha creado toda la base de datos, solo la tabla que se requiere para almacenar los mensajes, es decir la tabla anuncios.

Parte de la comunicación del agente anuncios con la Base de Datos se muestra a continuación procesando el tipo de evento Tareas Pendientes:

```
else if (tipo.equals("TareasPendientes")) {
System.out.println("Agente Anuncios se encuentra
procesando el evento" + tipo);
//Se crea la conexión
bd = new BD();
bd.conexion();
//Verificando si el anuncio existe en la BD
String anuncioExiste = "SELECT tipo FROM anuncios
WHERE anuncio='Tiene pendiente por realizar " +
contenido.obtenerNombreTarea() + " Para la fecha "
+ contenido.obtenerFecha() + "'";
if (verificarAnuncios(anuncioExiste, "tipo") ==
false) {
System.out.println("Tiene pendiente por realizar "+
contenido.obtenerNombreTarea()+"Para la fecha " +
contenido.obtenerFecha());
System.out.println("Enviando tarea pendiente para
ser almacenada en BD");
 //Construyendo la consulta para almacenar el
registro en la BD
Date fechaActual = new Date();
SimpleDateFormat formato = new
SimpleDateFormat("yyyy.MM.dd");
String cadenaFecha = formato.format(fechaActual);
String consulta = "INSERT INTO anuncios
(anuncio,fecha,tipo,curso) VALUES ('Tiene pendiente
por realizar " +
contenido.obtenerNombreTarea() + " Para la fecha "
+ contenido.obtenerFecha() + "','" + cadenaFecha +
"','TareasPendientes','" + contenido.obtenerCurso()
+ "')";
bd.update(consulta);
}
//Se cierra la conexión
bd.cerrar();
//aqui debe generar el xml
archivo.crear();
}
```

La comunicación entre el agente Anuncios y la página web que visualiza los anuncios, se realiza a través de un archivo XML llamado anuncios.xml. Para la creación y lectura del archivo por parte del agente Anuncios y de la página web respectivamente, se utilizó JDOM un API que realiza dichas funciones con archivos XML. Se ha escogido JDOM por su compatibilidad con Java y por su facilidad para la manipulación de este tipo de archivos.

Al igual que con la Base de Datos se utilizó una fachada para acceder al archivo anuncios.xml con el fin de mantener la integridad de las capas. Esta fachada ha sido llamada CrearXML y provee los métodos crear y leer para la manipulación del archivo por parte del agente anuncios y la

página web. En el fragmento del código anterior se puede visualizar el llamado que hace el agente anuncios al método crear() (archivo.crear();) una vez se ha recibido un mensaje.

La página web que contiene los anuncios utiliza JavaScript para hacer la presentación de los mismos en forma de pestañas o tabs como lo muestra la figura 12.

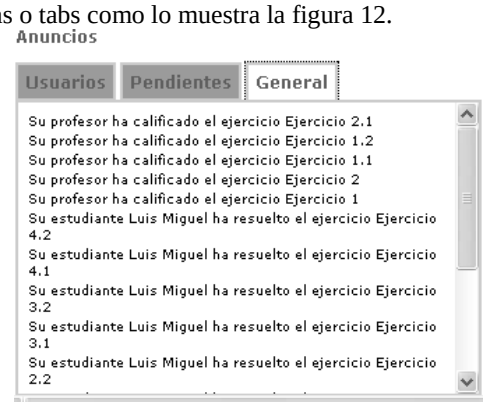

Figura 12. Fragmente de la GUI Anuncios

El código JavaScript utilizado solicita por cada pestaña un archivo con la información a mostrar en cada una de ellas. Por consiguiente se implementaron tres (3) archivos que hicieran la lectura del archivo anuncios.xml para procesar los anuncios de acuerdo a cada pestaña.

El código que desde una página web que hace uso de la fachada CrearXML es el siguiente:

```
<table width="100%" border="0" align="center">
 <tr>
 <%
 CrearXML anuncios = new CrearXML();
 ArrayList lista = anuncios.leer("general");
 int elementos = lista.size();
 int i = 0;
 %>
 <% for (i = 0; i < elementos; i++) {%>
 <td style="font-family: Verdana, Arial, Helvetica,
sans-serif; font-size: 9px; COLOR: black;">
 <%
 out.println(lista.get(i));
 %>
 </td> </tr>
 <%}%>
</table>
```

Adicionalmetne, en la figura 13 se muestra la comunicación entre las diferentes tecnologías que se han descrito anteriormente.

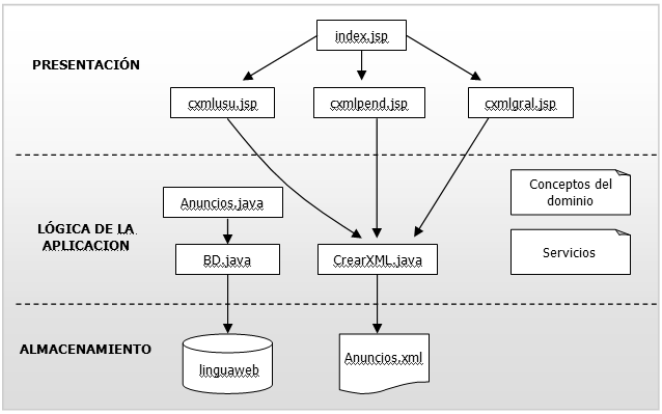

Figura 13 Interfaces de comunicación

5. CONCLUSIONES Y TRABAJO FUTURO

Este trabajo contribuye en el campo de los sistemas e-learning, muy específicamente en el área de idiomas, ya que se propone una arquitectura novedosa en la cual se contemplan funcionalidades que no son tenidas en cuenta en las plataformas de aprendizaje virtual de mayor uso.

La arquitectura SE-MAS establece aquellas actividades que dentro del sistema e-learning son las más convenientes para ser implementadas con agentes de software, como son: Anuncios a los usuarios, dar soporte y acompañamiento a los usuarios durante su desempeño en el sistema. Permitiendo con ello una asistencia en línea aún cuando el docente no esté presente.

Como trabajo futuro se espera definir un modelo de asistente virtual que permita guiar el proceso de producción textual, para ello se pretende diseñar un editor especial el cual incluye los elementos necesario para crear párrafos y textos, de acuerdo al módelo pedagógico tomando como base y en el cual actuaría el agente de asistencia

AGRADECIMIENTOS

Los autores agradecen a Colciencias, y las Escuelas de Ingeniería de Sistemas y Computación, y Ciencias del Lenguaje de la Universidad del Valle.

REFERENCIAS

[1] Lewis, Calico Journal 23 (3), Pág. 499-515. 2006

[2] Whelpton M, Arnbjörnsdóttir B, Teaching Needs Report. Covcell State of the Art Report. University of Iceland, Reykjavík Iceland, 2006.

[3] Hernández F, Kostina I, Curso virtual para docentes de inglés: una alternativa para la profesionalización, Íkala 10 (16), 2006.

[4] Bañados E, A Blended-learning Pedagogical Model for Teaching and Learning ESL Successfully Through an Online Interactive Multimedia Environment, CALICO Journal, 23 (3), Pág. 533-550, 2006

[5] Berdugo Martha, Pedraza Nancy. Ambiente Web para la enseñanza de Idiomas. Editorial Univalle 2006.

[6] Brusilovsky Peter, Adaptive Navigation Support in Educational Hypermedia: the Role of Student Knowledge Level and the Case for Meta-Adaptation, School of information Sciences University of Pittsburgh, Pittsburgh PA 15260. 2003

[7] Mas Ana, Agentes Software y Sistemas Multi-Agente: Conceptos, Arquitecturas y Aplicaciones. ISBN 84-205-4367-5. Edición Pearson Prentice Hall, Madrid, 2005.

[8] Cossentino Massimo, Bernon Carole, Pavón Juan, Modelling and Meta-modelling Issues in Agent Oriented Software Engineering: The AgentLink AOSE TFG Approach. 2005

[9] Bellifemine F, Caire, G at all. JADE a White Paper. 2003. Consultado en http://jade.tilab.com

[10] Lin Fuhua, Designing Distributed Learning Environments with Intelligent Software Agents. INFOSCI, Information Science Publishing, 2005.