

PIZARRA ELECTRÓNICA USANDO AGENTES COLABORATIVOS

Fernando Zacarías Flores¹, Mauricio Javier Osorio Galindo² y Francisco Javier Tobón Vázquez³

^{1,2}Universidad de las Américas Puebla

CENTIA

Sta. Catarina Mártir Cholula Puebla

C.P. 72820, Puebla, México

¹fzflores@siu.buap.mx

²josorio@mail.udlap.mx

³Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

³pacotv@mail.cs.buap.mx

RESUMEN

En este artículo presentamos una nueva propuesta en el diseño y construcción de agentes colaborativos. Pensamos que un agente inteligente debe contener las siguientes características: representación del conocimiento acerca de su entorno, razonamiento a partir de este conocimiento, planeación de las acciones a ejecutar y capacidad en la resolución de problemas. Por esto, proponemos el uso de la programación lógica en la construcción de dichos agentes. Para esto, nosotros desarrollamos una aplicación llamada “*pizarra electrónica*”, que es una generalización del trabajo presentado en [1]. En esta nueva propuesta nosotros pensamos que el uso de answer sets apoyado en Lups da al sistema una base sólida en la construcción de los agentes, en aspectos tales como: representación de las creencias en una forma más clara, sencilla y formal (basado en la lógica), además, nos permite realizar el proceso de revisión de creencias de una manera más eficiente, esto debido a que Lups es un lenguaje para el manejo de programas lógicos cambiantes (updates). El uso de answer set en nuestra implementación de agentes nos permite que las soluciones a un problema sean representadas por un conjunto de soluciones, y no por sustituciones de respuestas producidas en respuesta a una pregunta [24]. Por otro lado, el sistema permite a los miembros de una comunidad poder establecer sus reuniones de trabajo de forma física o virtual. El sistema cuenta con herramientas tales como: chats, e-mail y un conjunto de agentes que se encargan de la negociación de reuniones requeridas por los miembros de la comunidad. Es importante señalar que nuestros agentes realizan una revisión de creencias acerca de sus usuarios con la finalidad de lograr el acuerdo más conveniente para su usuario.

Palabras clave: agentes colaborativos, agentes inteligentes, planeación, representación de conocimiento, pizarra electrónica, creencias, HCI.

1. INTRODUCCIÓN

Una de las principales actividades de nuestra comunidad es el trabajo colaborativo, este se realiza entre los diferentes grupos de investigación pertenecientes a nuestra universidad. Aquí, surge la necesidad de contar con un sistema que les apoye en el desarrollo de sus actividades, no solo a los investigadores sino también, a asistentes de investigadores, estudiantes, colaboradores externos, etc. que se ven involucrados en el desarrollo científico y tecnológico que se genera en nuestra comunidad. En este contexto, es que estamos desarrollando este nuevo prototipo, el cual tiene como función principal permitir establecer reuniones con los investigadores. Este prototipo a diferencia del presentado en [7] no solo juega el papel de una agenda, sino que también proporciona un ambiente más completo, ya que cuenta con herramientas tales como correo electrónico, salas de charla que permiten a los usuarios del sistema poder llevar a cabo reuniones virtuales en sincronía o asíncrona. Nuestro prototipo se apoya fundamentalmente en agentes lógicos racionales que tienen como características principales: razonar, planear, aprender, negociar y actuar en dominios dinámicos de forma autónoma [2,3,4]. Nosotros pensamos que el uso de answer set le da un sustento formal sólido en el manejo de creencias, tanto en la definición de éstas, como en su revisión.

Nosotros proponemos la construcción de nuestros agentes haciendo uso de la programación lógica [6,7], ya que ha mostrado ser un lenguaje muy apropiado para modelar las características antes mencionadas. En esta primera instancia proponemos el uso de Lups [5], como el lenguaje para modelar conocimiento dinámico, y ya en una segunda versión proponemos incorporar el uso de Answer sets como una alternativa en el modelado de representación de conocimiento, así como, el manejo de información incompleta e incluso, lograr que nuestros agentes puedan seguir operando aún, con la detección de inconsistencias, para lo cual haremos uso sólo de la información independiente a este problema. El tratamiento que hacemos respecto de las creencias que nuestros agentes deben manejar esta basado en el uso de algunas transformaciones sobre el programa lógico que las representa, de tal forma que el programa resultante pueda depurar las creencias del agente, tal como se muestra en [6].

La estructura que guarda nuestro artículo es la siguiente: en la sección 2 describimos de forma general el diseño e interacción con nuestro prototipo, además definimos algunos conceptos relevantes en el diseño de la interfaz de usuario; enseguida, damos un panorama general de nuestro concepto de agentes lógicos, los cuales son la base de nuestro prototipo, sección 3; en el siguiente apartado (sección 4) presentamos parte del código utilizado en el modelado de nuestros agentes, que ejemplifica como nuestros agentes establecen el proceso de negociación; en la sección 5 presentamos nuestro trabajo a futuro, dando algunas de las líneas de investigación en las que continuamos enriqueciendo nuestro prototipo; en la sección 6 damos algunas de las conclusiones obtenidas durante nuestras investigaciones; Finalmente, en la sección 7 ofrecemos nuestros agradecimientos a la institución que nos ha apoyado en este proyecto.

2. PIZARRA ELECTRÓNICA

Nuestro prototipo tienen como objetivo principal, proporcionar a los usuarios de nuestra comunidad universitaria un ambiente virtual que les permita desarrollar sus actividades de una forma más ágil, sencilla y transparente, apoyados en el uso de agentes inteligentes colaborativos. La programación de actividades tales como: horarios de cursos, horario de asesoría y la negociación de reuniones de trabajo entre miembros de nuestra comunidad, es la tarea principal de nuestros agentes inteligentes, que se describen en la sección 3. Sin embargo, el diseño e implementación de una interfaz que cumpla con los requerimientos antes descritos requiere que sea fácil en su comprensión y su uso. Es en este punto, en donde necesitamos hacer uso de una disciplina en el campo de las ciencias de computación denominada Interacción Humano Computadora (Human-Computer Interacción, HCI), HCI es la que marca los lineamientos necesarios para el correcto diseño de una interfaz que cumpla con sus objetivos, por lo tanto es conveniente prestar atención a estos lineamientos y estándares para la elaboración, diseño, implementación y funcionamiento de la Pizarra Electrónica. Nuestra aplicación brinda algunos beneficios en la manera en que se diseño (usando diseño orientado a objetos), ya que este modelo nos permite poder integrar fácilmente cualquier nuevo módulo (herramienta) o característica adicional a nuestro prototipo. Por otro lado, el hecho de que las creencias se modelen en XSB nos permite poder mantener de forma muy sencilla y rápida nuestros agentes, pues el uso de reglas para la definición y mantenimiento de creencias es transparente.

2.1 HCI

Una interfaz es una superficie de contacto, que refleja las propiedades físicas con las que el usuario interactúa. A continuación se presenta los lineamientos más relevantes marcados por HCI que se tomaron en cuenta para la elaboración de la Pizarra Electrónica.

El Factor Humano. Ya que los seres humanos están sujetos a pérdidas de concentración la Pizarra Electrónica brinda las funciones que activan o desactivan herramientas de la pantalla, esto permite al usuario centrar su atención en el horario de trabajo [16,17].

Dispositivos. Los dispositivos implementados para el manejo de la Pizarra son los estándares: teclado, ratón, monitor.

Accesibilidad. Ya que los seres humanos son diferentes entre sí, las interfaces deben acoplarse a las diferencias entre estos, de tal modo que nadie se vea limitado en el uso por causa de esas diferencias. Por tal motivo, la Pizarra Electrónica fue desarrollada con el lenguaje de programación Java [19,20], ya que nos permite implementar sistemas accesibles en diferentes plataformas y de manera modular [16,18].

Los colores y resolución empleados en la Pizarra Electrónica son legibles desde monitores blanco y negro hasta monitores de alta resolución[16,17,18].

Diseño gráfico. El diseño es la actividad encaminada a conseguir la producción en serie de objetos útiles y bellos. De esta forma los diseñadores gráficos utilizan estrategias visuales para comunicar información e influenciar emociones. Así, los gráficos que se presentan en la Pizarra Electrónica están encausados a que la información se presente de manera clara y las herramientas principales sean comprensibles [18].

Trabajo cooperativo. En nuestro prototipo el trabajo cooperativo es tarea fundamental de nuestros agentes, y es aplicado en el establecimiento y coordinación de los horarios adecuados para las reuniones requeridas por los miembros de nuestra comunidad.

Sistema de apoyo al usuario. Ya que esta es una parte importante para el usuario, se elaboraron herramientas de ayuda tales como: manual de usuario, referencias de terminología, así como información de etiquetas de ayuda para orientación del usuario[18].

Estándares y guías. En el diseño de nuestra interfaz se puede apreciar como hemos seguido los estándares propuestos en color, ubicación, imágenes, etc.[18] todo esto, con la finalidad de proporcionar a nuestros usuarios un uso fácil, sencillo y claro en el uso de nuestra interfaz.

2.2 DESCRIPCIÓN DEL FUNCIONAMIENTO

Enseguida se da una descripción del funcionamiento (instrucciones y herramientas) de la Pizarra Electrónica, con la finalidad de mostrar lo fácil y simple que resulta trabajar con nuestra interfaz. En la figura 1 se muestra la interfaz principal de la Pizarra Electrónica.

El propósito de ésta, es brindar un panorama general y claro de las actividades programadas para cada uno de nuestros miembros de la comunidad[16]. Las actividades de un miembro se presentan por semana, tomando la correspondiente a la marcada por el calendario de nuestro servidor. Las actividades son clasificadas según el color, como se explica enseguida:

Negro (Disable): indica las actividades ya programadas por el usuario y que no son susceptibles a modificación: tales como congresos, seminarios, horarios de comidas, etc.

Gris (Class): indica los horarios de clases asignados al usuario, así como, las reuniones que van siendo solicitadas a través de nuestra interfaz.

Blanco: indica que el horario está disponible para ser utilizado por cualquier miembro de la comunidad incluso por el mismo propietario para establecer una nueva actividad.

Las opciones localizadas en la parte superior de la figura 1, tales como Cargo (Employment) y Persona (Person), ayudan a facilitar la búsqueda de acuerdo con las necesidades del usuario, sea de profesores -conociendo los nombres- o de miembros administrativos -conociendo los cargos-.

En la parte superior derecha de la figura 1 se visualiza un calendario ordinario que permite al usuario seleccionar la semana de actividades del académico o administrativo; la Pizarra Electrónica toma esta selección y se encarga de obtener los datos correspondientes de la tabla de Actividades de la Base de Datos para su visualización en la Pizarra.

En el cuadro de Cargo (Employment) en la parte inferior de la Pizarra se hace la visualización del cargo del académico (si lo tiene) y de las materias que imparte; esto último se da como información adicional para el usuario que le puede ser útil según sus necesidades.

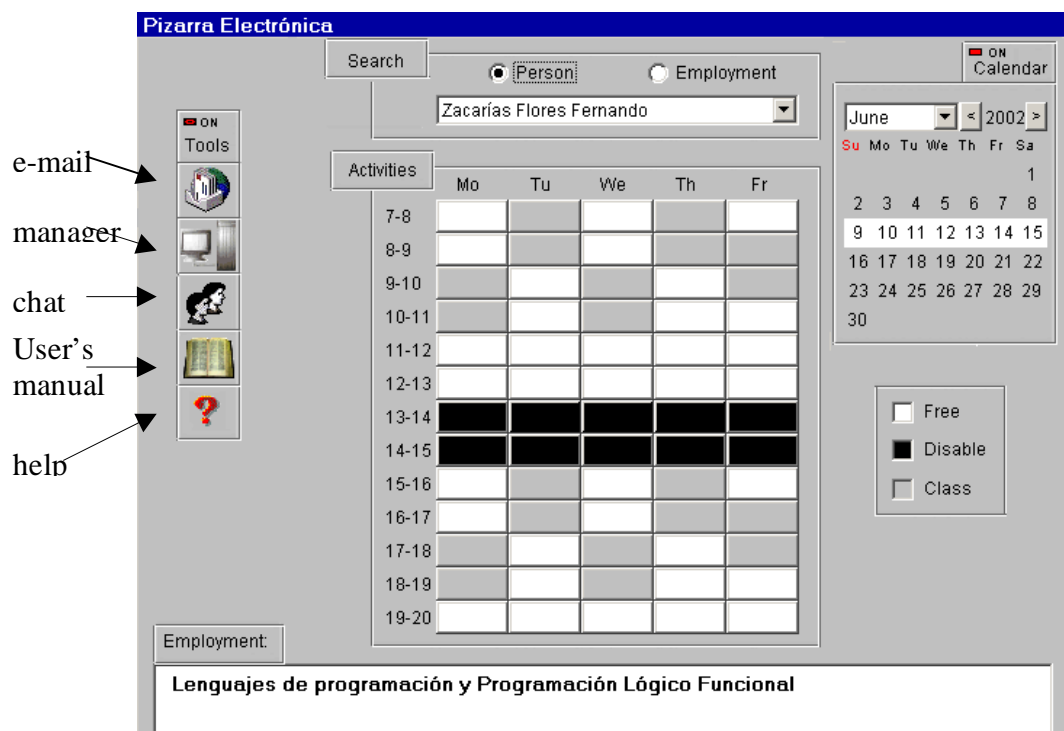


Figura.1 Pizarra Electrónica

Las herramientas incorporadas a la pizarra electrónica son:

- Chat. Proporciona este servicio con salas correspondientes a diferentes áreas de interés o investigación y para poder estar en una cita personal.
- Email. Proporciona el servicio de correo electrónico de forma asíncrona.
- Manager. Permite a los miembros de la comunidad académica administrar sus actividades, solicitándole login y password para garantizar la seguridad del sistema.
- User's manual. Describe el uso de nuestra interfaz en el ámbito de usuario
- Help. Da una ayuda general describiendo cada uno de los tópicos con que cuenta nuestro sistema.

El Administrador (Manager) además de lo descrito anteriormente permite realizar las siguientes acciones:

- Crear una nueva actividad.
- Modificar / eliminar actividad y/o reunión.
- Agregar / Eliminar miembros a nuestra interfaz.
- Modificar las propiedades de un miembro de nuestra comunidad.

Por otro lado, otro de los procesos relevantes en nuestro sistema corresponde al establecimiento de una reunión, esta se define en nuestra interfaz tal como se muestra en la figura 2. El usuario puede seleccionar a la persona y dar clic en el horario disponible que más le convenga. En ese momento el sistema le presentará la interfaz de solicitud de reunión (Appointment Interface). Una vez que es facilitada la información, el usuario debe presionar el botón de enviar (Send), esto con la finalidad de activar al agente que se encargará de realizar el proceso de negociación de la cita solicitada. Antes de que el agente inicie el proceso de negociación, se ejecuta un proceso de validación de los campos siguientes: nombre del solicitante (Applicant), correo electrónico (Email) del solicitante, lugar (Place) y

tipo, esto con la finalidad de que la información sea dada de forma correcta. Como es sabido, algunos de estos procesos de validación se han incorporado con la finalidad de cumplir con los estándares que se definen en el área de HCI.

The screenshot shows a Java Applet window titled "Appointment interface". On the left is a calendar grid with days of the week (Mo, Tu, We, Th, Fr) and time slots (7-8 to 19-20). Some slots are highlighted in black. On the right is a form with the following fields:

- To: Fernando Zacarías Flores
- Applicant: Francisco Javier Tobón Vázquez
- Subject: tesis asesoria
- Email: fjtobon@hotmail.com
- Note: Presentacion de avence de tesis
- Type: Personal (dropdown)
- Place: S1 (dropdown)
- Solicitud: Send button
- State: Please, wait ...
- Buttons: Ok, Cancel, Help

At the bottom, it says "Java Applet Window".

Figura 2. Interfaz para reunión

La parte derecha de la figura 2, muestra una forma correcta en la que se capturará la información requerida por nuestro sistema para que el agente pueda proceder al establecimiento formal de dicha solicitud. Finalmente, otra de las partes fundamentales en la Pizarra Electrónica es la concerniente a las actividades que cada miembro de nuestra comunidad puede programar (Activity Dessinger). Estas pueden ser de diversas formas, debido a las necesidades o preferencias del usuario. La complejidad de estas, hace necesario una interfaz apropiada junto con una estructura para la programación de estas, a fin de que cumpla las expectativas del usuario.

The screenshot shows a Java Applet window titled "Manager" with a sub-window titled "Activity Designer". It contains the following fields and controls:

- Activity: Seminario de Logica
- Subject: Platicas sobre investigación en Lógica Matemática
- DateB: 15/04/02
- Hour: 18-22
- ☒ Periodic
- Days: Thr, Fri (dropdowns)
- Months: Jan, Feb (dropdowns)
- DateE: 26/08/02
- Add buttons for Days and Months
- Result: Th;April-Agost;26/08/2002.
- Reset button
- Buttons: Ok, Cancel, Help

At the bottom, it says "Java Applet Window".

Figura 3. Administración de Actividades

En la figura 3 se visualiza una actividad a ser programada por el usuario miembro de nuestra comunidad. Como se observa cada uno de los miembros de nuestra comunidad puede registrar todas y cada una de las actividades que normalmente realiza, pudiendo agregar en cualquier momento alguna nueva actividad que sea única o periódica. La

actualización de las actividades le solicita al propietario un Login y Password para permitirle el acceso a esta opción (cuidando la seguridad en todo momento).

3. AGENTES LÓGICOS COLABORATIVOS

Antes de describir cual es el comportamiento de nuestros agentes daremos un panorama general de lo que pensamos deben ser los agentes [7,8,9] para lograr que trabajen de forma colaborativa [10] como en nuestro prototipo. Existen varios enfoques de lo que un agente considerado inteligente debe ser, sin embargo, muchas de estas propuestas se basan en la definición de un aprendizaje “ingenuo” [11], esto hace que estos agentes dependan totalmente de la descripción de conocimiento que su usuario le proporciona. En nuestra propuesta nosotros proponemos el uso de un mecanismo de validación que le permite a nuestros agentes verificar que el nuevo conocimiento que se les presenta no producirá inconsistencias [6] con el conocimiento previo que posee. Esto les permite garantizar al usuario que su aprendizaje es confiable y seguro.

Las arquitecturas siguientes son las que comprenderían de mejor manera el tipo de agentes modelados para nuestro prototipo.

- Agentes lógicos (deliberativos)
- Agentes BDI que presentan características referentes a creencias, deseos e intenciones

Respecto a sus características principales, los agentes inteligentes tienen:

- Una máquina de inferencia que le permite hacer una cierta clase de deducción
- Planeación, es decir, proponer un conjunto de acciones que le permitan alcanzar sus metas
- Una base de conocimientos que es actualizada a través de un proceso de aprendizaje
- Un proceso deliberativo a través de una función de mapeo entre su ambiente y la acción a realizar
- Significado, metas y razonamiento.
- Un proceso de revisión de creencias por medio de un proceso deliberativo
- Colaboración entre los agentes contenidos en un sistema.

Enseguida, presentamos nuestra propuesta de cómo diseñamos nuestros agentes colaborativos. De acuerdo a [7], la función de mapeo entre el ambiente del agente y las acciones que ejecuta es un proceso de aprendizaje basado en la programación lógica, es decir, es aquí donde usamos el formalismo de la lógica para lograr que este proceso de aprendizaje sea cauteloso y confiable. Esto con la finalidad de que nuestros agentes alcancen un aprendizaje pleno y seguro. Basados en esta caracterización de agentes, nosotros debemos modelar agentes que sean capaces de establecer un conjunto de acciones que definan el plan a ejecutar para alcanzar cada una de las metas propuestas a nuestros agentes. Incluso, de ser necesario, deben ser capaces de la re-planeación, para cuando se detectan cambios en el conocimiento del agente. Algunos de los procesos necesarios para alcanzar este objetivo son:

1.- Definir un proceso de comunicación que les permita a nuestros agentes poder comunicarse de una forma clara, sencilla y transparente. En nuestro prototipo, nosotros hacemos uso del medio de comunicación más popular en nuestros días, el uso del correo electrónico, esto les da la facilidad de comunicarse con cualquiera de los agentes que requieran establecer algún tipo de comunicación a través de la web.

2.- Que nuestros agentes sean capaces de tomar sus propias decisiones en todo momento, es decir de forma autónoma. Es aquí en donde debemos seguir investigando acerca de cómo nuestros agentes pueden manejar su experiencia, es decir, incorporar mecanismos para la toma de decisiones haciendo uso de paradigmas que exploten su experiencia, esto significa, emplear los resultados tanto buenos como malos de sus acciones pasadas. Algunos de los paradigmas que hemos explorado son: Razonamiento basado en casos y programación lógica inductiva.

3.- Que sean capaces de establecer un plan que los conduzca hacia las metas establecidas. En un sentido más general, podemos pensar en que nuestros agentes sean capaces de definir incluso el orden en que deban ejecutarse cada una de las acciones contenidas en el plan trazado. Aún más, que tengan la capacidad de la re-planeación, para situaciones que así lo requieran.

4.- Nuestros agentes manejan un proceso de aprendizaje que les permite validar que el nuevo conocimiento que se genera no cause inconsistencias o generación de conocimiento incompleto, y en caso de ser así no se incorporará a la base de datos de las citas y tampoco a la base de conocimiento del agente. Es en este punto, que pensamos nuestra propuesta da una alternativa en la solución a este tipo de problemas. Veamos el siguiente ejemplo, que nos permite visualizar cual es nuestra propuesta:

Ejemplo. Revisión de inconsistencias

Supongamos que tenemos un programa P que representa las creencias del agente y, en este programa supongamos que tenemos el siguiente conjunto de reglas como las siguientes:

- 1) d.
- 2) e.
- 3) a:- d.
- 4) b:-e.
- 5) :- a,b.

Es decir, tenemos que 1) y 2) son hechos (True), 3) y 4) son reglas que dependen de 1) y 2) y la restricción de que no se deben cumplir tanto a como b. Entonces, este segmento de código produciría inconsistencias en las creencias del agente, lo que causaría que nuestro programa completo (no solo este segmento de código) no tuviera modelos. Es aquí, que nosotros empleamos un proceso que valida que si esta situación se presenta, nosotros podamos realizar una depuración (debuging) que le indique al usuario la presencia de esta situación, y que además, le muestre cuáles son las reglas involucradas en este problema, esto con la finalidad de que decida si el agente las deja de lado (no las incorpore), y pueda continuar funcionando con el resto del programa completo, o bien, detenga su ejecución para que el usuario las analice y determine que hacer. Para situaciones en que el programa contenga muchas reglas (como es normalmente) esto es muy conveniente.

En esta primera etapa, los agentes de nuestro prototipo tiene características que les permiten trabajar de manera conjunta básica en el establecimiento de reuniones colectivas entre los diferentes miembros de nuestra comunidad. En todo momento son tomadas en cuenta las preferencias del usuario con la finalidad de alcanzar un acuerdo conveniente para él. El establecimiento de reuniones, se basa en un proceso en el cual los agentes involucrados deben alcanzar un acuerdo aceptable para ambos, tal como lo muestran [12,13].

La negociación es definida como un protocolo para el intercambio de mensajes (vía correo electrónico) conteniendo propuestas (proposal), preferencias y explicaciones. Toda esta información le permite a un agente poder decidir sobre la aceptación o rechazo de la propuesta recibida. De igual manera, el contar con información acerca de las preferencias de su usuario le permite decidir sobre estas propuestas. Enseguida, se muestra el formato de la propuesta generada y enviada por nuestros agentes:

```
proposal ( --Junta martes por la tarde
          20021015, 1500, --Fecha, hr inicial
          20021015, 1700, --Fecha, hr final
          examen, --Actividad
          meeting, --Tipo
          9, --Prioridad
          pupils, --Con
          none --Notas
          )
```

Una vez que un agente recibe esta propuesta se procede a la revisión de creencias del usuario que deberá decidir aceptar o no esta reunión. El proceso de revisión de creencias es como se presenta en [6], cuidando que esta nueva información no conduzca a contradicciones con la información previa que tiene el agente o bien que el dominio de nuestro agente quede incompleto. En nuestras investigaciones futuras, abordaremos la línea de investigación concerniente a que deben hacer nuestros agentes cuando nuevo conocimiento produce inconsistencias, es decir, nosotros esperaríamos que el agente continúe trabajando con toda aquella información que es independiente a la que causó la inconsistencia, sin embargo, cuando hacemos uso de answer set, nuestro nuevo programa no tiene modelos lo que implica que nuestro agente no actuara de forma adecuada. Nosotros pensamos que este tipo de situaciones pueden resolverse de manera satisfactoria cuidando cual debe ser el programa que debemos considerar. En la

definición de nuestros agentes, nosotros hacemos uso de algunas reglas de transformación [6,21,22,23] que le permiten a Lups trabajar con programas simplificados, lo que hace que su desempeño sea aceptable.

4.- CÓDIGO DE NUESTROS AGENTES

Como mencionamos en la introducción para el diseño de nuestros agentes requerimos que nuestros agentes puedan manejar actualizaciones referentes al conocimiento que este posee. Por tal razón, requerimos de un lenguaje que nos permita modelar nuestros agentes con esta característica, es aquí donde proponemos el uso de "LUPS - a language for updating logic programs" [5]. Lups es un lenguaje diseñado para modelar el mundo cambiante en que se desempeñan nuestros agentes, es decir, nos permite representar un mundo dinámico a través de la base de conocimientos. Lups es un lenguaje que hemos evaluado y que pensamos cumple con los requerimientos para el modelado de nuestros agentes. Existen algunas aplicaciones similares referidos en [9] y [10]. En nuestra aplicación los agentes son los encargados del proceso de negociación de las reuniones requeridas entre los miembros de nuestra comunidad para después darlo de alta en la Base de conocimientos para cada uno de los involucrados en dicha actividad.

Por otro lado, uno de los aspectos relevantes en el modelado de agentes inteligentes es el manejo de preferencias del usuario, las cuales pueden ser del tipo: desear la tarde del viernes libre, reuniones los jueves por la tarde, no fijar citas los lunes en la mañana, etc. Así el agente deberá ser capaz de manejarlas según convenga con la única finalidad de lograr un acuerdo mutuo en el establecimiento de reuniones. Concretamente, el conocimiento que se tiene del contexto, preferencias y creencias lo modelamos en lups [5]. Enseguida presentamos parte del código de nuestro agente:

```
always(proposal(  
    NewStartDate, NewStartTime,  
    NewEndDate, NewEndTime,  
    Subject,  
    Type  
    Priority,  
    With,  
    Notes  
)) when(appointment(  
    StartDate, StartTime,  
    EndDate, EndTime,  
    Subject,  
    Type,  
    Priority,  
    With,  
    Notes    )).
```

Como se puede observar, esta es la regla que se genera en Java y que es enviado al agente definido en Lups, el cual corre sobre el sistema XSB (similar a Prolog). Es importante hacer notar que en principio, el agente recibe la solicitud de reunión como una propuesta (proposal), es decir el usuario solicitante ha solicitado una cita en una fecha y hora que tentativamente puede ser aceptada por el miembro de la comunidad al que se le solicita. Enseguida, nuestro agente verifica si basado en las creencias, deseos, intenciones y preferencias de su usuario es factible aceptar dicha solicitud, de ser así, el agente envía un mensaje a través del correo electrónico notificando al solicitante que la reunión ha sido aceptada en los términos por él propuestos. En caso contrario, el agente genera una propuesta para el solicitante de la reunión en la que le propone una nueva fecha y hora vía correo electrónico para esperar la respuesta del solicitante, la cual puede ser de aceptación o bien de una nueva propuesta, este proceso se repite las veces que sea necesario hasta que los alcanzar un acuerdo mutuo.

5. TRABAJO A FUTURO

El prototipo que hemos presentado se encuentra en una etapa de enriquecimiento que pensamos puede ser extendido enormemente. Como se describió en las secciones anteriores las herramientas de chat y e-mail han sido incorporadas

con la finalidad de proporcionar a nuestros usuarios un medio de comunicación (con sincronía y asíncrono) para que puedan realizar de manera virtual sus reuniones. En nuestra siguiente versión pensamos incorporarle algunas otras herramientas para trabajo cooperativo (CSCW), esto con la finalidad de brindarle al usuario un ambiente de trabajo mas completo para la realización de sus actividades. Algunas de las herramientas contempladas son: Una herramienta como Lotus, un editor de documentos para trabajo cooperativo, etc.

Por otro lado, pensamos seguir nuestras investigaciones en la línea de la programación lógica, con la finalidad de incorporar a nuestros agentes algunos resultados [14,15] que se han obtenido en el grupo de lógica del CENTIA. Algunos en la línea de Answer set, reglas de transformación, revisión de creencias, manejo de preferencias, etc.

Finalmente, pensamos incorporar agentes de usuario que nos permitan modelar el perfil de nuestros usuarios de tal forma que puedan ser auxiliados en el desarrollo de sus actividades relacionadas con la investigación, este punto, pensamos, puede ser alcanzado a mediano plazo debido a que el sistema en su estado actual ya cuenta con alguna información respecto al perfil de usuario tal como: El tipo de materias que imparte (definidas en la parte inferior de la figura 1, employment), el tipo de cargos administrativos; que son definidos en la misma sección anterior (en caso de que tenga algún cargo), las creencias, deseos, intenciones y preferencias definidas para cada uno de los miembros de nuestra comunidad y definidos en nuestro sistema, etc.

Finalmente, un trabajo a largo plazo es la definición de un metalenguaje que nos permita modelar de forma adecuada, clara, sencilla y directa el tipo de agentes inteligentes que hemos definido y queremos enriquecer, de tal forma que cada una de sus características sean completas y correctas.

6. CONCLUSIONES

Hemos presentado un sistema que proporciona algunos de los servicios necesarios para los miembros de nuestra comunidad. Y pensamos que este sistema es un primer esfuerzo que intenta resolver uno de los problemas fundamentales en el desarrollo de nuestra comunidad, tal es el caso de la investigación, tanto en el área de la programación lógica como en la de agentes. De igual forma proporcionamos las condiciones necesarias para alcanzar que los usuarios puedan trabajar de forma cooperativa y más aún, colaborativamente, ya que es un hecho que este es el camino para lograr un estatus competitivo, tanto académica como tecnológicamente. Por otro lado, hemos presentado (como se muestra en el ejemplo de revisión de creencias) en la sección 3, como podemos atacar el problema de inconsistencias, que pueden presentarse a un agente cuando percibe nueva información de su ambiente y que de no resolver satisfactoriamente este problema, puede provocar un caos total. Finalmente, nosotros continuamos trabajando con el paradigma de answer set, ya que lo consideramos robusto, completo y altamente adecuado para la revisión de creencias que un agente debe tener y, que en aplicaciones reales, como la presentada en este artículo nos permitirá aterrizar los avances matemáticos logrados en answer set.

7. AGRADECIMIENTOS

Agradecemos el apoyo prestado a el primer autor de este artículo, quien es apoyado por el proyecto financiado por el Consejo Nacional de Ciencia y Tecnología (CONACYT project 35804-A) de la Universidad de las Américas, Puebla, CENTIA. De igual manera, agradecemos el apoyo brindado para la realización de este prototipo a la Benemérita Universidad Autónoma de Puebla.

8. REFERENCIAS

- [1] F. Zacarías and Juan Carlos Acosta, "Agentes racionales usando lógica intuicionista e inductiva", XII congreso internacional de ingeniería electrónica comunicaciones y computadoras, Conielectcomp 2002, Acapulco, México.
- [2] F.Sadri, F.Toni, "*Computational Logic and Multi-Agent Systems: a Roadmap*" Imperial College, Dec. 1999.

- [3] M.Gelfind, V.Lifschitz. Logic Programs with Classical Negation. In Proc. ICLP'90. 7th International Conference on Logic Programming , pages 579-597. MIT Press, 1990.
- [4] P.Dell'Acqua, L.M.Pereira. Updating Agents. In Proc. Of the Workshop on Multi Agents Systems in Logic Programming. In conjunction with ICLP'99. New México State University, US, December 1999
- [5] J.J.Alferes,L.M.Pereira,H.Przymusinska and T. Przymusinski, "*LUPS - a language for updating logic programs*" , in logic programming and Non-monotonic Reasoning, pages 162-176, 1999.
- [6] J.C. Acosta y M. Osorio, "Exploring Belief Revision with LUPS", MICAI2002,
- [7] Rodney A.Brooks, Intelligence without representation, num. 47 in Artificial Intelligence, pages 139-159, 1991.
- [8] G. Weiss, Multiagents Systems, 1999.
- [9] H.Nwana, "*Software agents: An Overview*", The Knowledge Engineering Review II
- [10] IBM, IBM Agent building Environment (ABE) – A tool kit for building intelligent agent applications . <http://www.networking.ibm.com/iag/iagsoft.htm>, 1997.
- [11] N.R. Jennings, E.H.Mamdani, I.Laresgoiti, J.Perez and J.Corera, GRATE: A General Framework for Co-operative Problem Solving. IEE-BCS, Journal of Intelligent Systems Engineering, 1(2 (Winter)): 102-114, 1992
- [12] S. Parsons and N.R.Jennings. Negotiation through argumentation. In proceedings of 2nd International Conference on Multi-Agents System, pages 267-274, Kyoto Japan, 1997.
- [13] S.Parsons, C.Sierra and N.R. Jemmings. Agents that reason and negotiate by arguing. Journal of Logic and Computation. 8(3):261-292, 1998-
- [14] Mauricio Osorio, Juan Antonio Navarro and José Arrazola. Equivalence in Answer set programming. In proceedings of LOPSTR, pages 18-28, 2001.
- [15] Mauricio Osorio, J.C. Nieves and Chris Giannella. Useful transformation in answer set programming: Towards efficient and Scalable knowledge representation and reasoning, pages 146-152. AAAI Press, Stanford, USA, 2001
- [16] Wendy Chisholm. Techniques for Web Content Accessibility, Guidelines 1.0. W3C Note 6 November 2000 W3C; Gregg Vanderheiden, Trace R & D Center, University of Wisconsin – Madison
- [17] Wendy Chisholm. Web Content Accessibility Guidelines 1.0. W3C Recommendation 5-May-1999 Trace R & D Center, University of Wisconsin
- [18] <http://griho.udl.es/ipo/libroe.html>
- [19] Deitel y Deitel. "Cómo Programar en Java". Ed. Prentice Hall
- [21] J.Dix, M.Osorio,C.Zepeda, "A general theory of Confluent Rewriting Systems for Logic Programming and its Applications", Anales of Pure Logic.
- [22] M.Osorio and F. Zacarías, High-Level Logic Programming, Lecture Notes in Computer Science, 1762, Foundations of Information and Knowledge Systems, Foiks 2000, Springer, Berlín Germany.
- [23] Jurgen Dix and Mauricio Osorio, Confluent rewriting systems
- [24] V. Lifschitz, "[Answer set planning](#)," in *Proceedings of the 1999 International Conference on Logic Programming*, 1999, pp. 23-37.