Integración de Ontologías y Capacidades de Razonamiento en Agentes de Software Inteligentes para la Simulación del Proceso de Negociación de Contratos de Energía Eléctrica

Integration of Ontologies and Reasoning Abilities within Intelligent Software Agents for the Simulation of Electric Energy Contracts Negotiation Process

Francisco Arias, Ing.(c), Julián Moreno, MSc., Demetrio Ovalle, PhD.

GIDIA: Grupo de Investigación y Desarrollo en Inteligencia Artificial
Escuela de Ingeniería de Sistemas, Facultad de Minas
Universidad Nacional de Colombia Sede Medellín
fjarias@unal.edu.co, jmoreno1@unal.edu.co, dovalle@unal.edu.co

Recibido para revisión 26 de Marzo de 2007, aceptado 15 de Junio de 2007, versión final 31 de julio de 2007

***Resumen*—La utilización de Ontologías llega a ser imprescindible ante la complejidad inherente a las aplicaciones desarrolladas en el contexto de los Sistemas Multi-Agente. Lo anterior, debido a que se presentan las siguientes dificultades: abundancia de comunicación entre agentes, interoperabilidad de sistemas y plataformas, y problemas semánticos. Sin embargo, construir manualmente las Ontologías siguiendo las especificaciones de la plataforma JADE (Java Agent DEvelopment framework) resulta muy dispendioso por lo que se propone, en este artículo, su desarrollo mediante su modelamiento con la herramienta Protégé para luego integrarlo con JADE; todo ésto con la ayuda del plug-in Bean Generador, el cual permite generar código en forma automática. Adicionalmente, la implementación de mecanismos de razonamiento en agentes deliberativos puede ser lograda mediante la integración de las plataformas JADE y JESS (Java Expert System Shell) con las que es posible implementar agentes de software híbridos (reactivos + deliberativos) mediante mecanismos de razonamiento basados en reglas. El propósito de este artículo es ilustrar y resaltar las bondades en la integración de estas tres herramientas y su validación mediante una aplicación en concreto: Simulación de la negociación electrónica de contratos en el Mercado de Energía Eléctrica en Colombia.**

***Palabras Clave*—Sistemas Inteligentes, Sistemas Multi-Agente, Ontologías, Mecanismos de Razonamiento, Sistema Electrónico de Contratación de Energía, Plataformas JADE, JESS, Protégé.**

***Abstract*—The use of Ontologies becomes to be essential facing the inherent complexity of Multi-Agent applications that have been developed. This is due to the following issues: abundance of communication between agents, semantic interoperability of systems and platforms, and semantic problems. However, to manually construct the Ontologies following the specifications of the JADE (Java Agent DEvelopment framework) platform is a very expensive task. That is why, in this paper, it is proposed its development by means of its modeling with the Protégé tool, and then, its integration with the JADE framework. This integration is facilitated by the Bean Generator plug-in, which allows to automatically generating code. In addition, the implementation of reasoning mechanisms in deliberative agents can be reached by means of JADE y JESS (Java Expert System Shell) frameworks integration. In fact, with those it is possible to build hybrid software agents (reactive + deliberative) using rule based reasoning mechanisms. Thus, the aim of this paper is to illustrate and underline the advantages when integrate previous three tools and its validation using a concrete study case: the Simulation of Electric Energy Contracts Negotiation Process.**

***Keywords*—Intelligent Systems, Multi-Agent Systems, Ontologies, Reasoning Mechanisms, Electronic System of Electric Energy Contracting, JADE, JESS, Protégé Frameworks.**

I. INTRODUCCIÓN

SEGÚN la propuesta hecha por la Comisión Reguladora de Energía y Gas CREG [1,2], la negociación electrónica de contratos en el Mercado de Energía Eléctrica en Colombia se transará en un mercado organizado, con un administrador central, el cuál se encargará de hacer cumplir las reglas operativas del nuevo sistema: contratos estandarizados, anonimato y señales de precios. Estos contratos se formarán mediante subastas en el SEC, en el que se invita a los participantes a poner sus ofertas en un sitio Web en donde éstos puedan ver las posturas de los demás oferentes, y tengan la posibilidad de mejorar sucesivamente sus ofertas a fin de intentar ganarse el contrato. Debido a la inminente adopción de este nuevo sistema y puesto que este problema presenta características intrínsecas para ser desarrollado haciendo uso de los sistemas Multi-Agente, se propuso en el proyecto DIME "simulación del proceso de negociación de contratos normalizados bilaterales de energía para el sector eléctrico colombiano" [3] la creación de un sistema Multi-Agente que simule el comportamiento de los agentes del mercado bajo estas condiciones.

En este sistema se vio la necesidad de crear agentes híbridos debido a que algunos actores propios del mercado de energía, que se pretenden modelar, presentan características de razonamiento para la toma de decisiones con el fin de cumplir con sus objetivos. Dichos agentes integran dos tipos de agentes: agentes reactivos, cuya única capacidad es reaccionar inmediatamente a través de una acción sencilla cuando alguna condición se cumple y agentes deliberativos quienes poseen la capacidad de efectuar operaciones complejas, son individualmente inteligentes, pueden comunicarse con los demás agentes y llegar a un acuerdo con todos o algunos de ellos, sobre alguna decisión a tomar [4].

Por otro lado, para este mismo proyecto, también se vio la necesidad de desarrollar una ontología específica, mediante la cual es posible implementar un modelo del dominio que asegure una semántica completa y sin ambigüedades, y por ende facilita la comunicación e interoperabilidad entre diferentes sistemas y plataformas.

En este artículo se pretende ilustrar la integración de las plataformas JADE [5] y JESS [6] para el diseño y desarrollo de los agentes híbridos, aplicada en el caso particular de un Sistema Multi-Agente que simula el proceso de negociación de energía eléctrica en Colombia. Además, para este mismo caso se ilustra la integración de la herramienta Protégé y la plataforma JADE, a través del plug-in Bean Generador, con el cual se puede observar cómo es la estructura inicial de las ontologías especificadas en JADE [7] y cómo se facilita su generación mediante la integración propuesta en este artículo.

II. PROBLEMA A RESOLVER

A. *Estructura del Mercado Eléctrico en Colombia*

El caso de estudio propuesto para generar e integrar los mecanismos de razonamiento y las ontologías consiste en el modelamiento de Negociación electrónica de contratos bilaterales en el Mercado de Energía Eléctrica en Colombia implementando un mecanismo de subasta como protocolo de negociación. Cabe señalar, que este caso de estudio posee las características intrínsecas necesarias para la implementación de los mecanismos de razonamiento y las ontologías. Los primeros, debido a las complejas decisiones que algunos agentes del mercado, que van a ser simulados, deben de tomar en ciertos momentos. Las segundas, debido al conocimiento común que deben manejar los participantes del negocio (e.g. los agentes del mercado de energía eléctrica) y a la gran cantidad de información que se maneja en cada una de las transacciones asociadas al proceso de negociación de energía. El Mercado de Energía Mayorista -MEM- en Colombia esta organizado como se muestra en la Figura 1. En este mercado hay varios tipos de entidades involucradas [1], a saber:

- Generadores: su papel principal es generar y vender energía en el mercado, bien sea en la Bolsa o a otros comercializadores en el largo plazo. Sin embargo, también pueden asumir el rol de compradores para protegerse de eventualidades que les impidan cumplir las obligaciones contraídas con sus clientes a largo plazo, o estructurar portafolios para obtener ganancias de los movimientos de los precios en los mercados de corto y largo plazo, en cuyo caso actúan como comercializadores.

- Comercializadores: venden energía a otros agentes del Mercado o a los consumidores finales; la venta de energía puede realizarse de dos formas, por medio de contratos bilaterales o transacciones en la Bolsa de Energía. Actualmente los contratos se deben realizar de tipo "Pague lo Contratado", en los cuales el comercializador paga la cantidad estipulada en el contrato independiente de si lo consumió en su totalidad o no. En los casos en los que el contrato es mayor que la demanda la diferencia se transa en la bolsa al precio que rija en ese momento.

- Transmisores: son las empresas propietarias de activos en el Sistema de Transmisión Nacional (STN) que operan a tensiones iguales o superiores a 220 kV. Son actores pasivos en el MEM, ya que tienen prohibido participar en la comercialización o en la generación de la energía en razón del monopolio de su actividad.

- Distribuidores: son las empresas eléctricas propietarias de redes de distribución que operan a tensiones menores de 220kV. Tienen tarifas reguladas para cubrir los costos de su actividad económica, al igual que los transmisores. En la actualidad todos los distribuidores son comercializadores en el MEM.

- Clientes o consumidores: son los usuarios finales de la energía. Los usuarios finales se dividen en dos categorías: Usuarios regulados y usuarios no regulados. La diferencia entre estas categorías está en el manejo de los precios o tarifas que son aplicables a la venta de electricidad. Para los usuarios regulados, la tarifa es establecida por la CREG mediante una fórmula tarifaria. Para los usuarios no regulados, los precios son libres y acordados entre las partes.

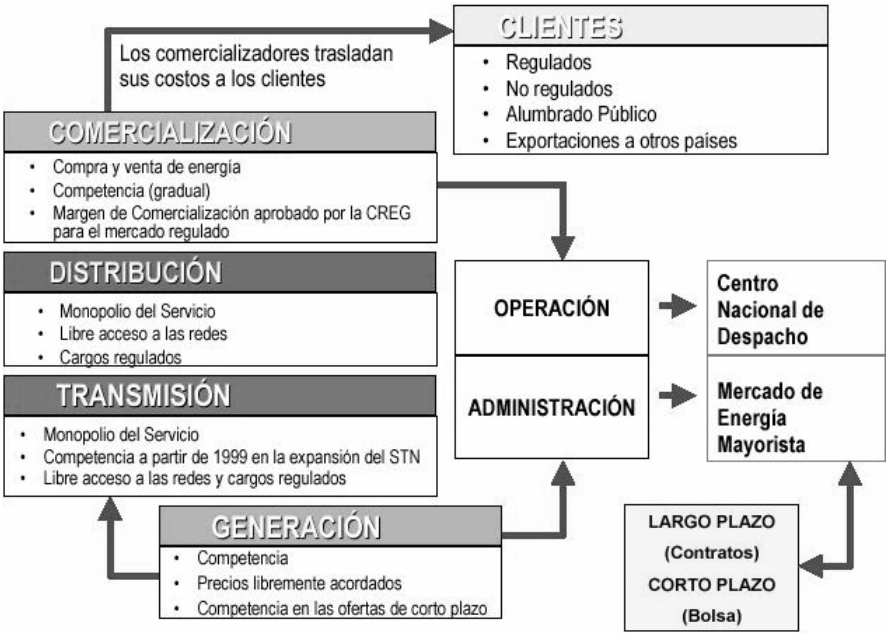

Figura 1. Estructura del mercado eléctrico en Colombia

B. Sistema Electrónico de Contratos –SEC–

El Sistema Electrónico de Contratos –SEC- [2], busca darle al mercado una propuesta para transar electricidad a través de contratos bilaterales a largo plazo o en bolsa, que les permita a los agentes contar con nuevos mecanismos para cubrir su riesgo. Aunque el diseño de este sistema se encuentra en discusión, la propuesta de la CREG busca dar solución a algunos de los problemas que se presentan con la forma actual de contratación. Algunas de las características que poseerá este sistema son:

Contratos estandarizados: Los agentes pueden a través de derivados financieros, diseñar estrategias para cubrir su riego de precio, con el objetivo de estabilizar su flujo de caja y contrarrestar los cambios de precios, tomando la posición contraria a su perfil de riesgo. Es así como si un generador que a manera general tiene vender energía, es decir, está largo, cubre su posición tomando una posición corta. Para un comercializador en este caso sucede al contrario.

Anonimato: El sistema está diseñado de tal forma que los agentes no conocen con quien están transando y es el ASIC quien se convierte en la contraparte para cada agente. Este hecho según los principios del SEC, busca fomentar la entrada de nuevos agentes, es decir, incrementar la competencia.

Señal de precios: La propuesta busca que el mercado desarrolle un indicador, basado en el cálculo de promedios ponderados, con el objetivo de desarrollar una señal para los agentes que les permita tener un mejor pronóstico de los precios futuros.

Además, el proceso de negociación que se realizara en el SEC estará basado en un mecanismo de subasta de venta. Las subastas son mecanismos para la venta o compra de bienes que bajo un protocolo específico determinan quién obtiene el bien transado y cuánto se debe pagar por él. Las subastas tienen como característica fundamental la existencia de información asimétrica entre quienes participan.

III. Modelo de Simulación Propuesto

Este modelo consiste en un sistema Multi-Agente compuesto por 3 tipos de agentes: generador, comercializador y administrador. Los dos primeros tipos de agentes pueden poseer varias instancias, mientras que el tercer tipo de agente solo puede tener una instancia. Estos agentes fueron abstraídos del problema a resolver expuesto en la sección 2. Los agentes generador y comercializador actúan de forma similar a como se explico anteriormente, el agente administrador es un agente intermediador, el cual tiene como objetivo principal controlar el desarrollo de la negociación. Los demás tipos de agentes, ilustrados en la figura 1, no son modelados debido a que estos no son necesarios para el desarrollo de la negociación.

A. Modelo de Razonamiento

En el desarrollo de la aplicación fueron identificadas dos tareas principales en las cuales intervienen tres tipos de agentes: generador, comercializador y administrador; estas tareas corresponden a la compra de energía en el SEC y la compra de energía en bolsa. Para mostrar la implementación de los mecanismos de razonamiento se seguirá todo el proceso para una actividad contenida en la tarea “comprar energía en el SEC”, desde la identificación y modelamiento hasta su implementación e integración. La tarea “Comprar energía en el SEC” se lleva a cabo de la siguiente manera: el generador envía sus ofertas al SEC, este las registra y las publica a los comercializadores. Posteriormente, los comercializadores revisan las ofertas y teniendo en cuenta parámetros internos y externos envían incrementos sobre ciertas ofertas al SEC. El SEC analiza estas contra-ofertas para determinar las contra-

ofertas ganadoras hasta el momento. Si la subasta no ha finalizado, el SEC publica a los comercializadores las contra-ofertas ganadoras y el proceso se repite nuevamente. Cuando la subasta finaliza, el SEC envía al administrador las contra-ofertas ganadoras en la subasta. En esta tarea se identificaron 5 actividades que requieren de un determinado razonamiento (ver Tabla 1). La actividad que se seguirá para explicar su modelamiento, implementación e integración será la actividad 4 "determinar incremento", ya que esta requiere ser realizada cuidadosamente, debido a que si el incremento es muy bajo se podría perder la subasta y si es muy alto las ganancias podrían reducirse significativamente.

Tabla 1. Actividades identificadas que requieren de algún mecanismo de razonamiento

ACTIVIDADES	AGENTE ASOCIADO
Actividad1: Determinar precio y cantidad de energía a ofertar.	Generador
Actividad2: Determinar cuanta energía comprará en contratos de largo plazo y cuanto en bolsa de energía.	Comercializador
Actividad3: Determinar por cuales ofertas realizará pujas o contraofertas.	Comercializador
Actividad4: Determinar el incremento de las contraofertas en el desarrollo de las subastas.	Comercializador
Actividad5: Determinar el precio de reserva de cada tipo de contrato para cada periodo.	Comercializador

Para modelar el razonamiento de esta actividad es necesario identificar los criterios (relaciones relevantes compuestas por 1 o mas parámetros ya sean del mercado, de la oferta o internas del comercializador) que afectan de forma directa al razonamiento, para luego asignar un porcentaje de salida (entre 0.3% incremento mínimo y 5% incremento máximo) que se fijará por medio de funciones de relación, como se muestra en la sección 3.1.1. Además, se asignará un valor entre 0 y 1 (peso) según la importancia del criterio sobre el razonamiento. Tal peso dependerá de la duración del contrato (CE-mes o CE-año). Un resumen de los análisis hechos sobre algunos de los criterios y su respectivo porcentaje basado en la importancia se muestran en la Tabla 2. Luego de tener claros los criterios y teniendo en cuenta que las ofertas por las que el comercializador va a pujar son las ofertas iniciales o las contraofertas ganadoras en un determinado periodo, entonces el primer paso que se debe tener en cuenta para determinar un incremento es obtener el nombre o identificador del ofertante, el cual será comparado con el nombre o identificador propio del comercializador. En caso de ser iguales, el incremento para la oferta en la puja actual debe ser cero, ya que el comercializador ganador de la oferta en ese instante es él mismo. En caso de ser diferentes el razonamiento para obtener el incremento debe continuar debido a que la oferta puede ser la inicial puesta por los vendedores, u otro comercializador se encuentra ganado dicha oferta en este instante. Luego de verificar que los identificadores sean diferentes se debe calcular la reserva (Reserva = Precio tope – Precio ganador. el precio ganador, es el precio actual de la oferta). Si la reserva es menor que cero es por que el precio de la contra-oferta por la cual el comercializador desea pujar excede su precio tope, y en este caso el incremento debe ser cero; si la reserva es mayor que cero el razonamiento para obtener el incremento debe continuar. Posteriormente se verifica la duración del contrato (CE-mes o CE-año), se calcula la diferencia entre el precio promedio contratos propio contra el precio ganador y se calcula la diferencia entre el precio promedio contratos mercado contra el precio ganador. Esto con el fin de verificar que porcentajes de salida asociar a cada uno de los criterios. A medida que se obtengan los porcentajes de salida, estos se deben ir multiplicando por los pesos asociados e ir sumándolos, de modo que al finalizar el razonamiento se pueda obtener un porcentaje de incremento que conlleve a un incremento lógico para la oferta. Esta es una manera de ponderar, ya que se asocian todos los parámetros relevantes en este tipo de razonamiento en criterios definidos, que pueden ser calificados por medio de un porcentaje de importancia, de modo que al sumar la salida de todos los criterios se obtendrá un valor ponderado que reúne todos los parámetros relevantes.

Tabla 2. Factores de ponderación de los criterios de entrada en la inferencia

Criterio	Descripción	Proporcionalidad	Peso CE-mes	Peso CE-año
Reserva	Diferencia entre precio tope del comercializador y precio ganador de la oferta hasta el momento.	Directa	0.2	0.22
Propensión riesgo	Indica que tan arriesgado es un comercializador para comprar energía en bolsa	Inversa	0.04	0.4
Diferencia demanda proyectada demanda actual CE_anual	Igual que el anterior, pero teniendo en cuenta que se pujando para un contrato CE-año	Directa	0	0.08
Embalse	Indica el nivel de embalse ofertable general en todo el país.	Inversa	0.14	0.1

1) Calculo de porcentaje de salida para el criterio embalse.

A continuación se definirá una función de relación que determina un porcentaje de incremento según el criterio embalse. El embalse es un criterio compuesto principalmente por los parámetros embalse actual, embalse histórico, periodo actual y periodo inicio del contrato. Si el embalse es alto entonces el precio en bolsa es bajo, debido a que es posible generar mayor energía a menor costo, entonces comprar la energía en bolsa resultaría más favorable para los comercializadores y no tendrían la necesidad de comprar energía por medio de contratos largo plazo. En cambio si el embalse es bajo el precio en bolsa es alto, entonces comprar la energía en bolsa resultaría muy caro para los comercializadores y tendría la necesidad de ganar los contratos largo plazo para no bajar la utilidad teniendo que comprar su demanda propia de energía en la bolsa. Del análisis anterior se puede concluir que el criterio embalse actúa de forma significativa e inversamente proporcional en este razonamiento (ver figura 1).

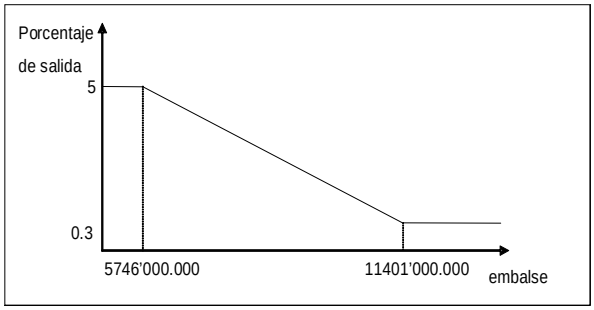

Figura 2. Grafica función de relación asociada a la criterio embalse

La función matemática para asignar el porcentaje de salida para este criterio es como se muestra en la ecuación 1.

$$0 \le embalse < 5.746'000.000 \Rightarrow$$
$$Porcentaje\ Salida = 5$$

$$5.746'000.000 \le cantidad \le 11.401'000.000 \Rightarrow$$
$$Porcentaje\ Salida = Pendiente * Embalse + Intercepto$$
$$Donde:$$
$$Pendiente = \frac{Porcentaje\ Min - Porcentaje\ Max}{EmbalseMax - EmbalseMin}$$
$$Intercepto = Porcentaje\ Max - Pendiente * EmbalseMin$$

$$Embalse > 11.401'000.00$$
$$0 \Rightarrow Porcentaje\ Salida = 0.3$$

B. Modelo de Ontologías

Los Sistemas Multi-Agente tratan sobre la coordinación inteligente entre una colección de agentes de software autónomos o semi-autónomos, los cuales existen dentro de cierto contexto o ambiente, se pueden comunicar entre sí y definen cómo pueden coordinar sus conocimientos, metas, propiedades y planes para la toma de decisiones o para resolver problemas complejos [8]. Muchos sistemas, en los que las comunicaciones juegan un papel fundamental, entre ellos los sistemas Multi-Agente, hacen uso de las ontologías como una base para dichas comunicaciones. Las ontologías son vocabularios formados por conceptos que son comunes a las personas y/o aplicaciones que trabajan en un dominio específico. Según el consorcio W3C (World Wide Web Corsortium) [9], una ontología define los términos que se usan para describir y representar un cierto dominio de aplicación. Se usa la palabra "dominio" para denotar un área específica de interés o un área de conocimiento. En la figura 2 podemos identificar los principales conceptos que van a ser parte de nuestra ontología.

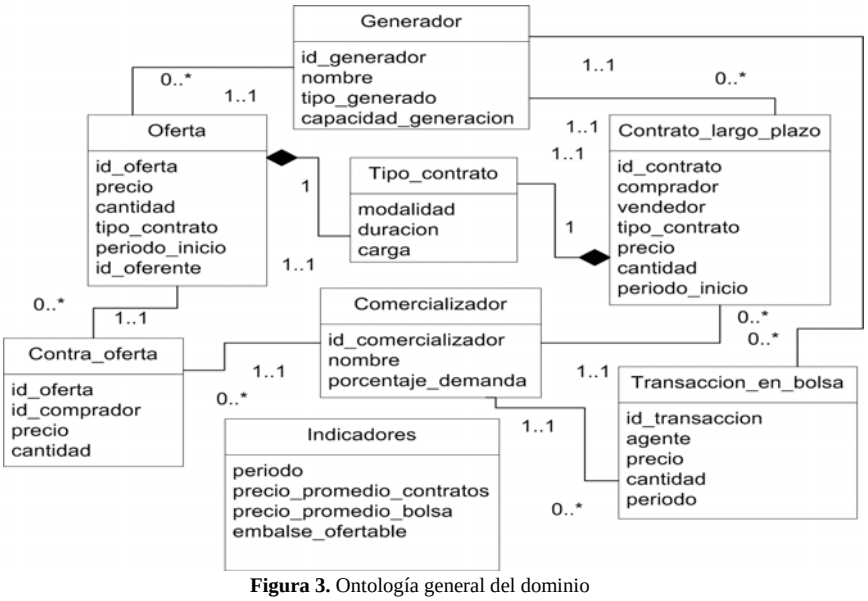

Figura 3. Ontología general del dominio

Los conceptos generador y comercializador son utilizados para el registro de estos agentes ante el administrador; los

conceptos oferta y contra-oferta son utilizados en la negociación para que los generadores puedan emitir sus ofertas y los comercializadores puedan realizar pujas (contra-ofertas); los conceptos contrato largo plazo, tipo contrato y transacción en bolsa se utilizan para que el administrador registre los contratos a largo plazo y en bolsa resultante de las negociaciones en el sistema; por ultimo, el concepto indicadores sirve para publicar los indicadores generales del mercado en la simulación a los generadores y comercializadores.

IV. HERRAMIENTAS UTILIZADAS

A continuación se presenta una breve revisión de las plataformas y herramientas de desarrollo utilizadas para la implementación del problema a resolver.

A. Plataforma JADE

JADE (Java Agent DEvelopment Framework) [10] es un middleware de código abierto y libre que proporciona tanto un entorno de desarrollo como un entorno de ejecución para la realización y mantenimiento de Sistemas Multi-Agente. El entorno de desarrollo está formado por una serie de librerías en Java que permiten la implementación de agentes de manera limpia e independiente de la plataforma sobre la que se va a ejecutar. El entorno de ejecución permite a los agentes de software actuar y comunicarse entre ellos. Está realizado enteramente en Java y proporciona una serie de herramientas gráficas fáciles de utilizar e interpretar que permiten al desarrollador controlar y depurar a los agentes en tiempo real. Después de realizar una comparación con plataformas de agentes como ZEUS o FIPA-OS, se decidió utilizar JADE debido a que esta diseñada para trabajar bajo Java, lo que la hace muy fácilmente integrable con otras plataformas Java. Además es una plataforma ampliamente usada por lo que es fácil encontrar material de apoyo de diversas fuentes. Por, cabe señalar que JADE cumple con las especificaciones FIPA [11], lo que la hace teóricamente ínter operable con otras plataformas de agentes que también cumplan con este estándar.

B. Plataforma JESS

JESS [6] (Java Expert System Shell) no es sólo un shell para sistemas expertos, como motor de reglas propiamente dicho, sino que también provee un lenguaje de script. Toda esta funcionalidad escrita completamente en el lenguaje Java, por Ernest Friedman-Hill en Sandia Nacional Laboratories en Livermore, Canadá, permite el desarrollo de sistemas expertos basados en reglas, que pueden acoplarse de diferentes formas con el lenguaje de programación Java.

JESS, aunque esta programado en Java permite modificar sus métodos y librerías, fue desarrollado en base a CLIPS lo que admite poder compilar casi cualquier archivo implementado en este lenguaje. Así como CLIPS maneja hechos y reglas, JESS los puede manejar igualmente, pero también instancias de objetos java, lo cual coloca a esta herramienta un paso más adelante en lo que a los lenguajes basados en reglas se refiere. El hecho de que java pueda comunicarse con JESS a través de un motor de inferencia es la característica más interesante en cuanto a la implementación de los distintos algoritmos en cada uno de los comportamientos deliberativos de los agentes que los requieren. Es por estas razones que hemos escogido el lenguaje JESS para la implementación de los razonamientos de los agentes.

C. Herramienta Protégé

Protégé [12,13] es una plataforma de código abierto y libre que proporciona un conjunto de herramientas para construir ontologías a partir de modelos del dominio para aplicaciones basadas en conocimiento. Protégé implementa un gran conjunto de estructuras para el modelamiento del conocimiento que soportan la creación, visualización, y manipulación de ontologías representándolos en varios formatos. Además, Protégé puede ser extendido por medio de una arquitectura de plug-ins y una interfaz de programación de aplicaciones basada en Java (API) para la construcción de herramientas y aplicaciones. Protégé soporta dos maneras de modelar ontologías: el editor Protégé-Frames y el editor Protégé-OWL. El primero permite construir ontologías basadas en frames o estructuras de modelamiento, de acuerdo con el protocolo abierto de conectividad basado en conocimiento OKBC (Open Knowledge Base Connectivity). El segundo permite construir las ontologías para la Web semántica, en particular en el lenguaje de Ontologías Web OWL (Ontology Web Language) de la W3C (World Wide Web Corsortium) [9]. La manera de modelar nuestras ontologías es mediante el primer método (frames). El aporte mas importante que nos hace protege, no es solo el modelamiento de las ontologías, sino la posibilidad de generar automáticamente el código que las represente en JADE mediante el uso de cierto plug-in.

V. INTEGRACIÓN DE JADE Y JESS

Para lograr la integración de las plataformas mencionadas e implementar los mecanismos de razonamiento definidos dentro de los agentes de software que hacen parte del modelo de simulación llevamos a cabo los siguientes pasos generales:

- Adicionar e importar la librería o API de JESS a nuestro proyecto de agentes que trabaja con el API de JADE sobre el lenguaje Java.
- Crear variables y métodos dentro de una clase que permitan la definición de los criterios de entrada del razonamiento y el cálculo de los porcentajes de salida de las funciones de relación para cada una de los criterios.
- Luego se instancia el algoritmo Rete, el cual es el encargado de controlar todo el mecanismo de búsqueda de hechos, disparar las reglas y todo el razonamiento en general.
- Crear las plantillas necesarias para la inserción de hechos que luego dispararán las reglas que se encargan de realizar el incremento.
- Luego, se debe insertar dentro del algoritmo la base de hechos iniciales (entradas) utilizando las plantillas del paso anterior.

- A continuación se crean las reglas que serán disparadas por el algoritmo Rete de acuerdo a los hechos iniciales. En cada uno de los consecuentes de las reglas se ponderan las salidas de las funciones de relación con su porcentaje de importancia de cada criterio (ver tabla 2). Cada una de estos resultados parciales se van totalizando hasta obtener el incremento definitivo.
- Finalmente se pone a correr el algoritmo Rete para que se realicen las creaciones e inserciones de hechos y disparo de reglas y de este modo obtener el incremento.

Luego de haber implementado el razonamiento según estos pasos, se realiza un llamado a este razonamiento desde un agente, en este caso un agente *comercializador*, y así obtener un incremento razonable de acuerdo a la información propia y a la información del mercado.

VI. Integración de Jade y Protégé

Plataformas para el desarrollo de agentes de software como JADE [5] brindan una estructura predefinida para el desarrollo de las ontologías en los Sistemas Multi-Agente. Sin embargo, cuando el dominio de aplicación de estos sistemas es muy extenso puede resultar muy compleja su implementación. Para aliviar esta situación es apropiado el uso y la integración de herramientas como Protégé [12,13] para realizar un modelamiento del dominio mediante interfaces gráficas. Junto con esta herramienta se puede utilizar el plug-in 'Bean Generator', muy práctico en su manejo, el cual permite generar las ontologías modeladas en Protégé de forma automática y listas para ser utilizadas directamente por la plataforma JADE.

Una ontología en JADE está compuesta por los conceptos relevantes al dominio del problema y la estructura de los predicados y acciones de los agentes de software [7]. Un concepto es una expresión que identifica una entidad con una estructura que puede ser definida en términos de campos o atributos. Un predicado es una expresión que dice algo acerca del estado del mundo, un predicado contiene conceptos, pero tambien puede contener atributos sencillos. Existe un tipo especial de concepto llamado acción, que indica una acción que puede ser desempeñada por algún agente. Además, para cada uno de estos elementos que se definen en la ontología se debe crear una clase (código) asociada. JADE permite el intercambio de mensajes utilizando información referida a una cierta ontología, esta información es transformada por JADE en una cadena de texto (String) estructurada que se incluye en el contenido del mensaje a transmitir. Luego de conocer cómo funciona la negociación de contratos normalizados bilaterales en el sector eléctrico colombiano (el dominio del problema) y los componentes básicos de una ontología en JADE, podemos identificar los principales conceptos que van a ser parte de nuestra ontología (ver figura 1). Cada elemento del dominio, que posteriormente se convertirá en un concepto, tiene sus propios atributos y cada predicado podría tener atributos de tipo concepto o primitivos. Cabe señalar que la plataforma JADE no permite enviar conceptos directamente, sino dentro de predicados.

Para la generación automática de ontologías (código) mediante Protégé se debe crear un proyecto nuevo y a este agregarle el proyecto SimpleJADEAbstractOntology (que viene con el plug-in Bean Generator [14]), el cual genera las clases abstractas Concept, Predicate y Action. De estas tres se extienden las clases para cada uno de los conceptos, predicados y acciones del dominio del problema (ver tabla 3). Para poder realizar dicha extensión debemos añadir cada uno de los conceptos como subclases de la clase Concept y los predicados se añaden como subclases de la clase Predicate. Finalmente se agregan cada uno de los atributos para todas las clases que fueron adicionadas (conceptos y predicados). Luego, a partir de una ontología desarrollada en Protégé, el plug-in Bean Generator [14] genera las clases necesarias para su manejo desde JADE.

Tabla 3. Conceptos y Predicados identificados en el dominio del problema

Conceptos	**Predicados**
Oferta	Ofertar
Contra oferta	Contra ofertar
Generador	Registrar generador
Comercializador	Registrar Comercializador
Indicadores	Publicar indicadores
Tipo contrato	Publicar compras
Contrato largo plazo	Listar ofertas
Transacción en bolsa	Listar contra ofertas
Listado ofertas	Listar contratos
Listado Contra ofertas	
Listado contratos	

VII. Análisis de Resultados de los Razonamientos

En esta sección se analizará el porcentaje de incremento propuesto por dos comercializadores (ver tabla 7) según los parámetros significativos (ver tablas 5 y 6) para determinar el porcentaje de incremento sobre la oferta inicial detallada en la tabla 4. A partir de la tabla 7 iteración 1, se puede ver que los comercializadores se encuentran pujando por la oferta inicial, ya que ambos están realizando un incremento porcentual y además el valor de la oferta es igual al precio de la oferta inicial. En esta primera iteración el comercializador 1 realizará un incremento del 3.46% y el comercializador 2 del 3.56%. Por lo tanto el ganador de la oferta en ese instante será el comercializador 2. Debido a que el embalse ofertable y los aportes de los ríos son bajos y siendo la cantidad ofrecida y la diferencia de los precios promedio de contratos contra el precio ganador altos (ver tabla 5), el valor porcentual de incremento debe ser mas cercano de 5% que de 0.3%, lo cual se puede observar en los incrementos pertenecientes a los comercializadores 1 y 2. La diferencia entre los porcentajes de incremento entre comercializadores en la primera iteración mostrada en la tabla 7, radica en los criterios que involucran parámetros propios de estos (ver tabla 6) los cuales afectan directamente a cada uno de ellos. Es posible observar en la

tabla 6 que él comercializador 1 posee una propensión al riesgo muy alta, lo cual indica que a este no le importa si tiene que comprar mucha energía en bolsa y por lo tanto no necesita realizar pujas altas; en cambio el comercializador 2 posee una propensión al riesgo muy baja, y por esta razón este comercializador debe realizar pujas con valores altos de modo que pueda obtener los contratos. Otros parámetros que hacen la diferencia en cuanto al porcentaje de incremento propuesto por los comercializadores son el precio tope y el precio promedio de los contratos propios. En cuanto al precio tope, es posible observar en la tabla 6 que este valor es mayor para el comercializador 2, lo cual le permite realizar pujas mas altas, ya que su reserva (precio tope menos precio de la oferta) será mayor que la reserva del comercializador 1; y en cuanto al precio promedio de los contratos propios, se ve que este valor es mayor que el precio de la oferta inicial en ambos comercializadores (ver tablas 4 y 6), y es menor para el comercializador 2, por lo tanto este comercializador requiere reducir el valor porcentual de su puja, en una cantidad menor a la que requiere el comercializador 1.

Tabla 4. Datos iniciales de las ofertas

ID oferta	OF_0012
Oferente	generador 2
Precio	$50.0
Cantidad	200.0 Kwh
Duración	CE-año
Periodo inicio del contrato	2

Tabla 5. Datos Generales

Periodo actual	3
Precio promedio contratos mercado	$60.29
Embalse ofertable	8064.70 Mwh
Aportes de los ríos	1758.51 Mwh
Aportes de los ríos (histórico)	1904.92 Mwh
Embalse ofertable (histórico)	6925.82 Mwh
Precio promedio contratos mercado (histórico)	$72.01

Tabla 6. Datos de los agentes

Parámetros	**Com 1**	**Com 2**
propensión riesgo	80 %	30 %
Precio tope	$74.75133	$75.075554
Precio promedio contratos propios	$60.77182	$59.803635
Demanda proyectada	105.5 Mwh	105.5 Mwh
Demanda contratada acumulada	45 Mwh	45 Mwh

Tabla 7. Desarrollo de la subasta

		Com. 1		**Com. 2**	
It.	**Val.**	**Inc.**	**Total**	**Inc.**	**Total**
1	$50.0	3.46 %	$51.73	3.56 %	$51.78*
2	$51.78	3.46 %	$53.57*	0.0 %	$51.78
3	$53.57	0.0 %	$53.57	3.56 %	$55.48*

***** Valor de la oferta ganadora en la iteración

Es posible distinguir parámetros (ver tabla 5) que afectan por igual a todos los comercializadores y no tienen un impacto relevante entre comercializadores que quieran razonar sobre el porcentaje de incremento sobre una misma oferta en un mismo periodo, sin embargo son de mucha utilidad para obtener en un principio un valor general del porcentaje de incremento que se debe tener en cuenta en cada uno de las ofertas. En la iteración 2 el valor de la oferta se encuentra en 51.78 para ambos comercializadores debido a que el comercializador 2 es quien gano la oferta inicial con un incremento del 3.56%. En esta misma iteración el incremento es igual para el comercializador 1 (3.46) debido a que el único criterio que podría afectar este valor en este instante (reserva) sigue siendo muy alto y por lo tanto su porcentaje de salida asociado será el mayor valor (5%), como ocurrió en la primera iteración; en cambio como el comercializador 2 es quien se encuentra ganado la oferta para la iteración 2, entonces este comercializador no requiere realizar incrementos sobre esta, y es por esta razón que el incremento propuesto por este es de 0.0%. Por ultimo es claro que el ganador de esta oferta es el comercializador 2, ya que este realizó una ultima puja proponiendo un incremento del 3.56% sobre la oferta con precio 53.57, y por lo tanto el comercializador 2 pagará por la oferta $55.48.

VIII. Análisis de Resultados de las Ontologías

A continuación se presenta el resultado de algunas comparaciones entre dos aproximaciones: el desarrollo e implementación de las ontologías codificadas a mano (desde cero), y las ontologías que se obtuvieron automáticamente a través de la integración de la plataforma JADE y la herramienta Protégé. Estas comparaciones se hacen desde el punto de vista de la cantidad de clases, líneas código, tiempo de modelamiento y tiempo de implementación de las ontologías.

A. Comparaciones con respecto a las clases

Cuando se desarrollan las ontologías de forma manual se deben crear diferentes tipos de clases: una clase por cada uno de los conceptos, una por cada uno de los predicados, una por cada una de las acciones y por último una clase general que representa la ontología en sí, la cual es usada en la comunicación. Esta clase contiene toda la estructura de la ontología indicando cada uno de los nombres de los conceptos, acciones y predicados con sus respectivos atributos, e incluyendo la referencia hacia la clase con las cuales son implementados. Con el desarrollo de la ontología de esta forma se obtuvo un total de 21 clases, 11 para los

conceptos, 9 para los predicados y 1 para la ontología en general como se puede ver en la tabla 3. Cabe señalar que no se desarrolló ninguna acción debido a que no se necesitaron en este problema.

Cuando se desarrollan las ontologías usando la integración de JADE y Protégé, luego de hacer el modelamiento como se explico en al apartado 6, se obtuvieron 24 clases automáticamente, las cuales están divididas de la siguiente manera: 11 para los conceptos, 9 para los predicados, 1 para la ontología en general y 3 clases especiales que genera el Bean Generator: ProtegeIntrospector, ProtegeTools y SlotHolder. Estas clases las utiliza la clase ontología generada y no agregan alguna funcionalidad diferente a la ontología desarrollada manualmente. Para no afectar el funcionamiento de la ontología en JADE se recomienda no modificar estas clases y hacer caso omiso a estas, y utilizar la ontología normalmente en las conversaciones entre agentes. Es importante señalar que de esta manera solo se debe hacer el modelamiento de los conceptos, acciones y predicados y no de la ontología general ya que esta se genera automáticamente con la información extraída del modelo.

En cuanto a las clases no hay diferencia notable entre las dos aproximaciones, tan solo son tres clases de diferencia sin importar el número de conceptos o predicados. Además, esta es una diferencia que no afecta el desempeño o el rendimiento en las comunicaciones entre agentes de software.

B. Comparaciones con respecto al código

Para comparar el código se tiene en cuenta la cantidad de líneas, el orden y la documentación de las clases que son generadas al crear las ontologías. Como se puede observar en la tabla 8, la cantidad de líneas de código generadas automáticamente es mayor que las generadas manualmente, alrededor de un 50%. Se podría decir que la diferencia en cantidad de líneas de código entre conceptos y predicados se debe principalmente a la documentación que produce el Bean Generator. La diferencia entre las ontologías generales se debe más a la estructura de la implementación que realiza el Bean Generator, pues efectúa una serie de llamadas a las otras clases mencionadas anteriormente (ProtegeIntrospector, ProtegeTools y SlotHolder). Además, estas clases adicionales también cuentan sustancialmente en la diferencia de líneas de código.

Al comparar un concepto o un predicado hecho manualmente siguiendo la estructura definida por JADE y otro generado automáticamente, se podría decir que son idénticos, ya que en ambos casos se tienen los mismos atributos y se desarrollan los mismos métodos, tan solo se diferencian en el orden en que son escritos. Aunque exista una diferencia de cantidad de líneas de código notable, esta no repercute en el desempeño de las comunicaciones entre agentes de software.

Por otro lado, al comparar las ontologías generales, hechas a mano y automáticamente, se puede notar una leve diferencia, pues la forma en que se adiciona el vocabulario, los conceptos, los predicados y sus respectivos atributos es igual. La diferencia radica principalmente cuando se adicionan los nombres de los slots a una estructura de tipo HashMap, que es usada por la ontología mediante las tres clases adicionales mencionadas anteriormente. Nuevamente cabe recordar que esto no es tan importante para el uso de la ontología y que la ontología puede ser utilizada normalmente en las conversaciones, haciendo caso omiso a esto.

C. Comparaciones con respecto al tiempo de modelamiento

El modelamiento que se debe hacer cuando se desarrollan las ontologías se realiza generalmente en la fase de análisis de la construcción del software, este modelamiento se puede hacer mediante el uso de algún editor grafico o a mano, como se muestra en la Figura 3.

Cuando se desarrollan ontologías manualmente, se usa este modelamiento y las especificaciones de JADE para construir el código y la estructura de la ontología. En cambio, cuando se utiliza Protégé para diseñar las ontologías, se debe partir del modelo realizado en la fase de análisis y reconstruirlo de nuevo, como se ve en la Figura 4. Este paso adicional puede consumir un poco más de tiempo que si no se utilizara Protégé, pero este tiempo se ve recompensado posteriormente en la generación de código. Además, Protégé brinda la posibilidad de utilizar otro plug-in llamado Ontoviz que genera una gráfica automáticamente, la cual es equivalente a la realizada manualmente en la fase de análisis (ver Figura 3).

Una comparación cuantitativa en cuanto al tiempo de modelamiento puede verse así: se tiene un tiempo de modelamiento de ontologías m en la fase de análisis. Si utilizamos Protégé, el tiempo requerido es p=m+m', donde m' es el tiempo utilizado en la transición del modelo de la fase de análisis al de Protégé, es decir lo que marca la diferencia entre las dos aproximaciones. En nuestro caso en particular, el tiempo m' fue de alrededor de 1 semana, teniendo en cuenta el tiempo que toma aprender a usar la herramienta Protégé. Hasta ahora vemos que, en cuanto al modelamiento, usar la herramienta Protégé toma un poco mas tiempo que si no se usara.

Hasta ahora vemos que, en cuanto al modelamiento, usar la herramienta Protégé toma un poco mas tiempo que si no se usara.

D. Comparaciones con respecto al tiempo de desarrollo

Este aspecto es el más significativo en cuanto a los resultados arrojados. Cuando se desea desarrollar la ontología manualmente se debe tener conocimientos previos acerca de cómo es la estructura de las ontologías según las especificaciones de JADE, lo que puede tomar normalmente unas 4 semanas, solo de aprendizaje. Por otra parte, en cuanto al paso del modelo conceptual al código de la ontología y teniendo en cuenta que ya se poseen los conocimientos anteriormente mencionados, el tiempo estimado para implementar la ontología del este caso de estudio podría tardar normalmente entre 2 semanas. Sumando todo esto, se puede ver que el tiempo necesario puede llegar a 6 semanas.

Tabla 8. Comparaciones acerca del código

Parámetros		Manualmente	Automáticamente
Cantidad de líneas de código	Conceptos	582	643
	Predicados	210	309
	Ontología General	263	296
	Otras Clases	0	321
	TOTAL	**1055**	**1569**
Orden		Depende del desarrollador	Es considerablemente ordenado
Documentación		Depende del desarrollador	Genera la documentación necesaria para crear automáticamente javadocs.

Si se utiliza Protégé para generar automáticamente el código de las ontologías no es necesario tener conocimientos previos acerca de cómo es la estructura de las ontologías según las especificaciones de JADE, lo anterior se debe a que generar el código es tan simple como dar un clic, después de haber realizado el modelamiento en Protégé, pues a partir de allí tan solo es necesario aprender a adicionar y utilizar el plug-in Bean Generador, lo que puede tardar máximo 1 semana.

Finalmente, para resumir, las ontologías desarrolladas manualmente toman 6 semanas, mientras que las ontologías generadas automáticamente toman 1 semana. Esto representa una reducción del tiempo en alrededor de un 83%, una cifra bastante significativa. Además, cabe señalar que cuando el dominio de aplicación abarca una ontología más grande, el tiempo que toma desarrollar las ontologías manualmente crece en mayor proporción que el tiempo que toma generarlas automáticamente Lo anterior sugiere que entre la ontología sea mas grande es mucho mas razonable generar el código automáticamente.

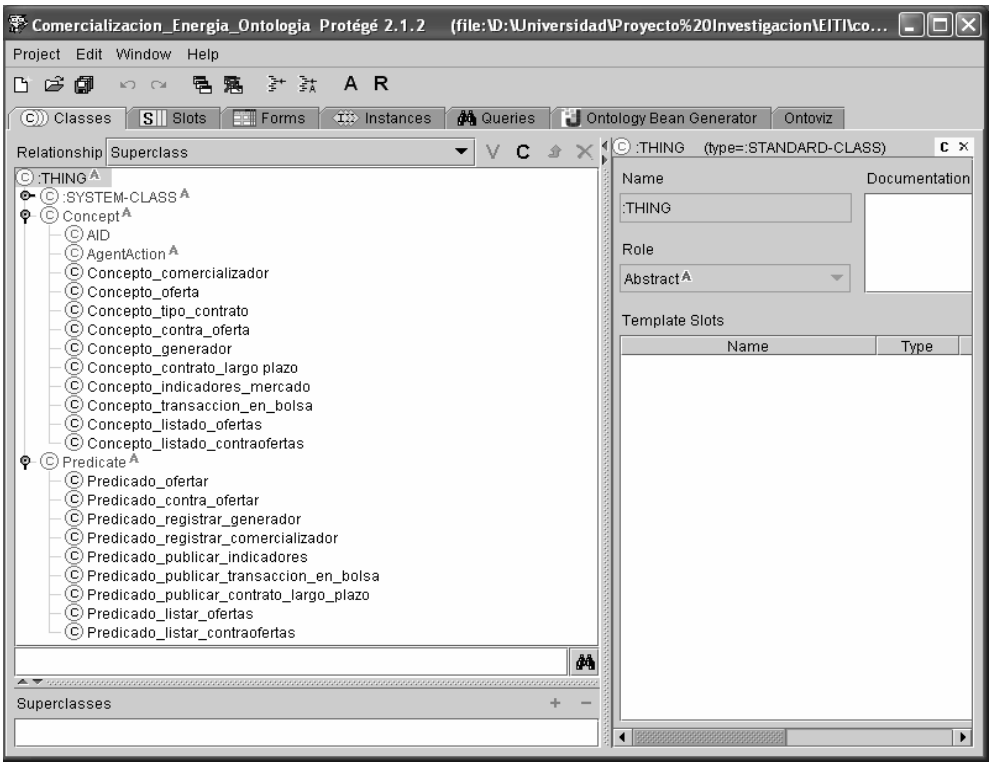

Figura 4. Modelamiento utilizando la herramienta Protégé

IX. Conclusiones y Trabajo Futuro

A En este artículo se ilustraron las bondades de la integración de mecanismos de razonamiento en los agentes de software, obteniendo agentes híbridos los cuales exhiben comportamientos reactivos y deliberativos.

La implementación de los mecanismos de razonamiento identificados en los agentes se realizó por medio de la plataforma JESS, la cual brindó buenos resultados en cuanto al desempeño, modularidad y por ende mayor control en su creación. Las plataformas utilizadas (JADE y JESS) son fácilmente integrables debido a que las dos fueron desarrolladas bajo el lenguaje Java. Del análisis realizado sobre el caso de estudio se puede concluir que la implementación desarrollada fue coherente, ya que los comercializadores simulados mostraron porcentajes de incremento acordes al valor de los parámetros de entrada para los criterios tenidos en cuenta en la definición de este

razonamiento.

Como se estableció en el apartado de análisis de resultados, el uso de Protégé para el desarrollo de ontologías para la plataforma JADE, puede tomar cierto de tiempo en cuanto al aprendizaje de esta herramienta y al modelamiento que se debe realizar de nuevo. Sin embargo este tiempo es insignificante comparado con el tiempo que ahorra en cuanto a la generación automática de código. Esto convierte a la integración de JADE y Protégé en una interesante y poderosa solución para el desarrollo de un Sistema Multi-Agente con el uso ontologías como la estructura de conocimiento para la interacción entre los agentes del sistema. Cuando se pasa del modelo de ontologías desarrollado en la fase de análisis al modelo en Protégé, y utilizando el plug-in Ontoviz para generar la gráfica del modelo, se puede realizar una comparación entre el modelo de la fase de análisis y el de Ontoviz, que permita evaluar la concordancia entre ambos. De esta manera se puede asegurar que la ontología generada por Protégé es la que se quería diseñar desde un principio. La ontología utilizada actualmente en el caso de estudio sobre la negociación de energía eléctrica que se esta desarrollando se realizó manualmente. Como trabajo futuro se pretende implementar la ontología desarrollada automáticamente con Protégé y así poder realizar una comparación en cuanto a la eficiencia del paso de mensajes en las conversaciones entre los agentes y por ende la del sistema en sí.

El mercado de energía mayorista en Colombia es un escenario apropiado para la implementación de agentes de software inteligentes, es decir, agentes de software que sean capaces de comportarse de la misma manera en que un agente real lo haría. Por otro lado, la comunicación inherente a un proceso de negociación basado en subastas, en donde pueden participar muchos agentes, hace necesario la implementación de las ontologías, con el fin de brindar una estructura de conocimiento como base de dicha comunicación.

AGRADECIMIENTOS

El trabajo presentado en este artículo fue financiado por el proyecto de investigación de la DIME titulado: “Simulación del Proceso de Negociación de Contratos Normalizados Bilaterales de Energía basado en un Modelo Multi-Agente de Subastas para el Sector Eléctrico Colombiano”, con código 30805914 de la Universidad Nacional de Colombia Sede Medellín.

REFERENCIAS

[1] ISA. Mercado de Energía Mayorista – MEM –. http://www.isa.com.co/, 2006.

[2] CREG. Comisión de Regilación de Energía y Gas. Sistema electrónico de contratos normalizados bilaterales – SEC –. Documento CREG- 005, 2004.

[3] Ovalle, D., Moreno, J., Marulanda, J., y Arias, F. Proyecto de Investigación – Simulación del proceso de negociación de contratos normalizados bilaterales de energía basados en un modelo Multi-Agente de subastas para el sector eléctrico colombiano. Código DIME 30805914 de la Universidad Nacional de Colombia – Sede Medellín, 2005.

[4] Durfee E. H., et. al. , Cooperative Distributed Problem Solving. The Handbook of AI, vol. 4, chp. XVII, A. Barr, P.R. Cohen, E.A. Feigenbaum, eds., Addison-Wesley, Reading, MA, 1989.

[5] Bellifemine, F., Poggi, A., Rimassa, G. JADE – A FIPA compliant agent framework, 1999.

[6] JESS Documentation. The Expert System Shell for the Java Platform, Version 6.1RC1, 2003.

[7] Caire, G., y Cabanillas, D. JADE TUTORIAL. APPLICATION-DEFINED CONTENT LANGUAGES AND ONTOLOGIES, 2004.

[8] Wooldridge, M. An Introduction to Multi-Agent System. Ed. John Wiley & Sons ltd, 2002.

[9] Consorcio W3C. http://www.w3c.es. Consultada el 7/11/2006.

[10] Bellifemine, F., Rimessa, G., Trucco, T., y Caire, G. JADE PROGRAMMER’S GUIDE, 2005.

[11] O’Brien, P. D. Nicol, R. C. FIPA – Towards a standard software agents, 1998.

[12] Protégé documentation. Getting Started With Protégé, 2000.

[13] Protégé documentation. User Guide, 2000. Beangenerator. Universiteit van Amsterdam. http://protege.cim3.net/cgi-bin/wiki.pl? OntologyBeanGenerator. Consultada el 7/11/2006.

[14] Beangenerator. Universiteit van Amsterdam. http://protege.cim3.net/cgi-bin/wiki.pl? OntologyBeanGenerator. Consultada el 7/11/2006.

Francisco Javier Arias Sánchez. Estudiante de último semestre de Ingeniería de Sistemas e Informática de la Facultad de Minas de la Universidad Nacional de Colombia Sede Medellín. Integrante del GIDIA: Grupo de Investigación y Desarrollo en Inteligencia Artificial, Categoría A de Colciencias. Ha trabajado, durante los últimos dos años, como monitor académico de investigación, en los tres proyectos del GIDIA titulados: "Diseño e implementación de un sistema multi-agente que simule el comportamiento del mercado energético en Colombia", "Modelo de Sistema Multi-Agente de Cursos Adaptativos Integrados con Ambientes Colaborativos de Aprendizaje" y "Herramientas informáticas para el diagnóstico automático de eventos en líneas de trasmisión de energía eléctrica”, este último en colaboración con GAUNAL (Grupo de Automática de la Universidad Nacional de Colombia), ISA y cofinanciado por Colciencias.

Julian Moreno Cadavid. Profesional Empresas Públicas de Medellín E.S.P. Ingeniero de sistemas e informática, Universidad Nacional de Colombia - Sede Medellín (2004). Magíster en Ingeniería de sistemas, Universidad Nacional de Colombia Sede Medellín (2007). Integrante del GIDIA: Grupo de Investigación y Desarrollo en Inteligencia Artificial, Categoría A de Colciencias. El área de énfasis de su investigación es la Inteligencia Artificial aplicada a diversos problemas de la industria. Más específicamente ha trabajado en sistemas híbridos que integran Sistemas Expertos, Sistemas Neuro-Difusos, Sistemas Multi-Agente y otras herramientas para el apoyo a la toma de decisiones en áreas como los Mercados de Energía, la Detección de Fallas en Líneas de Transmisión, así como en diversas aplicaciones de simulación y pronóstico.

Demetrio Arturo Ovalle Carranza. Profesor Asociado, Universidad Nacional de Colombia sede Medellín. Director de la Escuela de Ingeniería de Sistemas de la Universidad Nacional de Colombia Sede Medellín. Director del GIDIA: Grupo de Investigación y Desarrollo en Inteligencia Artificial, Categoría A de Colciencias. Ingeniero de Sistemas y Computación, Universidad de los Andes, Bogotá, Colombia (1984). Magíster en Informática del Institut National Polytechnique de Grenoble, Francia (1987). Doctor en Informática de la Université Joseph Fourier, Francia (1991). El área de énfasis de su investigación es Inteligencia Artificial, más específicamente Sistemas Híbridos Inteligentes integrando Redes Neuronales, Sistemas de Lógica Difusa y Sistemas Multi-Agente aplicados a la Simulación de los Mercados de Energía y a la Detección de Fallas en Líneas de

Transmisión. Otros tópicos de investigación que trabaja actualmente son: Inteligencia Artificial en Educación, Sistemas Tutoriales Inteligentes, Sistemas basados en CBR (Case-Based Reasoning) y Técnicas de Planificación Inteligente aplicadas a la Construcción de Sistemas de Composición de Servicios Web.