Desarrollo de Multiagentes para Minería de Datos con JADE

Lic. Karina Mino Polanco
kmino10@hotmail.com

M.C. Rogelio Ferreira Escutia
rferreir@itmorelia.edu.mx

Instituto Tecnológico de Morelia
Maestría en Ciencias en Ciencias de la Computación, Av. Tecnológico 1500, Col. Lomas de Santiaguito, C.P. 58120
Morelia, Michoacán, México

Resumen.- En este trabajo se muestra el desarrollo de multiagentes distribuidos utilizando JADE, que proporciona un conjunto de librerias y herramientas ad hoc para el desarrollo de multiagentes. Los agentes desarrollados utilizan algoritmos de Minería de Datos Descriptiva para encontrar información útil en red y que son programables vía web para permitir la visualización de resultados y la reconfiguración de los parámetros de minería de datos a través de Internet o cualquier dispositivo móvil (como celulares y asistentes personales), de tal manera que el proceso de minería de datos sea más automático, flexible y rápido que el proceso tradicional.

Palabras claves: multiagentes, minería de datos, JADE.

I. INTRODUCCIÓN

La minería de datos es una tecnología reciente, debido al crecimiento de las base de datos, que permite el análisis de información para la toma de decisiones, además del planteamiento y *descubrimiento automático* de hechos e hipótesis, ya sean patrones, reglas, grupos, funciones, modelos, secuencias, relaciones, correlaciones, etc. Esto permite a las empresas la posibilidad de obtener información para la toma de decisiones, de sus grandes bases de datos, que ordinariamente no se obtendría sin utilizar técnicas de minería de datos, haciéndolas más competitivas en el mercado.

Los agentes desarrollados proveen un proceso de minería de datos más rápido y flexible que un proceso de minería convencional, pues un agente en cada máquina realiza un subproceso de minería y la configuración del proceso puede hacerse desde un dispositivo móvil o de una página web.

Al utilizar JAVA y JADE en el desarrollo de los multiagentes se provee una aplicación heterogénea independiente de los sistemas operativos donde se ejecute.

II. MINERÍA DE DATOS

La Minería de Datos, también conocido en inglés como **Data Mining**, es un proceso que, a través del descubrimiento y cuantificación de relaciones predictivas en los datos, permite transformar la información disponible en conocimiento útil de negocio. Esto es debido a que no es suficiente realizar consultas sin rumbo por los datos para resolver los problemas de negocio, sino que se hace necesario seguir una metodología ordenada que permita obtener rendimientos tangibles de este conjunto de herramientas y técnicas de las que dispone el usuario.

Existen dos tipos de Minería de Datos:

1. *Minería Descriptiva*.- Describe los datos en forma concisa y presenta en forma general propiedades interesantes de los datos. Se aplica para descubrir patrones en datos existentes para guiar el proceso de toma de decisiones.
2. *Minería Predictiva*.- Construye un conjunto de modelos, aplicando inferencia sobre los datos disponibles, haciendo intentos para predecir el comportamiento de nuevos conjuntos de datos.

III. AGENTES

Los agentes son sistemas que cuentan con las siguientes propiedades [1]:

- *Movilidad*.- Habilidad del moverse en forma autónoma.
- *Adaptabilidad*.- Puede ajustarse a su entorno, los métodos, hábitos de trabajo y preferencias del usuario.
- *Colaboración.*- Colabora con otros agentes para el cumplimiento de una tarea común. Permite corregir algunos errores causados por su usuario, como omisión de información, información ambigua, etc.

Los Agentes utilizados en la Minería de Datos se encargan de analizar la información para detectar patrones y relaciones, ya de forma automática o interactuando con el usuario analista [2]. Gracias a la flexibilidad, generalidad, posibilidad de distribución y modularidad que poseen los agentes, es posible pensar en construir diferentes tipos agentes que proporcionen una solución eficiente al proceso de minería de datos.

IV. MINERÍA DE DATOS CON AGENTES

Actualmente, la tendencia de las empresas es tener la mayor cantidad de información útil en bases de datos, por lo cual éstas pueden contener una gran cantidad de registros. La Minería de Datos se utiliza generalmente sobre bases de datos históricas gigantescas (contienen información de hace semanas, meses, años, etc.) para obtener información importante para la toma de decisiones que por otros métodos convencionales no se encuentra dado el gran volumen de información.

El problema de realizar la minería de datos sobre bases de datos históricas es que los resultados obtenidos reflejan relaciones pasadas, dado que son de datos de tiempo atrás. Los resultados de una minería de datos son más relevantes y oportunos para una empresa si se realiza la *minería sobre datos actuales* (de hoy, de ayer, de hace pocos días, etc.), de esta manera la toma de decisiones se puede basar sobre información que describe la situación actual de la empresa y no la pasada.

Existe una gran cantidad de herramientas para realizar minería de datos. Muchas de las cuales cubren la mayoría de las técnicas de minería [3], y otras sólo algunas. Estas herramientas en la mayoría de los casos no son de dominio público por lo que se deben de comprar a precios altos.

Otro problema consiste en que el proceso de Minería de datos se desarrolla en forma "manual", es decir el usuario tiene que iniciar el proceso y ver los resultados desde la máquina donde reside el minero, lo cual involucra una dependencia total del minero hacia el usuario.

Por lo tanto, los multiagentes distribuidos desarrollados automatizan y agilizan el proceso de Minería de datos, puedan mostrar resultados y ser programados desde donde se encuentre el usuario a través de una página web, un celular o un asistente personal.

V. ARQUITECTURA PROPUESTA

El proyecto contempla el desarrollo e implementación de multiagentes distribuidos programables vía web para minería de datos descriptiva para una aplicación de mercadotecnia. Se programaron los agentes en lenguaje Java y JADE para hacerlos multiplataforma.

Para la realización de minería de datos descriptiva se utilizó el enfoque de Asociación de Reglas[4], para una aplicación de "análisis de canasta" de una tienda comercial, es decir, se analizan las compras de artículos de los clientes.

A. Aplicación

El análisis de canasta de compras toma su nombre del análisis que se hace de los artículos que un comprador lleva en el momento de la compra en el supermercado o tienda comercial en busca de afinidades para mejorar las ofertas o también para desarrollar las promociones adecuadas [5].

Sin embargo, este análisis se puede realizar en cualquier situación en donde se requiera identificar el tipo de productos o servicios que una persona consume o utiliza. A partir de la identificación de las afinidades se pueden identificar las oportunidades para venta cruzada (venta de artículos de manera conjunta), o para elevar el valor promedio de compra.

Entre los principales beneficios que una tienda comercial obtiene al realizar un análisis de canasta, están:

- Conocimiento del patrón de consumo de sus clientes
- Identificar oportunidades para la venta cruzada
- Desarrollo de ofertas más atractivas
- Ubicación de los productos dentro de la tienda

El resultado del análisis muestra las relaciones entre los artículos comprados en determinado espacio y tiempo.

B. Minería de datos descriptiva

La minería de datos descriptiva se aplica para describir patrones en datos existentes para guiar el proceso de toma de decisiones. Para la realización de este tipo de minería, existen diversos algoritmos [5].

El algoritmo implementado es la asociación de reglas, debido a que es el que se adapta mejor al análisis de canasta [5] que los otros algoritmos por la facilidad de encontrar en los datos reglas para identificar los patrones de comportamiento de las compras de los clientes. Por ejemplo, la regla de asociación Compra_Cerveza $\Rightarrow$ Compra_Pañales [30% confianza, 2% soporte] indica que "el 30% de las transacciones que contienen cerveza también contienen pañales; el 2% de todas las transacciones contienen ambos artículos".

Para cada regla de asociación existen dos conceptos conocidos como el *soporte* y la *confianza*. El *soporte* se define como la probabilidad de que un registro satisfaga tanto a X como a Y. La *confianza* se define como la probabilidad de que un registro satisfaga a Y dado que satisface a X. Para el ejemplo anterior, 30% es el nivel de confianza, y el 2% es el *soporte* de la regla. Por lo tanto, la regla $X \Rightarrow Y$ tiene el *soporte* s en el conjunto de transacciones D si el s% de las transacciones en D contienen $X \cup Y$, y la regla $X \Rightarrow Y$ en el conjunto de transacciones D tiene *confianza* c si el c% de las transacciones en D que contienen X también contienen Y.

C. Multiagentes distribuidos

Los agentes son capaces de realizar el proceso de minería de manera automática, es decir, arrancar el proceso a determinado tiempo de acuerdo con parámetros establecidos (que pueden ser modificados vía web). El ambiente de trabajo entre los agentes (Figura 1) se especifica como una red de computadoras con n sitios sobre las cuales se encuentran bases de datos grandes con una estructura fija y son dinámicas en el

sentido de que su contenido va variando en tiempo real (debido a la utilización propia de la base de datos como inserciones, eliminaciones y modificaciones), existen *n* agentes de procesamiento que se encargan de analizar la información minada de cada sitio y que mantiene comunicación con un agente de monitoreo capaz de detectar cambios en la base de datos para que el agente de procesamiento reinicie el proceso de minería de datos.

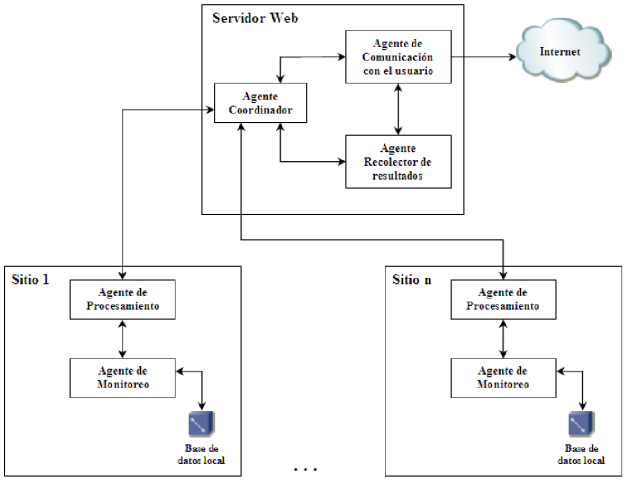

Fig. 1 – Arquitectura de comunicación.

Cada agente de procesamiento comunica sus resultados parciales al agente recolector para que procese el resultado final de minería. El resultado final es enviado al agente de comunicación con el usuario, el cual se encarga de la visualización de los resultados y permite al usuario especificar los parámetros de minería que desee establecer para un nuevo análisis; éstos nuevos parámetros son enviados al agente coordinador para que defina un nuevo proceso de minería de los agentes de procesamiento en los *n* sitios.

La coordinación del sistema completo esta a cargo de el agente coordinador el cual coordina todo el proceso de minería de datos y la comunicación entre los diversos agentes. Al dividir el trabajo entre los agentes existentes en cada sitio el proceso de minería de datos se agiliza. En resumen, las funciones que cada agente debe realizar se ilustran en la Tabla 1.

Tabla 1
Funciones generales de los distintos agentes del sistema.

Agente	Funciones
Agente de monitoreo	Verifica constantemente si hubo cambios significativos en la base de datos local del sitio para reportarlos al agente de procesamiento.
Agente de procesamiento	Con los datos obtenidos de la base de datos, aplica un proceso de minería de datos descriptiva para generar un resultado, el cual es enviado al agente recolector de resultados.
Agente recolector de resultados	Recolecta los resultados de los *n* sitios para generar un resultado general, el cual es enviado al agente de comunicación según lo solicite éste. Para recibir todos los resultados, el agente recolector solicita al agente coordinador que establezca la comunicación con los n sitios y le envíe los resultados, de manera que el agente coordinador funciona como un agente de comunicación entre los demás agentes.
Agente de comunicación con el usuario	Establece la comunicación con el usuario vía web para enviarle a éste los resultados del proceso de minería de datos descriptiva, además de permitirle especificar los nuevos parámetros de minería de datos para realizar un nuevo análisis, los nuevos parámetros son enviados al agente coordinador para que este los comunique a los agentes de procesamiento de cada sitio.
Agente coordinador	Permite la comunicación entre los diferentes agentes del sistema, por lo que es el responsable de comunicar a los *n* sitios los nuevos parámetros del proceso de minería de datos y de coordinar la recopilación de resultados de cada sitio.

Obviamente, se requiere una interfaz para la comunicación usuario-agente que permita programar al agente: especificar los parámetros de minería de datos, patrones que deseen encontrarse, visualización del conocimiento descubierto, entre otras cosas.

La importancia de esta interfaz de comunicación es que permita al usuario comunicarse con el agente principal desde cualquier parte donde éste se encuentre, por lo cual la interfaz se programará en HTML (para la página web) y en WML (para los dispositivos móviles) para que el agente principal pueda ser programado dinámicamente y mostrar resultados a través de paginas web y dispositivos móviles como celulares y asistentes personales (PDA). La Figura 2 muestra esta arquitectura de comunicación con el agente principal.

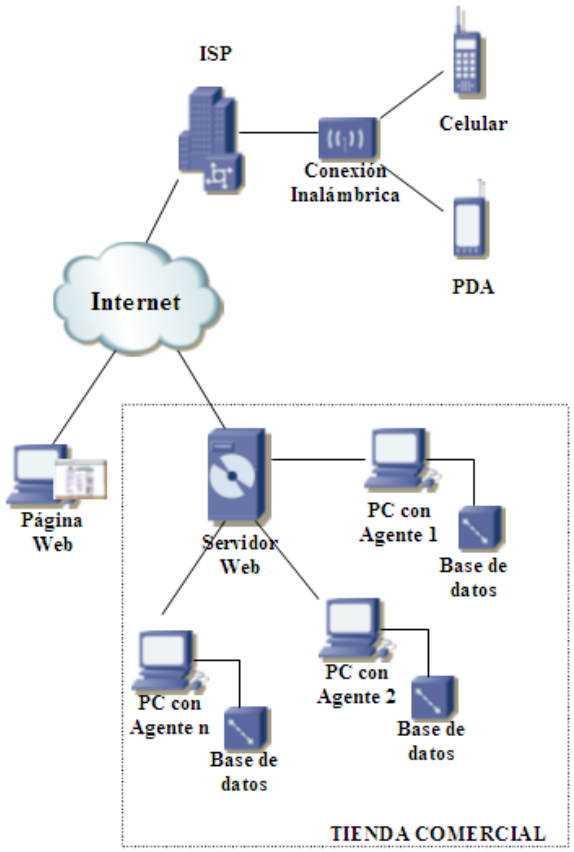

Fig. 2 - Arquitectura de red.

VI. JADE

A. Características

JADE (Java Agent DEvelopment Framework) es un entorno de trabajo que simplifica la construcción de sistemas multiagente que siguen el estándar FIPA[6], así como un conjunto de herramientas para la depuración de las plataformas. JADE está implementado en JAVA y es un software libre y de código abierto, que ha sido desarrollado por el CSELT (Centro Studi e Laboratori Telecomunicación) del grupo Telecom Italia[7].

Dado que JADE y los agentes que el usuario define para una aplicación específica utilizan el lenguaje de desarrollo JAVA, la plataforma de agentes tiene una total independencia de los sistemas operativos empleados.

Entre las características más relevantes de JADE se destaca:

- Provee un conjunto de herramientas para el desarrollo y administración de los agentes distribuidos.
- Facilidad para la movilidad y clonación de agentes entre las máquinas.
- El programador solo debe de preocuparse por la programación de los comportamientos (behaviours) o tareas que el agente debe realizar ya que la comunicación entre ellos es facilitada por JADE.

B. Estructura básica

En un ambiente convencional de agentes con JADE (Figura 3) existe una plataforma que se compone de un conjunto de contenedores (instancias de JADE), y de un contenedor principal especial denominado *Main container*, que se activa al crear la plataforma y al cual todos los demás contenedores deben registrarse tan pronto como son creados.

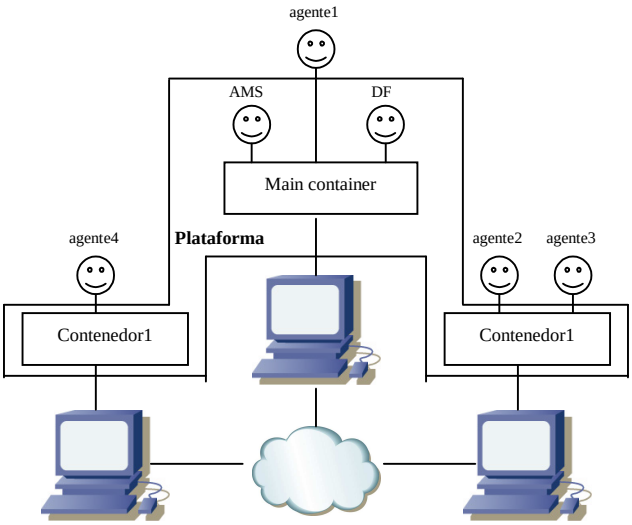

Fig. 3 –Ambiente convencional de JADE.

Dentro de cada contenedor pueden existir agentes, de tal forma que la plataforma de agentes puede estar distribuida entre distintas máquinas (cada una con un contenedor como mínimo) con agentes trabajando entre los contenedores. Es importante la relación entre los contenedores debido a que JADE permite la movilidad y clonación de agentes de un contenedor a otro.

Para el soporte de comunicación entre los agentes, JADE implementa una estructura interna [8] que sigue el estándar FIPA[6].

C. Herramientas para la administración y desarrollo

JADE tiene disponibles un conjunto de herramientas con interfaz gráfica para simplificar la administración y monitoreo de los agentes[9]:

1. **RMA** (Remote Management Agent) para la gestión de la plataforma. Permite crear, eliminar, suspender o enviar un mensaje a un agente, así como la eliminación de contenedores.
2. **Dummy Agent** para la envío de mensajes y visualización de los mensajes enviados o recibidos a un agentes.
3. **Sniffer Agent** para la visualización de la comunicación entre varios agentes.
4. **DF GUI** para controlar el conocimiento de los DFs sobre los agentes con sus servicios registrados, con la posibilidad de formar redes complejas con DFs de otras plataformas.
5. **Introspector Agent** para el monitoreo y control del ciclo de vida de un agente, así como su cola de mensajes y de comportamientos (behaviours).

VII. DESARROLLO DE AGENTES CON JADE

A. Conceptos básicos

Se tiene una clase *Agent* que proporciona la estructura básica del agente, la funcionalidad éste se define en comportamientos llamados *behaviours*. La estructura básica de un agente se define:

```
public class MiAgente extends Agent{
  public void setup(){
    //Instrucciones de inicio
  }
  public void takeDown(){
    //Instrucciones de finalización
  }
}
```

Generalmente en la función setup() se definen las funciones iniciales del agente y los comportamientos con los que inicia. Mientras que la función takeDown() se utiliza para realizar las tareas de finalización como por ejemplo cerrar una base de datos.

B. Comportamientos

Existen varios tipos de comportamientos como son: Behaviour, OneShotBehaviour, CyclicBehaviour,

FSMBehaviour, SequentialBehaviour y ParallelBehaviour, entre otras[8]. Cada tipo de comportamiento se deferencia en los elementos que lo componen y de la forma como se ejecuta, termina, activa, etc.

Para la realización de minería de datos cada **agente de procesamiento** tiene un comportamiento llamado *AsociacióndeReglas* extendido de la classe *Behaviour* que además de realizar el proceso de minería de datos también crea un agente monitor para monitorear los cambios en la base de datos. Para definir el comportamiento se utiliza la función addBehaviour:

```
public class Mineria extends Agent{
  public void setup(){
    ...
    addBehaviour(
        new AsociaciondeReglas(argumentos)
    );
    ...
  }
  ...
}
```

La definición del comportamiento se realiza de la siguiente manera:

```
class AsociaciondeReglas extends Behaviour {
  public void action(){
    //Instrucciones para definir la tarea a realizar
  }
  ...
}
```

Todos los comportamientos definen las instrucciones de la tarea a realizar en su función action().

Para el **agente coordinador** y para el **agente recolector de resultados** se utilizó un comportamiento *CyclicBehaviour* pues este comportamiento se repite indefinidamente; en este caso, el agente coordinador esta revisando su cola de mensajes en espera de que llegue una solicitud de un nuevo proceso de minería de datos y el agente recolector espera la llegada de los resultados de los agentes de procesamiento.

El **agente de monitoreo** requería que cada cierto tiempo se revisara el tamaño la base de datos local, por lo tanto el comportamiento utilizado fue *TickerBehaviour*. En el ejemplo siguiente cada 2 segundos se ejecuta la función onTick().

```
public class AgenteMonitor extends Agent {
  ...
  addBehaviour(new TickerBehaviour(this,2000) {
    public void onTick() {
               //Instrucciones de monitoreo
               //de la base de datos
    }
  });
  ...
}
```

Un ejemplo de aplicación de un comportamiento *SequentialBehaviour* (mostrado en la sección D.Movilidad) se encuentra en el **agente de procesamiento**, el cual obtiene el nombre de los contenedores activos para enviarles un agente clonado con los parámetros de minería a todas las máquinas conectadas. Para hacer esto, utiliza un SequentialBehaviour que se compone de varios subcomportamientos que se ejecutan secuencialmente. Cada subcomportamiento realiza la tarea de clonación del agente.

C. Comunicación entre agentes

El envío y recepción de mensajes entre los agentes es transparente pues JADE se encarga de elegir la vía más adecuada para transportar ese mensaje, así, si los agentes emisor y receptor están ejecutándose sobre la misma máquina virtual de JAVA, utilizará los eventos JAVA, si están en distintas máquinas virtuales, RMI y si están en distintas plataformas, IIOP o HTTP[8].

Un agente puede enviar o recibir un mensaje utilizando los métodos predefinidos[8]. Por ejemplo, el agente de monitoreo envía un mensaje al agente de procesamiento:

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.setContent(ALERTA);
msg.addReceiver(new AID(agproces, AID.ISLOCALNAME));
myAgent.send(msg);
```

D. Movilidad

JADE provee un conjunto de funciones para mover o clonar un agente[8]. Para el proyecto se requería que el agente de procesamiento se clonara todas las máquinas de la plataforma, para que iniciará en ella su proceso de minería de datos. Se utilizó la función doClone() cuyos parámetros son el nombre del contenedor y el nombre del clon. Un agente clonado puede definir tareas antes y después de clonarse con las funciones beforeClone() y afterClone() respectivamente.

```
public class Mineria extends Agent{
  public void setup(){
    addBehaviour(new ObtenerLocalidades());
  }
  protected void beforeClone() { }
  protected void afterClone() { }
  ...
}
```

Dentro de ObtenerLocalidades:

```
SequentialBehaviour sb = new SequentialBehaviour();
for (contador=1; sitios.hasNext(); contador++){
  sb. addSubBehaviour(new OneShotBehaviour(myAgent){
        public void action() {
        ...
         myAgent.doClone(contenedor,nombreclon);
        ...
        }
      });
}
```

E. Creación de agentes en tiempo de ejecución

Cuando se crea al **agente coordinador** este crea sin intervención del usuario y en tiempo de ejecución a dos agentes: el agente recolector de resultados y al agente de comunicación con el usuario, esto lo logra utilizando las funciones de *jade.core.Runtime*.

Por ejemplo, para crear al agente recolector:

```
Runtime rt = Runtime.instance();
AgentContainer ac = getContainerController();
AgentController agente1=
ac.createNewAgent("AgenteRecolector","misprogramas.c
lonacionconmineria.AgenteRecolector",args);
Agente1.start();
```

F. Monitoreo de los agentes

Para monitorear este proceso se utilizó la herramienta de administración RMI (Figura 4), donde se observa gráficamente como al crear al agente coordinador este crea su *agente recolector* y queda en espera de solicitudes de minería de datos.

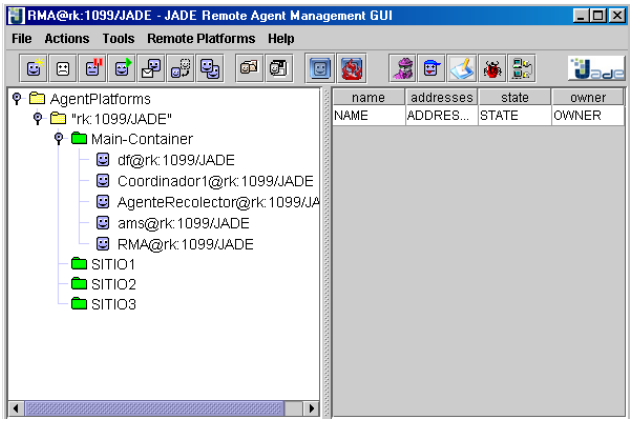

Fig. 4 –Herramienta RMI con los agentes desarrollados.

Al recibir una solicitud el agente de procesamiento (llamado en este caso *Coordinador1*) crea un agente llamado MineroX, donde X es el identificador del proceso de minería (*Minero01*). El *Minero01* es clonado en todos las máquinas disponibles (sitio1, sitio2, etc.) con el nombre de AgenteMineroY, donde Y es el número del clon, y este a su vez crea a su propio *agente monitor* de la base de datos (Figura 5). Una Vez terminado el proceso el agente *Minero01* se elimina a el mismo y a su *agente monitor*.

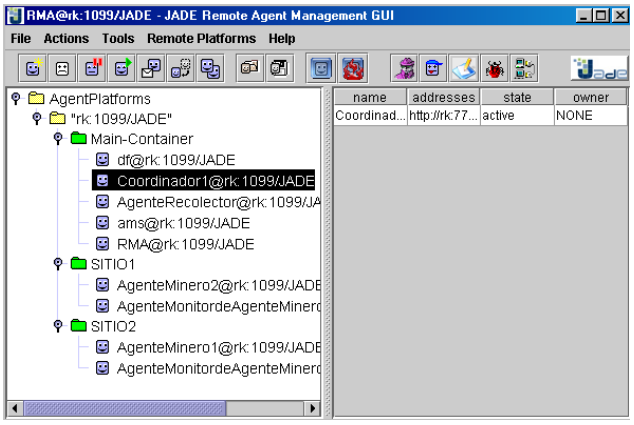

Fig. 5 –Distribución de los agentes creados y clonados en los contenedores.

VIII. CONCLUSIONES Y TRABAJO FUTURO

El desarrollo de agentes utilizando JADE permite al programador preocuparse por sólo definir las tareas que desee que realice el agente, ya que JADE es quién maneja y controla los aspectos internos de la comunicación entre los agentes, siendo una herramienta que facilita su creación.

El proceso de minería de datos se distribuye eficientemente en las diferentes máquinas dentro de una red local, pero aún falta hacer pruebas en máquinas remotas.

El proyecto puede reforzarse incluyendo en los agentes cierta "inteligencia" para que determinen que procesos de minería son más viables de realizar y cuáles no, y aspectos sobre las preferencias en los patrones de búsqueda del usuario. Asimismo, seria factible incluir minería predictiva para predecir las compras futuras de los clientes.

VII. REFERENCIAS

[1] Liu, Jiming. "Autonomous Agents and Mult-Agent Systems: Explorations in Learning, Self-Organization and Adaptive Computation." 2001, Hardcover.

[2] Bigus, Joseph P.; Bigus, Jennifer; Bigus, Joe; "Constructing Intelligent Agents Using Java: Professional Developer´s Guide." 2000, Paperback.

[3] Written, Ian H., Eibe Frank,. "Data Mining: Practical Machine Learning Tools with Java Implementations." 2000, Publishers.

[4] Jean-Marc, Adamo. "Data Mining for Association Rules and Sequential Patterns: Sequential and Parallel Algorithms." 2001, Hardcover.

[5] Jiawei, Han; Kamber, Micheline; Kaufmann, Morgan. "Data Mining: Concepts and Techniques." 2001, Publishers.

[6] Foundation for Intelligent Physical Agents, "Foundation for Intelligent Physical Agents. Specifications." <http://www.fipa.org>, [Consulta Junio 2005]

[7] Telecom Italia Lab, <http://jade.cselt.it/>,[Consulta Junio 2005]

[8] Bellifemine, Fabio; Caire, Giovanni. "JADE Programmer's Guide". 2004. <http://jade.cselt.it/doc/>,[Consulta Junio 2005]

[9] Bellifemine, Fabio; Caire, Giovanni. "JADE Administrator's Guide" 2003.<http://jade.cselt.it/doc/>,[Consulta Junio 2005]

VII. BIOGRAFÍAS

Karina Mino Polanco, nació en el Distrito Federal en 1978. Obtuvo el grado de Licenciada en Informática en el Instituto Tecnológico de Tepic en el año 2001. Actualmente es candidata a obtener el grado de Maestro en Ciencias en Ciencias de la Computación con especialidad en sistemas distribuidos en el Instituto Tecnológico de Morelia.

Rogelio Ferreira Escutia, nació en Morelia, Michoacán en 1971. Obtuvo el grado de Ingeniero en Electrónica en el Instituto Tecnológico de Morelia en 1995 y el grado de Maestro en Ciencias de la Computación en Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Morelos en 1998. Actualmente es Profesor del Departamento de Sistemas y Computación y de la Maestría en Ciencias en Ciencias de la Computación en el Instituto Tecnológico de Morelia.