P-91

Multiagentes Distribuidos Desarrollados con JADE para Minería de Datos

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.- Este trabajo muestra el desarrollo de multiagentes distribuidos utilizando JADE, que proporciona un conjunto de librerías 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 configurables vía web para permitir la visualización de resultados y la configuración 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.

Dado que los agentes fueron desarrollados utilizando Java y JADE, su ejecución es independiente de los sistemas operativos donde se ejecute.

ROC&C'2005 – CP-43 PONENCIA RECOMENDADA POR EL **COMITÉ DE COMPUTACIÓN** DEL **IEEE SECCIÓN MÉXICO** Y PRESENTADA EN LA **REUNIÓN DE OTOÑO, ROC&C'2005**, ACAPULCO, GRO., DEL 29 DE NOVIEMBRE AL 4 DE DICIEMBRE DE 2005.

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 para la toma de decisiones, pues no es suficiente realizar consultas sin rumbo por los datos, sino que es necesario seguir una metodología ordenada.

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

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, tales como: Enterprise Miner, Intelligent Miner, Clementine, Bussiness Miner, SQL Análisis Services, SPSS Data Mining, entre otros.

También existen herramientas de dominio público; entre las que realizan el método de asociación de reglas existen: WEKA, DBMiner, ARMiner recientemente Papyrus, este último es el único que realiza minería de datos distribuida sobre clusters y superclusters [4-8].

Sin embargo, todas las herramientas antes mencionadas (excepto Papyrus) no distribuyen el proceso de minería de datos en varias computadoras, lo que implica que en un servidor tienen una gran base de datos histórica (información de hace semanas, meses, años, etc.) y sobre esta se realiza el proceso de minería de datos. El problema de realizar la minería de datos sobre bases de datos históricas es que los resultados obtenidos reflejan relaciones pasadas, dado que son datos de tiempo atrás. Los resultados de una minería de datos son más relevantes y oportunos para una empresa si se realiza la sobre datos actuales, de esta manera la toma de decisiones se basa sobre información que describe la situación actual de la empresa y no la pasada.

Dado los problemas que presenta realizar el proceso de minería de datos sobre una sola gran base de datos se desarrolló el concepto de minería de datos distribuida. Distribuir el proceso de minería de datos es importante ya que disminuye el tiempo de ejecución del proceso, dado que se realiza el análisis en fragmentos de la base de datos, aprovechando que la mayoría de las empresas tienen sus bases de datos fragmentadas en diferentes computadoras o sitios.

Otro inconveniente de las herramientas anteriores es que el usuario gerencial tiene que iniciar el proceso y ver los resultados desde la máquina donde reside el software de minería, cuando lo más factible es hacer la solicitud de minería y recibir los resultados donde quiera que éste se encuentre.

En la Tabla 1 se muestra una comparación entre las características del proyecto desarrollado y las herramientas de dominio público antes mencionadas [4-8].

Tabla 1
Comparación del proyecto desarrollado con herramientas de dominio público para minería de datos.

Características	**Proyecto desarrollado**	**DBMiner**	**ARMiner**	**WEKA**	**Papyrus**
Distribución del proceso de Minería de Datos	Si	No	No	No	Si
Minería de datos Cliente-Servidor	No	Si	Si	Si	No
Freeware	Si	Si	Si	Si	Si
Multiplataforma	Si	No	Si	Si	No
Interfaz en Web	Si	No	No	No	No
Interfaz en dispositivos móviles	Si	No	No	No	No
Análisis de datos actualizados en tiempo real	Si	No	No	No	No
Análisis de datos históricos	Si	Si	Si	Si	Si

El presente proyecto utiliza multiagentes para distribuir el trabajo en diferentes computadoras sobre cualquier sistema operativo, pueden mostrar resultados y ser programados desde donde se encuentre el usuario a través de una página web, un celular o un asistente personal.

V. JADE

JADE (Java Agent DEvelopment Framework) es un entorno de trabajo que simplifica la construcción de sistemas multiagente que siguen el estándar FIPA [9], 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 [10].

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.

JADE tiene disponibles un conjunto de herramientas con interfaz gráfica para simplificar la administración y monitoreo de los agentes [11]:

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).

VI. ARQUITECTURA DESARROLLADA

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

En la realización de minería de datos descriptiva se utilizó el enfoque de Asociación de Reglas [12], enfocado a 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 [13].

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.

B. Asociación de reglas

El algoritmo implementado es la asociación de reglas, debido a que se adapta mejor al análisis de canasta [13] 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 Articulo_X $\Rightarrow$ Articulo_Y [30% confianza, 2% soporte] indica que "el 30% de las transacciones que contienen el artículo X también contienen el artículo Y; 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 desarrollados

Los agentes desarrollados son capaces de realizar el proceso de minería de manera flexible, es decir, inician el proceso de acuerdo con parámetros configurados vía web. El ambiente de trabajo utilizado entre los agentes se especifica en la Figura 1 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.

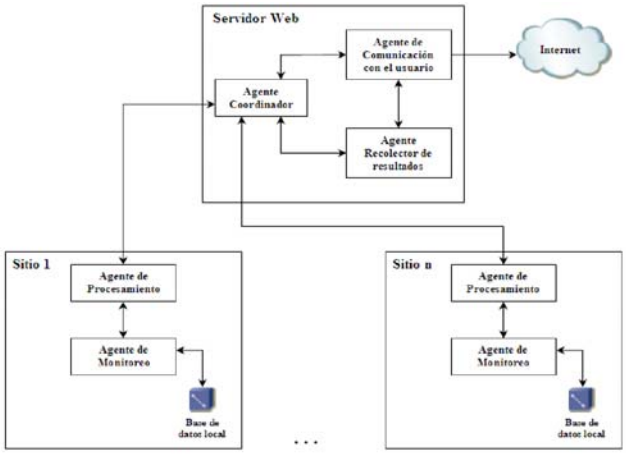


Fig. 1. Arquitectura de comunicación.

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 2.

Tabla 2
Funciones generales de los distintos agentes del sistema.

Agente	Funciones
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.
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 de monitoreo	Verifica constantemente si hubo cambios significativos en la base de datos local del sitio para reportarlos al agente de procesamiento y este reinicie el proceso.
Agente recolector de resultados	Recolecta los resultados de los *n* sitios para generar un resultado general, el cual es enviado al agente de comunicación.
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 genere los agentes de procesamiento 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, y visualización de resultados de cada sitio.

La importancia de esta interfaz de comunicación es que permita al usuario comunicarse con el agente principal desde cualquier parte donde éste se encuentre, 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.

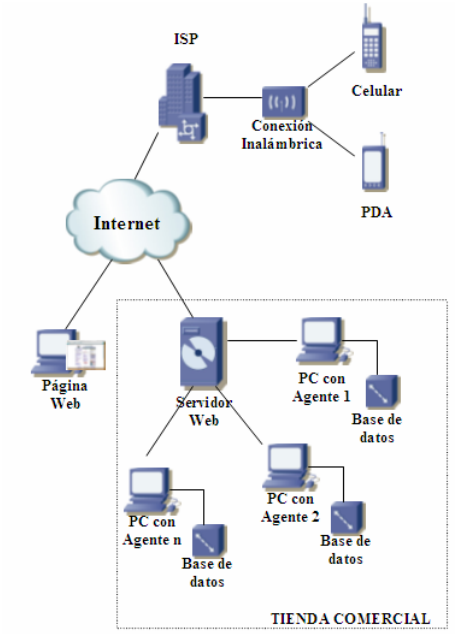
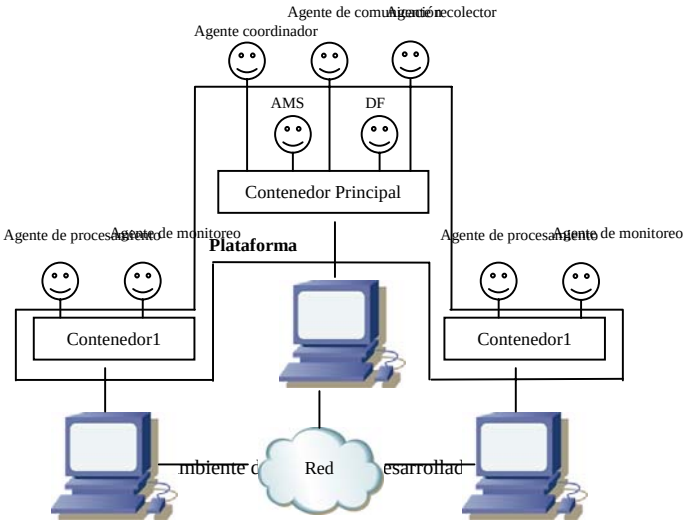


Fig. 2. Arquitectura de red.

VII. MULTIAGENTES PARA MINERÍA DE DATOS DESARROLLADOS CON JADE

En la Figura 3 se muestra el ambiente de los agentes desarrollados con JADE, 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 (ubicados en diferentes máquinas) se registran tan pronto como son creados.

En el proyecto desarrollado, en el contenedor principal esta el agente coordinador, el agente de comunicación con el usuario y el agente recolector de resultados. En cada contenedor secundario se encuentra un agente de Procesamiento y un agente de monitoreo de la base de datos. Esta distribución de los agentes también puede apreciarse en la Figura 1.



Dentro de cada contenedor existen varios 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 [14] que sigue el estándar FIPA [9].

A. Estructura básica de un agente

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
  }
}
```

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 de los agente desarrollados

Existen varios tipos de comportamientos en JADE que sirven como base para implementar tus propios comportamientos, algunos son: Behaviour, OneShotBehaviour, CyclicBehaviour, FSMBehaviour, SequentialBehaviour y ParallelBehaviour, entre otras [8]. Cada tipo de comportamiento se diferencia en los elementos que lo componen y de la forma como se ejecuta, termina, activa, etc.

Para la realización del **agente de procesamiento** se elaboró 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 iniciar el comportamiento se utiliza la función addBehaviour:

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

Se implementó un comportamiento *SequentialBehaviour* en el **agente de procesamiento** para obtener el nombre de los contenedores activos y 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.

Para la elaboración del **agente coordinador** y del **agente recolector de resultados** se utilizó un comportamiento *CyclicBehaviour* pues este comportamiento se repite indefinidamente; en este caso, se determino que el agente coordinador revise constantemente su cola de mensajes en espera de que llegue una solicitud de un nuevo proceso de minería de datos y el agente recolector que espere la llegada de los resultados de los agentes de procesamiento. El siguiente fragmento de código muestra el uso del comportamiento *CyclicBehaviour* para el agente coodinador:

```
addBehaviour(new CyclicBehaviour(this) {
  public void action() {
    //Escucha si llega un mensaje INFORM
     ACLMessage msg =receive (MessageTemplate.
     MatchPerformative(ACLMessage.INFORM));
     if (msg != null) {
      ContentElement p=manager.extractContent(msg);
      ...
     }
  }
  });
  ...
}
```

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
    }
  });
  ...
}
```

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[14].

Los agentes elaborados envían y reciben mensajes mensaje utilizando los métodos predefinidos[14]. Por ejemplo, el agente de monitoreo envía un mensaje al agente de procesamiento de la siguiente manera:

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

D. Clonación del agente de procesamiento

JADE provee un conjunto de funciones para mover o clonar un agente [14]. Para el proyecto se requería que el agente de procesamiento se clonara en 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() { }
  ...
}
```

Donde Obtener Localidades se desarrolló como sigue:

```
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= c.createNewAgent
("AgenteRecolector","AgenteRecolector",args);
agente1.start();
```

F. Monitoreo de los agentes

Para monitorear este proceso se utilizó la herramienta de administración RMI mostrada en la Figura 4, donde se observa gráficamente que al crear al agente coordinador este crea automáticamente y sin intervención del usuario dos agentes: el agente recolector y el agente de comunicación con el usuario y queda en espera de solicitudes de minería de datos.

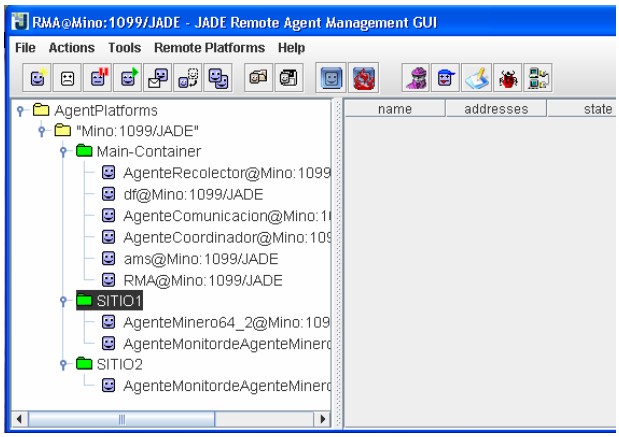


Fig. 4. Distribución de los agentes creados y clonados en los contenedores.

Al recibir una solicitud el agente de procesamiento (llamado en este caso *AgenteCoordinador*) crea un agente llamado AgenteMineroX, donde X es el identificador del proceso de minería (*AgenteMinero64*). El *AgenteMinero64* es clonado en todos las máquinas disponibles (sitio1, sitio2, etc.) con el nombre de AgenteMineroX_Y, donde Y es el número del clon, y este a su vez crea a su propio agente monitor de la base de datos (AgenteMonitordeAgenteMineroX_Y). Una Vez terminado el proceso el agente *AgenteMinero64* se elimina a el mismo y a su agente monitor.

VIII. INTERFAZ CON EL USUARIO VÍA WEB

Para la configuración del proceso de minería de datos se desarrolló una interfaz de comunicación vía web con dirección http://minominero.no-ip.info. Se programó en HTML y en WML (para los dispositivos móviles como teléfonos celulares y asistentes personales), de tal forma que el usuario puede :

1. Iniciar un nuevo proceso de minería de datos, especificando los parámetros, artículos de interés y la antigüedad de los datos a analizar.
2. Obtener el resultado del proceso iniciado, así como los sitios involucrados.
3. Obtener los resultados parciales de los *n* sitios involucrados en el análisis.
4. Monitorear que procesos ya terminaron o están en ejecución.
5. Revisar los parámetros de minería de un proceso.

IX. CONCLUSIONES

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 presente proyecto provee una herramienta para minería de datos distribuida de dominio público, donde puede probarse las ventajas que ofrecen los agentes y su programación con una herramienta para el desarrollo de multiagentes.

X. REFERENCIAS

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

[2] Bigus J.P., J. Bigus, J. Bigus. *Constructing Intelligent Agents Using Java: Professional Developer´s Guide*. 2000.

[3] Written I.H., F. Eibe. *Data Mining: Practical Machine Learning Tools with Java Implementations*. 2000.

[4] Written I.H., F. Eibe. *Data Mining: Practical Machine Learning Tools and Techniques*. 2005.

[5] Jiawei Han, Y. Fu, J. Chiang. *DBMiner: A System por Mining Knowledge in Large Relational Database.* 1997.

[6] Bailey S., R. Grossman, H. Sivakumar. *Papyrus: A System por Data over Local and Wide Area Clusters and Superclusters.* 1999

[7] Britos P., R. García. *Selección de Herramientas de Explotación de Datos*. 2004.

[8] Hernández J.M., J.A. Botia, A. Gómez. *Asistencia Personalizada a la Minería de Datos con Agentes Inteligentes.* 2004.

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

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

[11] Bellifemine Fabio; G. Caire. *JADE Administrator's Guide*. 2003.

[12] Jean-Marc Adamo. *Data Mining for Association Rules and Sequential Patterns: Sequential and Parallel Algorithms*.2001.

[13] Jiawei Han, M. Kamber. *Data Mining: Concepts and Techniques*. 2001.

[14] Bellifemine Fabio; G. Caire. *JADE Programmer's Guide*. 2004.

XI. 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.