17

DISEÑO DE UN SISTEMA MULTI-AGENTE PARA MONITOREO DE REDES UTILIZANDO JADE Y JPCAP*

Por: Alexis de la Hoz Manotas**

Fecha de recibido: 2 de julio de 2010 • Fecha de aceptación: 30 de septiembre de 2010

RESUMEN:

La programación orientada a agentes es un paradigma que ha tomado especial dedicación en los últimos años, en especial en investigaciones, desde sistemas de supervisión y control de redes de computadores, simulación de procesos industriales, recopilación de información, sistemas de oferta y demanda, etc.

El diseño de un sistema multiagente (MAS) requiere esfuerzos diferentes a los encontrados en un desarrollo de software tradicional, diversas estrategias como BDI, GAIA y MESSAGE pueden ser utilizadas como modelo teórico para tal fin, mas sin embargo, con los avances en la inteligencia artificial, los agentes inteligentes están ahora en capacidad de demostrar mucha mayor autonomía y aprendizaje que hasta ahora.

Se ha desarrollado un primer intento en esbozar lo que sería el diseño de un sistema multiagente para la monitorización de redes, escogiendo como plataforma de desarrollo a JADE y JPCAP como herramientas de captura.

PALABRAS CLAVE:

Agentes inteligentes, JADE, Sistemas multiagente, JPCAP.

Revista Inge-CUC / Vol. 6 - No. 6 / Octubre 2010 / Barranquilla - Colombia / ISSN 0122-6517

CUC
CORPORACION UNIVERSITARIA DE LA COSTA

* Construcción de soluciones con programación orientada a agentes utilizando la plataforma JADE. Ingeniería del Software y Redes. Ingeniería del Software.

** Especialista en Estudios Pedagógicos. Docente Medio Tiempo. adelahoz6@cuc.edu.co

CUC

MULTI-AGENT SYSTEM DESIGN FOR NETWORK MONITORING USING JADE AND JPCAP

By: Alexis de la Hoz Manotas

ABSTRACT:

Agent oriented programming is a paradigm that has taken a strong dedication in these last years, especially in research, passing from network control and management systems, industrial process simulations, information search, offer and demand systems, etc.

The design of a multiagent system (MAS) demands requirements different from those found in traditional software development, several strategies as BDI, GAIA and MESSAGE can be used as a model theory for it, nevertheless, with advances in artificial intelligence, intelligent agents are becoming capable of having more autonomy and self learning than before.

It has been developed a first attempt to sketch a design for a network monitoring multiagent system, based on JADE platform and JPCAP as capture packet tool.

KEY WORDS:

Intelligent agent, JADE, Multiagent systems, JPCAP.

CUC
CORPORACION UNIVERSITARIA DE LA COSTA

EDUCOSTA
EDITORIAL UNIVERSITARIA DE LA COSTA

INTRODUCCIÓN

Para comenzar es importante aclarar lo que es un agente inteligente, pues de allí se deriva toda la programación orientada a los mismos y es la base de los sistemas multiagente.

Un agente es esencialmente un componente especial de software que tiene autonomía, que provee una interface interactiva con un sistema cualquiera y/o se comporta como un agente humano, trabajando para algunos clientes con sus propios objetivos. Incluso si un sistema de agentes pudiera basarse en un único agente trabajando en un entorno y es necesario para interactuar con sus usuarios, usualmente están formados por múltiples agentes. Estos sistemas multiagente pueden modelar sistemas complejos e introducir la posibilidad de agentes con tareas comunes o conflictivas. Pueden interactuar el uno con el otro indirectamente (a través del entorno) o directamente (comunicación y negociación). Los agentes pueden decidir cooperar para beneficio mutuo o competir para servir sus propios intereses.

Algunas características de los agentes son:

- Un agente es autónomo, porque opera sin la intervención directa de humanos u otros y tiene control sobre sus acciones y su estado interno.
- Un agente es social, porque coopera con humanos u otros agentes para lograr sus tareas.
- Un agente es reactivo, porque percibe su entorno y responde de acuerdo a los cambios que suceden en él.
- Un agente es proactivo, porque no solo responde a su entorno, sino que también es capaz de exhibir comportamiento orientado a objetivos tomando iniciativa.
- Un agente, si lo necesita, puede ser móvil, con la habilidad de viajar entre diferentes nodos en una red de computadoras.
- Un agente es sincero, brindando la certeza que no comunicará deliberadamente información falsa.
- Un agente es benevolente, siempre intentando desarrollar lo que le fue solicitado.
- Un agente es racional, actuando siempre con miras a obtener sus metas y nunca evitando que se cumplan.
- Un agente puede aprender, adaptándose para encajar en su ambiente y a los deseos de sus usuarios.

SISTEMAS MULTIAGENTE

Un sistema multiagente es aquel compuesto por múltiples agentes inteligentes interactuando entre sí.

Los sistemas multiagente están siendo usados en una amplia variedad de aplicaciones, desde sistemas pequeños para asistencia personal hasta sistemas de misión crítica complejos para aplicaciones industriales.

Ejemplos de aplicaciones de sistemas multiagente en ambientes industriales incluyen control de procesos, diagnóstico de sistemas, manufactura, logística de transporte y administración de redes.

Uno de los campos de aplicación más importantes es la administración de información. En particular, Internet se ha mostrado como un dominio ideal para sistemas multiagente debido a su naturaleza distribuida intrínseca y el volumen de información disponible. Los agentes pueden ser usados, por ejemplo, para buscar y filtrar esa masa de información. Internet también ha promovido el uso de tecnologías de agentes en la administración de procesos de negocios y comercio. De hecho, antes de la propagación del comercio en Internet, este proceso era totalmente dirigido por interacciones humanas: humanos decidiendo dónde comprar sus bienes, cuánto están dispuestos a pagar y así sucesivamente. Ahora con el comercio electrónico y los procesos automatizados de negocios aparecen con un rol relevante en muchas organizaciones porque ofrecen oportunidades para mejorar la

forma en que las diferentes entidades involucradas en el proceso interactúan.

Transporte y tráfico es también un campo importante, donde la naturaleza distribuida del tráfico y los procesos de transporte y la fuerte dependencia entre las entidades involucradas en tales procesos, hacen a los sistemas multiagente una herramienta valiosa para construir soluciones comerciales efectivas.

Los sistemas de telecomunicaciones son otro campo de aplicación donde los sistemas multiagente han sido usados con éxito. De hecho, los sistemas de telecomunicaciones son grandes redes distribuidas con componentes interconectados que necesitan ser monitoreados y administrados en tiempo real, y forman un mercado competitivo donde las compañías de telecomunicaciones y los proveedores de servicio buscan distinguirse del resto de competidores proporcionando servicios mucho mejores, más rápidos y confiables.

Debido a la complejidad de los sistemas multiagente, es recomendable utilizar una plataforma de desarrollo que proporciona características básicas de comunicación y administración de agentes como ZEUS y JADE. En nuestro caso, utilizamos esta última, a continuación algunos apartes de la plataforma.

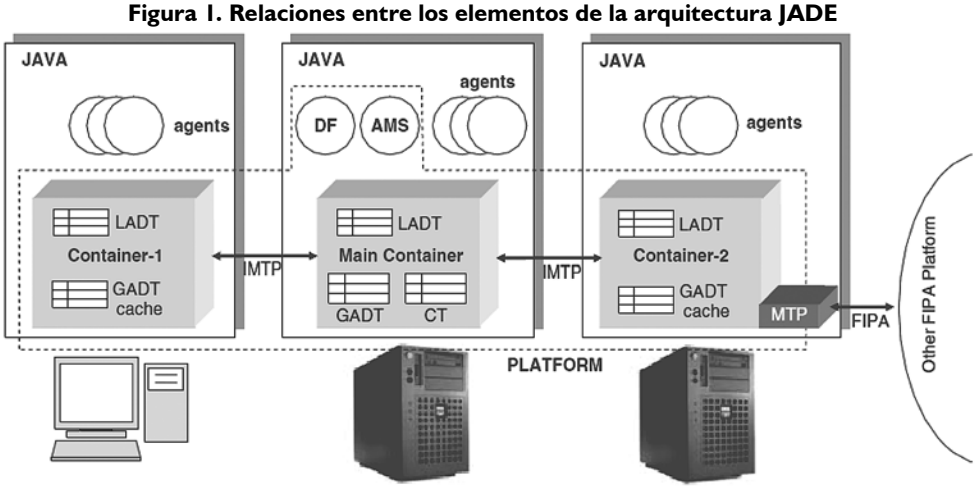
LA PLATAFORMA DE DESARROLLO JADE

Una plataforma JADE[1] está conformada por contenedores de agentes que pueden encontrarse distribuidos por toda la red. Los agentes viven en los contenedores, los cuales son los procesos Java que proporcionan la ejecución de JADE y todos los servicios necesarios para el alojamiento y ejecución de los agentes.

Existe un contenedor especial, denominado el contenedor principal, el cual representa el punto de inicio de una plataforma: es el primer contenedor que se construye y todos los demás contenedores deben unirse a él, a través de un registro.

El programador identifica los contenedores utilizando su nombre lógico, por defecto el contenedor principal es nombrado *Main Container* mientras que los demás *Container-1*, *Container-2*, etc.

Figura 1. Relaciones entre los elementos de la arquitectura JADE



1. (Bellifemine, Caire & Greenwood, 2007).

El contenedor principal tiene unas responsabilidades específicas:

- Administrar la tabla de contenedores, la cual es el registro de las referencias de objetos y las direcciones de transporte de todos los nodos contenedor que contiene la plataforma.
- Administrar la tabla global descriptora de agentes (GADT), la cual es el registro de todos los agentes presentes en la plataforma, incluyendo su estado actual y ubicación.
- Hospedar el AMS y el DF, los dos agentes especiales que proporcionan la administración de agentes y el servicio de páginas blancas y el servicio de páginas amarillas por defecto, de la plataforma.

Aunque el contenedor principal puede ser un punto de falla hacia el interior de la plataforma, existen soluciones como el Servicio de Replicación Principal, que le permite a la plataforma permanecer totalmente operacional en caso de una falla en el contenedor principal. Con este servicio el administrador puede controlar el nivel de tolerancia a fallos, el nivel de escalabilidad y el nivel de distribución de la plataforma.

Una capa de control compuesta por varias instancias distribuidas del contenedor principal puede ser configurada para implementar un sistema de arranque y un sistema de control distribuido. En casos extremos, cada contenedor puede ser programado para unirse al servicio de replicación principal y actuar como parte de la capa de control.

La identidad de un agente está contenida en su identificador de agente (AID), compuesto por un conjunto de espacios que satisfacen la estructura y la semántica definida por FIPA. Los elementos básicos del AID son el nombre del agente y su dirección. El nombre de un agente es un identificador global único que JADE construye al concatenar un nombre proporcionado por el usuario (también conocido como el nombre local) al nombre de la plataforma.

Las direcciones de los agentes son direcciones de transporte heredadas por la plataforma, donde cada dirección corresponde a un punto MTP (*Message Transport Protocol*), donde mensajes compatibles con FIPA pueden ser enviados y recibidos.

Los programadores de agentes pueden adicionar sus propias direcciones de transporte al AID cuando por algún motivo en especial, deseen implementar su propio MTP privado.

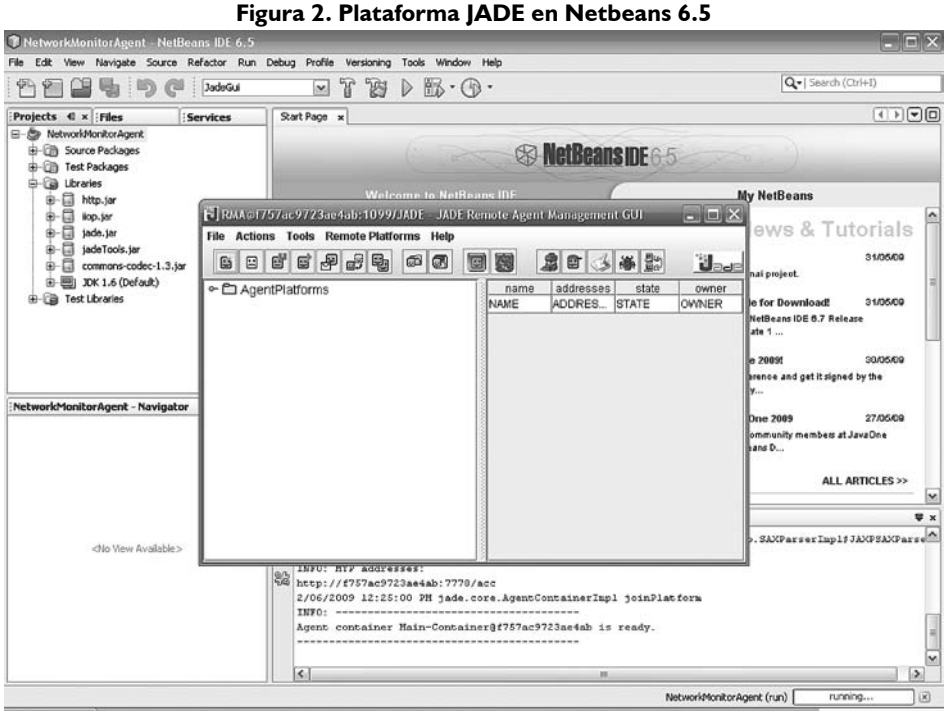
Cuando se ejecuta el contenedor principal, dos agentes especiales se crean automáticamente y se inicia por parte de JADE, cuyos roles están definidos por el estándar de administración de agentes FIPA:

- El agente de administración de sistema (AMS) es el agente que supervisa toda la plataforma. Es el punto de contacto para todos los agentes que necesitan interactuar con las páginas blancas de la plataforma así como administrar su ciclo de vida. Cada agente debe registrarse con el AMS para obtener un AID válido.
- El facilitador de directorio (DF) es el agente que implementa el servicio de páginas amarillas, usado por cualquier agente que desee registrar sus servicios o buscar otros servicios disponibles. El DF también acepta suscripciones de agentes que desean ser notificados cuando un registro de servicio o modificación ha sido realizado que cumple con algunas condiciones específicas. Múltiples DFs pueden iniciarse de forma concurrente para distribuir el servicio de páginas amarillas a través de varios dominios. Estos DFs pueden estar federados, si se necesita, estableciendo registros entre ellos lo que permite la propagación de las peticiones de los agentes a través de la federación.

DISEÑO DEL PROTOTIPO

Para la aplicación de la programación orientada a agentes, se decidió elaborar el diseño de un sistema multiagente prototipo y desarrollar un agente prototipo del sistema.

Figura 2. Plataforma JADE en Netbeans 6.5

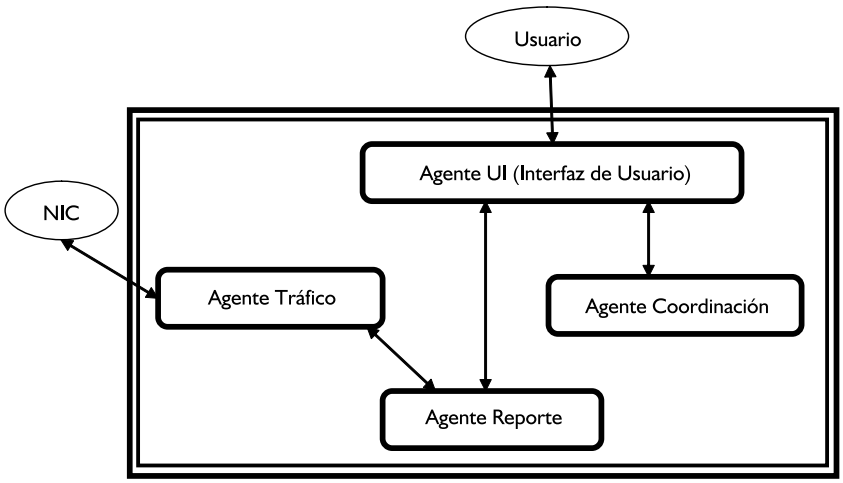


Teniendo en cuenta la alta aplicabilidad de los agentes en redes de información, se propone un diseño de un sistema multiagente para el monitoreo de la actividad de los clientes en una LAN, el cual presenta el siguiente diseño preliminar:

En la figura 3, se aprecian cuatro tipos de agentes distintos que se detallan a continuación:

Agente UI (Interfaz de Usuario)
Es el encargado de interactuar con el usuario de forma directa, proporciona una visión del sistema en general y su configuración actual. Se comunica con el agente de reportes para poder desplegar para el usuario gráficas y estadísticas recopiladas por los agentes tráfico. Se recomienda que el agente UI tenga facilidades de presentación en Web, para ampliar su acceso.

Figura 3. Estructura Sistema MAS Prototipo



Agente Reporte
Es el encargado de recibir los resúmenes de estadísticas de los agentes tráfico y organizarlas para su envío posterior al agente UI.

Agente Tráfico
Es el encargado de captar los paquetes que pasan por una tarjeta de red (NIC) y realizar periódicamente un resumen estadístico para enviarlo al agente reporte.

Se recomienda definir reglas de configuración para el filtrado de paquetes específicos o hacer sugerencias en base al flujo de tráfico detectado.

Agente Coordinación
Es el encargado de realizar la coordinación y control de los agentes tráfico en los diferentes equipos de la red, identifica su ubicación y recomienda una planificación para la mejor distribución de los mismos.

Para el agente prototipo, se eligió el Agente Tráfico, que está diseñado para captar el tráfico de una tarjeta de red específica, aunque sin el módulo de IA que podría aumentar sus capacidades y hacer sugerencias en cuanto a prevención de problemas y posibles soluciones.

IMPLEMENTACIÓN DEL PROTOTIPO

```
public class MonitorAgent extends Agent{
 @Override
    protected void setup() {
        // Printout a welcome message
        System.out.println("My local-name is " + getAID().getLocalName());
        System.out.println("My GUID is " + getAID().getName());
        System.out.println("My addresses are:");
        Iterator it = getAID().getAllAddresses();
        while (it.hasNext()) {
            System.out.println("- " + it.next());
        }
        //Define behaviour
        addBehaviour(new PacketPrinterBehaviour());
    }
}
```

Código Fuente 1. Clase MonitorAgent

El agente monitor, como todos los agentes extiende de la clase Agent, e implementa un comportamiento cíclico (PacketPrinterBehaviour) que realiza la captura de los paquetes.

```
public class PacketPrinterBehaviour extends CyclicBehaviour {

    JpcapCaptor captor; //Capturador de Paquetes
    final int MIN_NUM_PACKETS = 100; //Limite de paquetes para estructuras
    ArrayList<Packet> listaPaquetesActual; //Estructura 1
    ArrayList<Packet> listaPaquetesAnterior; //Estructura 2


    /* Constructor */
    public PacketPrinterBehaviour() {
        //Define la networkInterface a analizar
```

```
        this.captor = PacketPrinterThread.abrirNetworkInterface();
        //Construye instancia de ArrayList inicial
        listaPaquetesActual = new ArrayList<Packet>(MIN _ NUM _ PACKETS);
        listaPaquetesAnterior = new ArrayList<Packet>(MIN _ NUM _ PACKETS);
    }



@Override
    public void action() {
        Packet packetReceived = captor.getPacket();
        if (packetReceived != null) {
            System.out.println(packetReceived);
            if (listaPaquetesActual.size() == MIN _ NUM _ PACKETS) {
                listaPaquetesAnterior = listaPaquetesActual;
                listaPaquetesActual = new ArrayList<Packet>(MIN _ NUM _ PACKETS);
            }
            else {
                listaPaquetesActual.add(packetReceived);
            }
        }
    }
}
```

Código Fuente 2. Clase PacketPrinterBehaviour

El comportamiento PacketPrinterBehaviour, introduce la capacidad del agente Monitor de capturar los paquetes a través del objeto captor.

La creación de este objeto se hace a través del método abrirNetworkInterface:

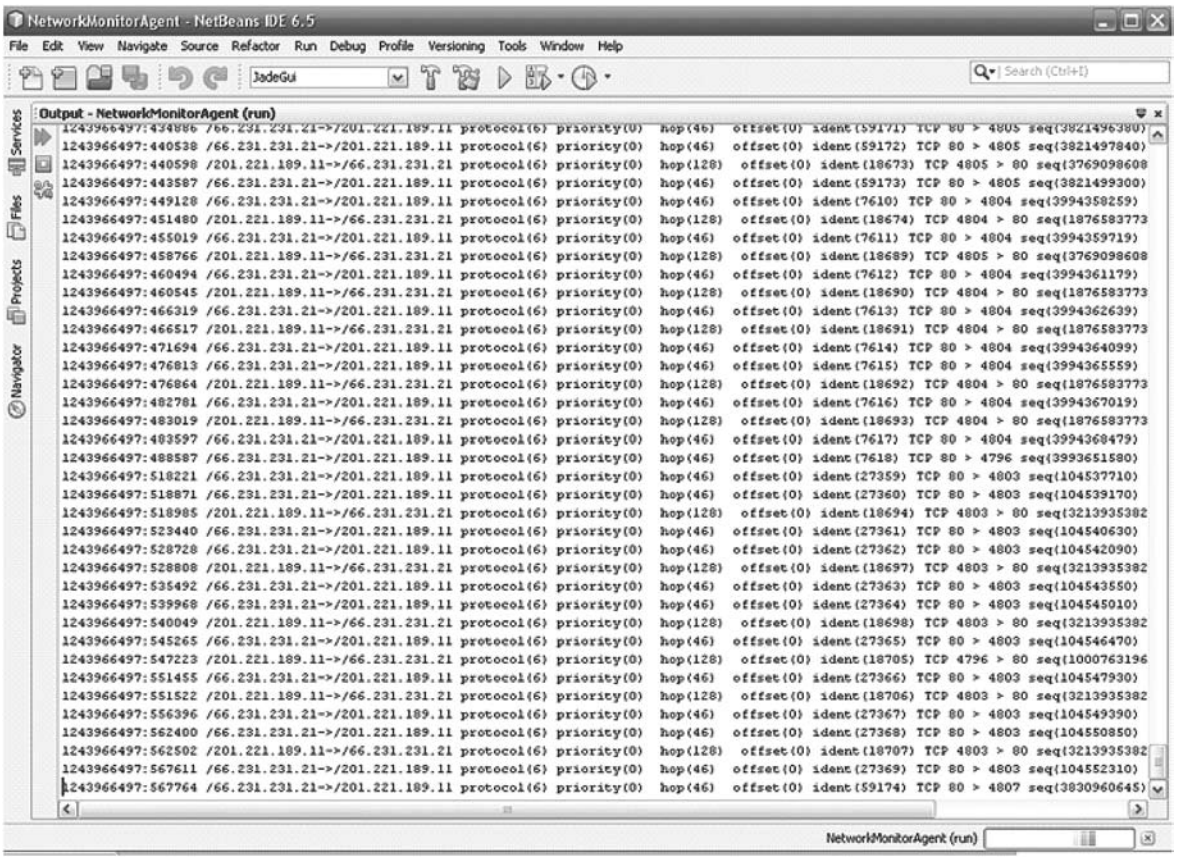
```
public static JpcapCaptor abrirNetworkInterface() {
        NetworkInterface[] devices = JpcapCaptor.getDeviceList();
        int index =0;  // set index of the interface that you want to open.
        try {
            // set index of the interface that you want to open.
            //Open an interface with openDevice(NetworkInterface intrface, int
snaplen, boolean promics, int to _ ms)
            JpcapCaptor captor = JpcapCaptor.openDevice(devices[index], 65535,
false, 20);
            return captor;
        } catch (IOException ex) {
            ex.printStackTrace();
            return null;
        }
```

Código Fuente 3. Método abrirNetworkInterface de la clase PrinterPacketThread

Una vez creado el capturador de paquetes, el método *action* repite la tarea de capturar paquetes, y almacenarlos en un objeto ArrayList para su uso posterior.

Por facilidad de envío de mensajes, se decidió definir un límite inicial de 100 paquetes, una vez alcanzada esta cifra, se hace una copia y se regenera la estructura para almacenar los siguientes 100 paquetes.

Figura 4. Salida por Consola del Agente Monitor



CONCLUSIONES

La programación orientada a agentes se destaca como una fuerte área de desarrollo, que aunque no soluciona todos los problemas, es una excelente solución en sistemas grandes y distribuidos.

El diseño de sistemas multiagente no es una tarea rápida y que se debe tomar a la ligera, implica una ingeniería de software diferente a la tradicional y aunque existan metodologías propuestas para ello, aún no existe un estándar acogido ampliamente como en el caso del proceso unificado en programación orientada a objetos.

La inteligencia artificial cuenta con muchos métodos desarrollados o en investigación, es aún un área exploratoria en muchos aspectos, promoverla es parte de nuestra obligación como profesionales en las ciencias de la computación y los agentes inteligentes son una herramienta idónea por sus múltiples ventajas y la facilidad que proporcionan las plataformas de desarrollo existentes en el mercado.

BIBLIOGRAFÍA

BELLIFEMINE, F.; CAIRE, G. & GREENWOOD, D. (2007). *Developing Multi-Agent Systems with Jade*. Wiley.

BRESCIANI, P.; PERINI, A.; GIORGINI, P.; GIUNCHIGLIA, F. & MYLOPOULOS, J. (2001). A Knowledge Level Software Engineering Methodology for Agent Oriented Programming. *AGENTS'01*.

GÓMEZ SANZ, J. J. (2003). Metodologías para el desarrollo de sistemas multi-agente. *Revista Iberoamericana de Inteligencia Artificial*, 51-63.

HORFAN ÁLVAREZ, D.; BAILEY, A. M. & GÓMEZ BLANDON, L. A. (2005). Sistemas de seguridad en redes locales utilizando sistemas multiagentes distribuidos Net-Mass. *Facultad de Ingeniería*, 101-113.

HUNTBACH, M. & RINGWOOD, G. (1999). *Agent-oriented Programming: From Prolog to Guarded Definite Clauses*. Springer.

JENNINGS, N. R. & WOOLDRIDGE, M. (2000). Agent-Oriented Software Engineering. *Artificial Intelligence*, 277-296.

PEÑA DE CARRILLO, C. I. (2000). Sistemas Multiagente para el Tratamiento de la información en la Web: Agentes de interfaz, agentes de información, agentes de aprendizaje, agentes intermediarios2. *I Congreso Internacional de Ingeniería de Sistemas*, (p. 24). Bucaramanga.

Wikipedia-Multi-agent System (s.f.). Recuperado el 2009, de http://en.wikipedia.org/wiki/Multi-agent_system

Wikipedia-Software Agent (s.f.). Recuperado el 2009, de http://en.wikipedia.org/wiki/Software_agent