EXPERIENCIA EN EL DESARROLLO DE UN SISTEMA MULTIAGENTE

Ángel Alonso Álvarez
Escuela de Ingenierías (Universidad de León), Campus de Vegazana s/n, 24071 León, [dieaaa@unileon.es]

José Ramón Villar Flecha
Escuela de Ingenierías (Universidad de León), Campus de Vegazana s/n, 24071 León, [diejvf@unileon.es]

Carmen Benavides Cuellar
Escuela de Ingenierías (Universidad de León), Campus de Vegazana s/n, 24071 León, [diecbc@unileon.es]

Isaías García Rodríguez
Escuela de Ingenierías (Universidad de León), Campus de Vegazana s/n, 24071 León, [dieigr@unileon.es]

Francisco Jesús Rodríguez Sedano
Escuela de Ingenierías (Universidad de León), Campus de Vegazana s/n, 24071 León, [diefrs@unileon.es]

Resumen

Los sistemas multiagente se vienen utilizando en muy variadas áreas de investigación y desarrollo. En este artículo se pretende ofrecer la experiencia del grupo investigador con dichos sistemas en el curso de un proyecto de investigación aplicada a una empresa para resolver un problema específico. Partiendo de conocimientos teóricos, se describen las diferentes decisiones tomadas en el diseño del sistema y finalmente la plataforma de agentes elegida y en uso. Finalmente, se describe la arquitectura planteada para la solución al problema y trabajos futuros.

Palabras Clave: Sistemas Multiagente, Sistemas Inteligentes de Soporte, Inteligencia Artificial Distribuida

1 INTRODUCCIÓN

El conocimiento, tanto explícito como tácito, se ha convertido en un elemento crucial en la creación de valor en las organizaciones, su uso y distribución son decisivos en la toma de decisiones [19].

La explosión en la cantidad de información disponible en formato electrónico, requiere sistemas de gestión de la información, que la transformen en conocimiento disponible en tiempo real y que puedan ser distribuidos sobre una red e interoperar con otros sistemas [3], [16]. Estos sistemas no se pueden implementar con la tecnología de software tradicional debido a los límites de esta tecnología en conseguir distribución e interoperabilidad. Las tecnologías basadas en agentes parecen ser una prometedora respuesta para facilitar la realización de estos sistemas ya que han sido creadas para hacer frente a la distribución y la interoperabilidad [2], [3], [9].

Con estas premisas, y ante el problema real que nos ha planteado una empresa que posee una gran cantidad de información tanto en papel (con intención de digitalizarlo) como en formato electrónico, se optó porque el sistema de gestión del conocimiento, que se va a describir en el siguiente apartado, se implemente mediante la tecnología de Sistemas Multiagente, lo que permitirá obtener un entorno de información descentralizado, abierto y modular. Los diversos agentes realizarán, entre otras, las funciones de almacenar, recuperar, gestionar y compaginar información de distintas fuentes distribuidas [15]. Utilizando la tecnología de Sistemas Multiagente se consigue un aumento de eficiencia ante operaciones de filtrado y recuperación de la información al trabajar los agentes de forma asíncrona y paralela. Por otra parte, la escalabilidad y flexibilidad de la aplicación es más sencilla de implementar, al implicar únicamente un aumento en el número de agentes en funcionamiento. Asimismo, la robustez y fiabilidad del conjunto serán mayores debido a la redundancia inherente a este tipo de sistemas [5], [23].

La comunicación entre los distintos agentes del sistema se basará en una concepción común y acotada del dominio de la aplicación, convenientemente especificada en una ontología.
Una vez que la aplicación planteada haya sido totalmente desarrollada se realizará la experimentación in situ para medir la eficacia del sistema, pretendiendo obtener datos sobre la

adecuación de la estructura propuesta al problema a resolver, así como los posibles beneficios para los empleados y la empresa.

2 DESCRIPCIÓN DEL PROYECTO

El sistema multiagente de cuya experiencia en el desarrollo trata este trabajo forma parte del proyecto de investigación financiado por la Junta de Castilla y León, con expediente LE038/UA. El objetivo de este proyecto es implementar las tareas de gestión del sistema documental de la Fundación Sociedad Anónima Hullero Vasco-Leonesa (en adelante, la *empresa*). Se plantea el uso de las técnicas de Ingeniería y Gestión del Conocimiento, todo ello para que la aplicación resultado de este proyecto no represente un cambio significativo en el espíritu marcado por la filosofía de trabajo de la empresa, pero sí en los rendimientos obtenidos.

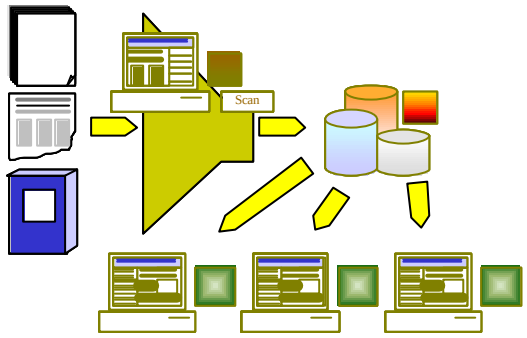

Figura 1. Descripción gráfica del proyecto

El sistema documental de la empresa es responsabilidad del departamento documentalista. Las labores de este departamento abarcan la recopilación de todo tipo de documento (monografías, revistas, artículos de revistas y prensa, ...), su etiquetado y posterior almacenado. Asimismo, es responsabilidad del departamento la edición de un boletín de prensa, de un dossier de medioambiente y de la distribución selectiva de la información. Finalmente, se encarga de resolver toda consulta documental que le sea remitida. Se asume que es posible la modificación de la estructura y contenido de los documentos editados en cualquier momento. En la figura 1 se puede observar una descripción del sistema a resolver.

La resolución del proyecto se basa en la aplicación de inteligencia artificial distribuida mediante tecnología de sistemas multiagentes [9], [15], [23]. Los agentes inteligentes implementados se basan en el modelo Creencias-Deseos-Intenciones (BDI, Beliefs-Desires-Intentions) [6], [10], [18] y en estructura multicapa. El uso de un sistema de agentes se decidió debido a la alta modularidad de las tareas realizadas por el departamento documental y además por su variabilidad, lo que hacía oportuno asimilar tareas a agentes y que éstos variasen su comportamiento según las necesidades del momento.

En la figura 2 se puede ver una descripción del sistema de agentes planteado. Puede observarse un sistema compuesto por diversos agentes. La labor de los mismos puede resumirse como sigue. Se dispone de dos tipos de interfases: el interfaz de escaneado y reconocimiento óptico de caracteres (SOI) - que representa la entrada de documentos al sistema documental - y el interfaz web, compuesto por un agente de gestión del personal de la empresa con acceso al sistema documental (GESPER), el agente de gestión del tesauro (GESTE), el agente de interfaz de consulta (ICON) y el agente de gestión de monografías (GEMA). Por otro lado, se dispone de un agente que analiza las consultas (TRACON), un agente analizador morfológico (STEMMING), un agente ETIQUETADOR, un agente de interfaz con base de datos (DAI), un agente utilidad general (GENERARBOL), un agente por cada tipo de documento a editar (BOP – boletín de Prensa, DOMA – dossier de medioambiente, DISE - difusión selectiva de información). Mas los agentes base de la plataforma JADE (AMS, DF, ...). Estos son los diferentes tipos de agentes, pudiendo haber más de una instancia con vida en cada momento de cada uno de ellos excepto del AMS, único por plataforma (especificaciones FIPA).

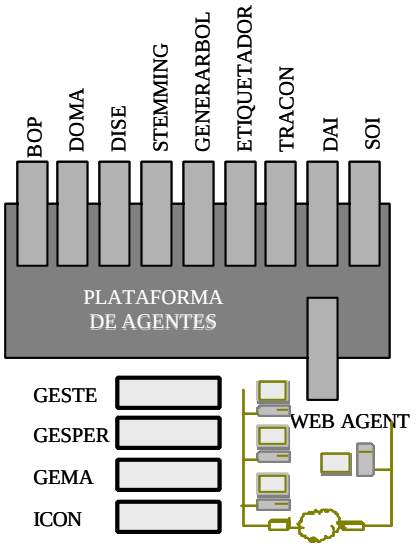

Figura 2. Sistema de Agentes planteado

Como WA – Web Agent – se quiere significar cada uno de los agentes interfaz web, que disponen de páginas web de acceso mas el código correspondiente al agente inteligente incluido en la plataforma.

La labor de cada uno de estos agentes se puede presuponer de la relación dada anteriormente, si bien,

es necesario llamar la atención sobre algunos aspectos. Primero, es necesario advertir que todo agente encargado de la edición de un documento representa instancias diferentes del mismo agente generador de documento maestro, ya que en la conceptualización de esta tarea se concluyó que todo documento a editar es un mismo problema caracterizado por atributos con valores distintos (índice, título, etc.) y reglas de asignación de documentos a cada ítem de índice o a partes del documento diferentes.

El segundo aspecto a incidir es que el SOI es un interfaz que incluye código nativo en C++ (a diferencia de la plataforma, desarrollada en java) y que requiere la existencia de un escáner instalado en la máquina para desarrollar todo su potencial. Un tercer aspecto es la inclusión de sistemas expertos como agentes: el STEMMING o analizador morfológico es un ejemplo. Finalmente, otro aspecto a valorar es la problemática del ETIQUETADOR y del TRACON, agentes que contienen bloques inteligentes para adaptar el lenguaje del experto etiquetador al lenguaje del sistema o como sistema inteligente de mejora de consulta en función de los perfiles de usuario del sistema.

3 EVOLUCIÓN DE LA PLATAFORMA DE AGENTES

Antes de entrar a describir los pasos realizados sobre la plataforma de agentes conviene visualizar cuál era el objetivo inicial de la aplicación. Originalmente el enfoque dado al proyecto estaba basado en la Ingeniería y Gestión del Conocimiento para resolver las tareas cubiertas en el mismo.

Con ese fin se desarrolló una intensa labor de Ingeniería del Conocimiento, la metodología utilizada fue la IDEAL [1], [12]. Tras la extracción de conocimiento a partir del historial de la empresa se sucedieron sesiones de educción con los expertos de la empresa. El resultado de esta metodología secuencial e iterativa se puede resumir en los siguientes puntos:

a) Se modeló el comportamiento de los expertos de catalogación de textos, así como de los documentos de conocimiento que formaban parte de los objetivos del proyecto.

b) Se modeló la aplicación que soportase con la suficiente versatilidad las necesidades del mapa de conocimiento del dominio.

Para la formalización se pensó en la técnica de los marcos [14]. Para la fase de implementación se optó por la utilización de una plataforma de agentes sobre JAVA© [5], [15]. La conclusión de la conveniencia dicha implementación se basó, entre otras, en las siguientes ideas:

a) Las tareas a realizar se podían modelar como entes que debían ser capaces de decidir por sí mismas llevar a cabo una serie de acciones, es decir, eran autónomas. Además, definidas sus metas con el mecanismo oportuno eran capaces de dirigirse a resolver el problema para el que estaban pensadas.

b) Al mismo tiempo, necesitaban prestar atención a su entorno para reaccionar adecuadamente, eran reactivos.

c) Para el desarrollo de sus tareas debían comunicarse con el resto de componentes del sistema (sociales).

La arquitectura de agentes que se decidió implementar es la de Creencias-Deseos-Intenciones (CDI, Belief-Desires-Intentions).

3.1 DECISIONES SOBRE LOS AGENTES Y SU ENTORNO

Una vez definido todo lo anterior, el siguiente paso fue acotar los posibles atributos de las propiedades del entorno, tanto de cada agente como del sistema de agentes. En cuanto al entorno de un agente, se concluyó:

a) Que se debía utilizar incertidumbre en la valoración de diversos parámetros importantes, como podían ser los intereses de un determinado perfil de usuario, de un departamento, etc.

b) Que por la misma razón, la predicibilidad no era cierta.

c) Que el entorno era controlable, es decir, que se podía realizar un control por medio de aprendizaje automático de los parámetros no predecibles en su totalidad.

d) En determinados atributos se debía, pues, utilizar experiencias pasadas para tomar decisiones en cada instante.

e) Que el entorno no se debe considerar como de tiempo real.

En cuanto al entorno del sistema de agentes hubo atributos que fueron cambiados en las diferentes versiones del sistema, mientras que otros han

quedado fijos. Como características fijas a lo largo del desarrollo caben destacar

a) Infraestructura orientada a mensajes, con uso de conexiones punto a punto y operaciones asíncronas.

b) Uso de servicios de directorio, tanto páginas amarillas como blancas.

Otros parámetros que definen el entorno del sistema de agentes se indicarán a continuación.

3.2 ETAPAS DE DESARROLLO DEL SISTEMA DE AGENTES

Se pueden diferenciar tres etapas claras en el desarrollo del sistema de agentes, las cuales conducen del sistema de agentes inicial al uso de una plataforma de agentes sobre el cual se implementa el sistema de agentes propuesto.

Tras el diseño de la primera etapa las características del sistema, además de las comentadas en el punto anterior, se pueden resumir en los siguientes puntos:

a) Cada agente lleva a cabo una tarea de forma autónoma.

b) La comunicación entre los agentes se realiza utilizando componentes de JAVA, en concreto, Remote Method Invocation (RMI) [22].

c) El lenguaje de comunicación a utilizar es KQML [7], [13].

d) La ontología se implementa implícitamente en código.

e) Los protocolos de interacción se basan en Contract Net [20] y el protocolo a tres vías para la gestión del sistema.

f) Se utilizan varios servidores de páginas amarillas y páginas blancas, todos ellos coordinados.

Este planteamiento, que puede observarse en la figura 3, disponía de una versatilidad muy alta, sin embargo adolecía de una complejidad de gestión que era difícil de resolver adecuadamente. Por otra parte, disponer de varios servidores de páginas blancas resultó ser un problema añadido que obligó a rediseñar el sistema.

En la segunda etapa de diseño se planteó la eliminación de la multiplicidad de agentes páginas blancas. Este cambio condujo a simplificar algunos elementos de gestión, con lo que el sistema resultaba más robusto. Así mismo, se añadió un agente que permitiese clonar agentes y/o servicios, además de la eliminación de recursos sin utilizar en la aplicación. En cuanto a las demás características de comunicación y protocolos, éstas se mantuvieron sin cambio.

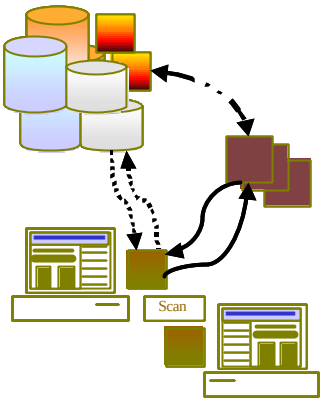

Figura 3. Primera versión del sistema de agentes

A partir de este momento, siguiendo en la fase de diseño, aparecieron dudas sobre la adecuación del protocolo de los mensajes, KQML. Se comprobaron las dolencias que en algunos foros se constataba sobre la no continuidad de este lenguaje, así como la cantidad de dialectos basados en KQML. Esta fue razón suficiente que hizo dar un vuelco en el diseño, abriendo las puertas a otras especificaciones.

Entre otras, se discutió la viabilidad de nuestra aplicación según las especificaciones FIPA (Foundation of Intelligent Physical Agents) [8]. Con ese fin el grupo de investigación estudió todas las especificaciones de las plataformas FIPA y su impacto en el desarrollo del proyecto en vigor.

Conclusión de este trabajo fue la migración del sistema de agentes a una plataforma FIPA a costa de dejar el estudio de diseño de sistemas de agentes para pasar a implementar sobre una plataforma según dicha especificación. Para concluir comentar que se probaron todas las plataformas FIPA accesibles desde el sitio web FIPA y el grupo se decantó por la plataforma JADE de CSelt, de Italia [4].

3.3 CARACTERÍSTICAS DE JADE

JADE (Java Agent Development) es una estructura software que simplifica el desarrollo de sistemas multiagente de acuerdo con las especificaciones FIPA [8] para la interoperabilidad de sistemas de agentes inteligentes [2], [3], [4].

Es un software libre y de código abierto, que ha sido desarrollado por el CSELT (Centro Studi e Laboratori Telecomunicación) del grupo Telecom Italia, en parte dentro del proyecto de investigación europeo ACTS AC17 "FACTS".

Tanto JADE, como los agentes que el usuario define para una aplicación específica, utilizan el lenguaje de desarrollo JAVA [21], lo que aporta a la plataforma una total independencia del sistema o sistemas operativos empleados.

La plataforma de agentes puede estar distribuida entre distintas máquinas y su configuración puede ser cambiada en cualquier momento, ya que JADE permite la movilidad y clonación de agentes de una máquina a otra.

JADE, siguiendo las especificaciones FIPA [8], implementa aquellos aspectos de un sistema multiagente, que no son particularidades internas del agente y son independientes del tipo de aplicación. Con esta premisa, JADE nos proporciona la siguiente estructura base (figura 4):

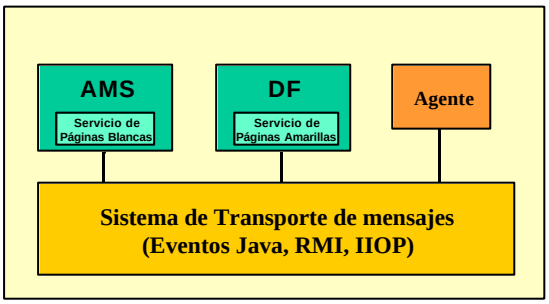

Figura 4. Estructura básica de JADE

Gestión de los agentes, que consiste fundamentalmente en la identificación, registro, ubicación, y estado de los agentes, así como en el registro de los servicios específicos que realizan los agentes de la aplicación. Los dos tipos de agentes encargados de realizar estas funciones son:

- *AMS (Agent Management System)* es el agente que ejerce el control de supervisión sobre el acceso y uso de la plataforma de agentes. Proporciona, entre otros, el servicio de páginas blancas y ciclo de vida de los agentes, manteniendo el directorio de identificador de agentes y el estado de los agentes.

- *DF(Directory Facilitator)* es el agente que proporciona el servicio de páginas amarillas a otros agentes.

De acuerdo con las especificaciones FIPA, los agentes AMS y DF se comunican usando el lenguaje FIPA-SL0 y la ontología fipa-agent-management, ambos también están implementados en JADE.

El Sistema de Transporte de Mensajes, también llamado *ACC (Agent Communication Channel)* es el componente software que controla todo el intercambio de mensajes dentro de la plataforma, incluyendo los mensajes hacia y desde plataformas remotas. El sistema es totalmente transparente al usuario, cuando un agente quiere enviar un mensaje basta con utilizar los métodos predefinidos y es JADE quien automáticamente selecciona 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.

La *clase Agent*, contiene las estructuras básicas para que los agentes realicen interacciones esenciales con la plataforma (registro, configuración, administración, etc.) y un conjunto básico de métodos para personalizar las tareas del agente (envío y recepción de mensajes, protocolos de interacción estándar para la construcción de las conversaciones entre agentes, etc.)

Con todo esto, el diseñador de la aplicación sólo tiene que centrarse en determinar las características particulares de los agentes (objetivos, tareas a realizar) para la aplicación específica que se está desarrollando, esto se compone de dos partes esenciales:

- Definición de las Tareas específicas de los agentes, para desarrollar un agente el programador parte de la *clase* base, *Agent* . Cada agente JADE esta compuesto de un único hilo de ejecución y cada tarea, funcionalidad, servicio o intención de un agente debe ser implementado mediante uno o varios *comportamientos*, que son unidades lógicas de actividad que pueden ser formadas de diversos modos (cíclicas, secuenciales, deterministas, no deterministas, ...) para alcanzar complejos patrones de ejecución. JADE, de forma automática y transparente para el programador, gestiona los comportamientos, soportando la ejecución concurrente y paralela de múltiples tareas mediante una política de round-robin.

- Definición de las ontología u ontologías específicas de la aplicación, donde se engloben todos los conceptos, acciones,

predicados y proposiciones que definen el dominio de la aplicación.

Por consiguiente, utilizar JADE como plataforma de partida, reduce drásticamente el esfuerzo que supone el diseño y desarrollo de un sistema multiagente.

Por otra parte JADE es de fácil instalación y utilización, asimismo proporciona una serie de herramientas que simplifican la administración y desarrollo de la aplicación, como son:

RMA (Remote Monitoring Agent), es una consola gráfica para la administración y control de la plataforma. Permite explorar el servicio de páginas blancas tanto de la propia plataforma como de plataformas remotas, controlar el ciclo de vida de los agentes (creación remota, migración, etc).

Dummy Agent, es una herramienta para la depuración del programa, permite enviar, recibir, cargar y almacenar mensajes ACL (Language Communication Agent).

Sniffer Agent, permite capturar y visualizar los mensajes que se intercambian los agentes.

DF GUI , permite explorar el servicio de páginas amarillas que proporcionan los agentes DF (Director Facilitator), así como gestionar complejos dominios y subdominios de DF's, páginas amarillas.

Socket Proxy Agent, es un agente que actua como un puente entre la plataforma JADE y una conexión TCP/IP.

4 ESTADO ACTUAL Y TRABAJOS FUTUROS

Actualmente el sistema está en fase de finalización del desarrollo, con lo que no es posible incluir resultados de esta experiencia en cuanto a algoritmos utilizados ni a respuesta de la plataforma y su evaluación. Sin embargo, la robustez de la solución FIPA y de la plataforma JADE no han generado ningún tipo duda sobre el buen funcionamiento final.

En cuanto a trabajos futuros caben destacar la inclusión de agentes meta-conocimiento sobre los sistemas gestores de base de datos disponibles para la gestión de múltiples sistemas documentales no conexos, dotar de migración y movilidad a los agentes de consulta, la integración de nuestro sistema con otras soluciones ya desarrolladas y el estudio e implantación de seguridad en sistemas de agentes móviles, temas de interés para el grupo de investigación.

Referencias

[1] Alonso, F.; Juristo, N.; Maté, L.; Pazos, J, "Software Engineering and Knowledge Engineering: Towards a Common Life Cycle", The Journal of Software and Systems, 1996

[2] Bellifemine, F.; Poggi, A.; Rimassa, G.; Turci, P.; "An Object Oriented Framework to Realize Agent Systems", Proceedings of WOA 2000 Workshop, Parma, May 2000, pp. 52-57.

[3] Bellifemine, F.; Poggi, A.; Rimassa, Y.; "JADE– A FIPA-compliant agent framework", CSELT internal technical report. Part of this report has been also published in Proceedings of PAAM'99, London, April 1999, pp.97-108.

[4] Bellifemine, F.; Tiziana, G.; Rimaza, G.; "JADE programmer's guide" [en línea] <http://sharon.cselt.it/projects/jade/>, [Consulta 28 may 2001]

[5] Bigus, J.P.; Bigus, J.; "Constructing Intelligent Agents with Java", Wiley Computer Publishing, USA, 1997

[6] Bratman, M. E., Israel, D. J., Pollack, M. E., Plans and resource-bounded practical reasoning. Computational Intelligence, Vol 4, 1998, pp 349-355

[7] Finin, T.; Labrou, Y.; Mayfield, J.; "KQML as an Agent Communication Language", Software Agents, MIT Press, 1997, pp 291-316.

[8] Foundation for Intelligent Physical Agents, "Foundation for Intelligent Physical Agents. Specifications. 1997". [en línea] <http://www.fipa.org>, [Consulta 28 may 2001]

[9] Genesereth, M. R., and Ketchpel, S. P., "Software Agents," Communication of the ACM, Vol. 37, No. 7 July 1994.

[10] Georgeff, M. P., Lansky, A. L., Reactive reasoning and planning. Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI 87), Seatle, WA, 1987, pp 677-682

[11] Jones, S.; Willet, K.; Willet, P.;1997, "Readings in Information Retrieval", San Francisco: Morgan Kaufmann, ISBN 1-55860-454-4.

[12] Juristo, N., Pazos, J. "Towards a Joint Life Cycle for Software and Knowledge Engineering", Knowledge Oriented Software Design, Ed. J. Cuena, North Holland, Amsterdam, Holanda, 1993

[13] Labrou, Y.; Finin, T.; "A Proposal for a New KQML Specification" Technical eport CS-97-03, University of Maryland Baltimore County, Febrero 1997.

[14] Minsky, M. "A Framework for Representing Knowledge", en Winston, P. ed., "The Psychology of Computer Vision", McGraw-Hill, New York, 1975, pp. 211-280

[15] Nwana, H. S.; "Software Agents: An Overview", The Knowledge Engineering Review 11 (3), 1996

[16] Oates, T.; Nagendra, M.; Lesser, V.; “Networked Information Retrieval as Distributed Problem Solving” Proceedings of CIKM Workshop on Intelligent Information Agents held in conjunction with the Third International Conference on Information and Knowledge Management (CIKM'94), December 1994.

[17] Porter, M.F., 1980, “An algorithm for suffix stripping”, *Program*, 14, 3:130-137.

[18] Rao, A. S.; Georgeff, M. P.; "{BDI}-agents: from theory to practice", Proceedings of the First Intl. Conference on Multiagent Systems, San Francisco, 1995.

[19] Rivero, S.; “Gestión del Conocimiento: una vía hacia la ventaja competitiva” SONITEC

[20] Smith, R. G.; “The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver”, IEEE Transaction on Computers, Vol G29, No. 12, Diciembre 1980, pp 1104-1113.

[21] Sun Microsystems, Inc, “Java TM 2 SDK, Standard Edition Documentation, 2000”. [en línea] <http://java.sun.com/products/jdk/1.3/docs/>, [Consulta 28 may 2001]

[22] Sun Microsystems, Inc., “Java Remote Method Invocation (RMI) Spec.” [en línea] <http://java.sun.com/products/jdk/1.1/docs/guide/rmi/-spec/rmiTOC.doc.html> [Consulta 28 may. 2001]

[23] Weiss, G. ed.; “Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence”, MIT Press, London, England, 1999