risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Revista Ibérica de Sistemas y Tecnologías de Información

Recebido / Recibido: 12/10/2011
Aceitação / Aceptación: 09/12/2011

SCODA para el Desarrollo de Sistemas Multiagente

Jesús A. Román[1], Dante I. Tapia [2], Juan M. Corchado [2]

zjarg@usal.es, dantetapia@usal.es, corchado@usal.es

[1] E. P. S. de Zamora (Universidad de Salamanca), 49022, Zamora, España.

[2] Universidad de Salamanca, 37008, Salamanca, España.

Resumen: El planteamiento de implementar sistemas multiagente de forma que puedan reutilizarse con distintos objetivos, dota a los desarrolladores de una metodología más eficaz en sus fines. Este artículo presenta SCODA, una arquitectura basada en cinco principios: estandaridad, especialización, facilidad de implementación, reutilización y computación distribuida. SCODA pretende facilitar el desarrollo de sistemas multiagente, basándose en el concepto de pequeños grupos de agentes denominados Comunidades Inteligentes Especializadas (CIE). Las CIE cuentan con funcionalidades específicas que permiten llevar a cabo la implementación de sistemas multiagente de forma escalable, dentro del marco de SCODA, pudiendo ser reutilizadas en diversos desarrollos.

Palabras clave: Agentes, Sistemas, Comunidades, Especialización, Reutilización.

Abstract: The approach to develop multi-agent systems that they can be reused for different purposes, provides to developers an effective methodology in their aims. In this paper is presented SCODA, an architecture based on five principles: standarization, specialization, facility of implementation, reuse and distributed computing. SCODA has intended to facilitate the development of multi-agent systems relying on the concept of small groups of agents, known as Intelligent Specialized Communities (CIE). The CIE have specific features that allow to carry out the scalable implementation of multi-agent systems, within the framework of SCODA and can be reused in several developments.

Key words: Agents, Systems, Communities, Specialization, Reuse.

1. Introducción

En general, las personas u organizaciones crean la capacidad necesaria para solventar los problemas que puedan producirse en un momento determinado, lo cuál es fácil pensar que existan otras entidades que dispongan ya de estas capacidades (Jensen, 1992). Este hecho, traducido a los sistemas computacionales, lo encontramos en la tecnología Orientada a Objetos (Elrad et al., 2001; Rashid et al., 2003; Gay et al., 2010; Langtangen, 2011), donde los objetos son encapsulados de forma independiente y

pueden ser reutilizados en diferentes desarrollos con finalidades muy distintas. El planteamiento de llevar a cabo este enfoque a los sistemas multiagente es, precisamente, uno de los objetivos de SCODA, en el cual se busca una arquitectura basada en *Comunidades Inteligentes Especializadas* (CIE) que puedan ejecutarse de forma independiente y tengan la capacidad de colaborar atómicamente entre ellas.

SCODA *(Specialized COmmunities for Distributed multiAgent systems)* es una arquitectura que se centra en el desarrollo de sistemas multiagente basándose para ello en cinco principios: estandaridad, especialización, facilidad de implementación, reutilización y computación distribuida. SCODA integra una o más Comunidades Inteligentes dotadas de especialización, que denominaremos **Comunidades Inteligentes Especializadas (CIE)**. Las Comunidades Inteligentes Especializadas tienen la capacidad de funcionar como un sistema multiagente independiente y especializado, donde los servicios que ofrece son ejecutados de forma distribuida, pudiendo colaborar con otras Comunidades Inteligentes Especializadas en la consecución de objetivos, que de forma individual no puedan alcanzar. Esta filosofía se basa en las *"Redes Empresariales"* (Sonquist y Koenig, 1975; Galaskiewicz, 1979; Borgatti y Foster, 2003; Viedma, 2004; Pöyhönen y Smedlund, 2004; Ibarra et al., 2005; Eraydin y Armatli-Köroglu, 2005; Cabus y Vanhaverbeke, 2006) a través de las cuales, un conjunto de grupos, instituciones u organizaciones interactúan entre ellos para obtener unos resultados favorables tanto a las unidades individuales como a la red en su conjunto.

SCODA puede ser implementada sobre cualquier plataforma para sistemas multiagente que soporte agentes BDI (Belief-Desire-Intention), los cuales tienen asociados una serie de estados mentales, como son creencias, deseos e intenciones (Bratman, 1988; Rao y Georgeff, 1995; Winikoff, 2009; Winikoff y Cranefield, 2010) mediante los que se implementa una capacidad de razonamiento que los hace idóneos para su integración en las Comunidades Inteligentes Especializadas. La implementación de SCODA se ha realizado sobre JADEX (Pokahr et al., 2003; 2007; Pokahr y Braubach, 2009), una extensión de la plataforma JADE (Bellifemine, 1999; Bellifemine et al., 2007) para agentes BDI, debido a que se considera como un motor de razonamiento y puede ser ejecutado de forma independiente, lo que implica una ventaja si se tiene en cuenta el enfoque distribuido que se pretende conseguir.

El presente artículo se estructura de la siguiente forma: En la sección 2 se describe la arquitectura SCODA, incidiendo en los aspectos más relevantes de la misma. La sección 3 describe la plataforma de agentes que integran SCODA. Finalmente, en la sección 4 se presentan las conclusiones obtenidas.

2. Descripción de SCODA

La mayor parte de los sistemas multiagente que se desarrollan tienen un diseño cerrado, es decir, ningún componente o agente externo tiene la capacidad de entrar a participar en el sistema, lo que implica que en la fase de diseño se ha de conocer el conjunto de los agentes que participan y las interacciones que se producen entre ellos (Zambonelli et al. 2000). En el desarrollo de sistemas abiertos la complejidad en cuanto a su diseño es mucho mayor, sobretodo en torno a las comunicaciones, ya que el diseñador no conoce a priori que tipos de componentes van a ser incluidos en el

sistema y por lo tanto la forma de comunicarse (Gonzalez-Palacios y Luck 2007). Es por ello que SCODA surge como alternativa intermedia entre estos dos enfoques, permitiendo la escalabilidad de Comunidades Inteligentes Especializadas, en tiempo de ejecución, para la consecución de objetivos complejos, que de forma individual no se alcanzarían.

La arquitectura SCODA se basa en cinco principios a través de los que se pretende una mayor eficiencia de los sistemas multiagentes desarrollados con la misma:

Estandaridad: A través de este principio se busca que a partir de SCODA, las diferentes Comunidades Inteligentes Especializadas que la conforman tengan una misma estructura que sea independiente de la finalidad que se persiga en el sistema multiagente que la implemente, es decir, la implementación de la estructura de una Comunidad Inteligente Especializada que tenga asociados determinados servicios ha de ser la misma que si los servicios fueran diferentes.

Especialización: Este principio se basa en el funcionamiento de las *"Redes Empresariales"* de forma que una Comunidad Inteligente Especializada persiga unas metas concretas dentro de un contexto determinado para lo cual, se especialice en la resolución de problemas determinados, y la cooperación entre Comunidades Inteligentes Especializadas de forma atómica, se resuma en la eficiencia que este tipo de cooperación genera en las organizaciones humanas.

Facilidad de Implementación: El diseño y la posterior implementación de un sistema multiagente es una labor costosa para los desarrolladores (Bellifemine et al. 2001), por lo que los sistemas construidos a partir de la cooperación de varias Comunidades Inteligentes Especializadas, dentro el marco de la arquitectura SCODA, han de ser fáciles de implementar. Esta finalidad se consigue ya que la estructura de las Comunidades Inteligentes Especializadas es estándar, siendo los servicios que ofrecen los que han de ser programados.

Reutilización: Debido a que dentro de SCODA cada Comunidad Inteligente Especializada es adoptada como un sistema multiagente independiente, el cual provee una serie de servicios especializados, es decir, orientados hacia unas metas concretas y comunes, la reutilización de estas Comunidades Inteligentes Especializadas ha de ser viable en cualquier desarrollo de sistemas multiagente que requiera los servicios que éstas puedan ofrecer, basado en la arquitectura SCODA. Esta visión se basa en el paradigma de la tecnología de orientación a objetos, los cuales son encapsulados y utilizados en diferentes desarrollos donde se requiera su funcionalidad.

Computación Distribuida: Este principio conlleva la necesidad de descentralizar, y por tanto distribuir la carga computacional de los agentes que conforman las Comunidades Inteligentes Especializadas de SCODA (Shen y Norrie, 1998; Camarinha-Matos y Afsarmanesh, 2007), de forma que los servicios requeridos no los presten directamente los agentes integrantes de las Comunidades Inteligentes Especializadas, sino que tendrán la capacidad de ejecutarse de forma distribuida para que la carga computacional asociada a los agentes disminuya y la estructura de la arquitectura no tenga variaciones.

Tal y como se muestra en la **Figura 1,** SCODA se define en seis módulos básicos: Aplicaciones Externas, Protocolo de Comunicaciones, Módulo de Control, Plataforma de Agentes, Comunidades Inteligentes Especializadas y Servicios de las Comunidades.

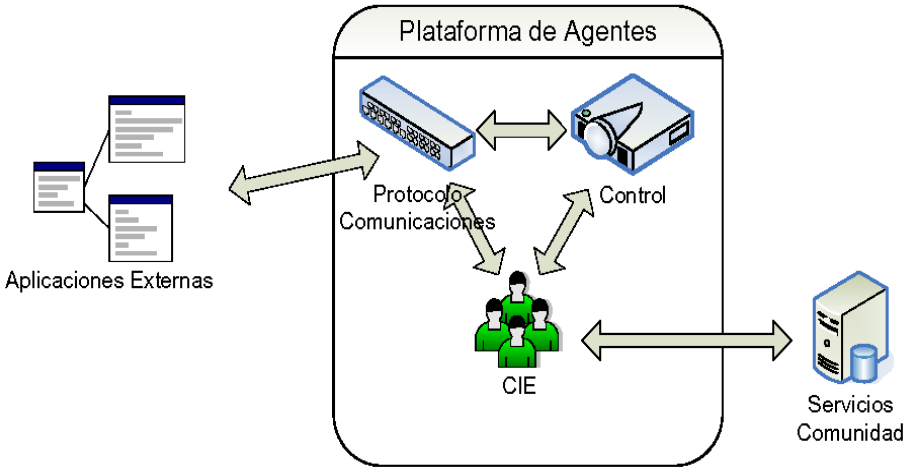

Figura 1 – Esquema estructural de SCODA

Las Aplicaciones Externas constituyen los programas y usuarios que hacen uso de SCODA solicitando los servicios que ofrece. El Protocolo de Comunicaciones es el encargado de atender las peticiones de las Aplicaciones Externas y solicitar una respuesta a las Comunidades Inteligentes Especializadas. El módulo de Control realiza un seguimiento de la coordinación y de las funcionalidades que la arquitectura provee atendiendo a una política de tolerancia a fallos en su funcionamiento. Las Comunidades Inteligentes Especializadas son el núcleo de SCODA, a través de las cuales se hacen efectivas las peticiones y respuestas de forma deliberada y optimizada. Los Servicios de la Comunidad se ejecutan de forma distribuida y es donde reside la capacidad de proceso de cada Comunidad Inteligente Especializada. Finalmente, la Plataforma de Agentes representa el entorno donde se ejecuta SCODA y está compuesta por los agentes que conforman la arquitectura y por la Comunidades Inteligentes Especializadas.

Los agentes que componen SCODA siguen el modelo deliberativo BDI y los servicios que ofrecen las Comunidades Inteligentes Especializadas son gestionados por este tipo de agentes. Concretamente uno de los agentes, el *PlannerAgent*, es el responsable de planificar que servicio es el óptimo y que parámetros son necesarios para que la solución demandada se ajuste a las necesidades del usuario o aplicación externa que realiza la petición. Otra característica aplicada a todos los agentes que componen SCODA, es que al ser agentes deliberativos BDI, éstos pueden hacer uso de mecanismos de razonamiento y técnicas de aprendizaje para realizar la gestión de funcionalidades y coordinación de las mismas, en función de las particularidades del contexto en que se ejecuten.

2.1. Aplicaciones Externas

Representan todos los programas que pueden utilizar las funcionalidades que provee el sistema multiagente implementados sobre SCODA. Estas aplicaciones son dinámicas y reaccionan de forma diferente de acuerdo a situaciones particulares como puede ser un gestor de agenda personal. Éstas pueden ser ejecutadas de forma local o remota, incluso desde dispositivos móviles con capacidad de proceso restringida, ya que las tareas que requieran una carga computacional alta, se realizarán de forma distribuida por los agentes que conforman SCODA a través de los servicios de las comunidades.

2.2. Protocolo de Comunicaciones

SCODA implementa un protocolo de comunicaciones basado en REST *(REpresentational State Transfer)* (Fielding, 2000; Griffin y Flanagan 2011), el cual permite a las aplicaciones externas comunicarse con los servicios ofrecidos por las Comunidades Inteligentes Especializadas a través de las mismas. El protocolo es completamente independiente de los lenguajes de programación utilizados, y está basado en peticiones sobre HTTP *(HyperText Transfer Protocol)* (RFC2616, 1999). Una petición HTTP *(HTTPRequest)* es enviada por las aplicaciones externas para especificar el servicio requerido, siguiendo el formato que se muestra a continuación:

Request = Simple-Request | Full-Request

Simple-Request = "GET" SP Request-URI CRLF

Full-Request = Request-Line

*(General-Header | Request-Header | Entity-Header)

CRLF

[Entity-Body]

En esta petición se informa de todos los parámetros necesarios para completar la tarea requerida. Todas las peticiones externas siguen las mismas pautas de comunicación y por lo tanto el mismo protocolo, mientras que las comunicaciones internas de la plataforma de agentes, siguen la especificación propia de la plataforma en la que se implemente SCODA. En el caso de JADEX se utiliza *FIPA Agent Comunication Language (ACL)* (FIPA, 2005). Finalmente la invocación de un servicio de una comunidad por parte de los agentes de la misma se realiza creando un nuevo hilo que se asocia a un socket, a través del cual se mantiene la comunicación abierta hasta que la tarea haya sido completada y el resultado se haya enviado a la Comunidad Inteligente Especializada que corresponde, o exista algún error en la ejecución de este servicio y por lo tanto también se informa del mismo. El formato de respuesta a la aplicación externa que solicita el servicio se realiza en forma de *HTTPResponse* como se muestra a continuación:

Response = Simple-Response | Full-Response

Simple-Response = [Entity-Body]

Full-Response = Status-Line

*(General-Header | Response-Header | Entity-Header)

CRLF

[Entity-Body]

Dentro del cuerpo de la *HTTPResponse*, y siguiendo la especificación REST se puede tener una representación de la respuesta en formato HTML, XML, etc (Muehlen et al., 2005).

La utilización de sockets en la invocación de servicios permite el procesamiento de múltiples solicitudes de forma simultánea. En el caso de procesamiento de servicios que requieran una alta carga computacional es la Comunidad Inteligente Especializada la responsable de realizar un balanceo de cargas invocando otro servicio que tenga la capacidad de satisfacer la petición. En la **Figura 2** se muestra un ejemplo de solicitud de un servicio a SCODA, a partir de la cual una aplicación externa realiza una petición a través de HTTP, siendo los agentes que conforman SCODA los responsables de seleccionar el servicio adecuado, llevar a cabo su ejecución y responder a la petición realizada.

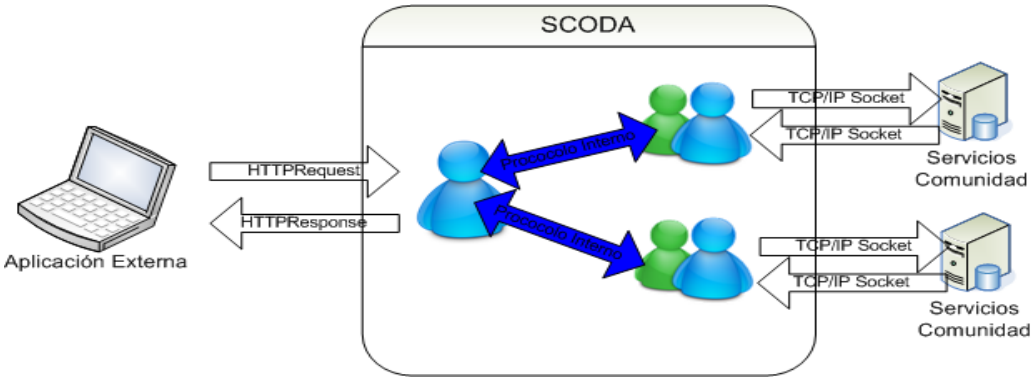

Figura 2 – Ejemplo de solicitud de un servicio a SCODA

2.3. Módulo de Control

SCODA implementa un mecanismo de control sobre los agentes integrantes de la misma, sobre los servicios, y sobre las comunicaciones realizadas. Así pues, a partir de este módulo, implementado por varios agentes, la arquitectura está preparada para asumir fallos en el funcionamiento de las aplicaciones y solventarlos de forma autónoma tomando las decisiones necesarias de forma deliberada, de forma que si alguno de los agentes activos en tiempo de ejecución tuviera un funcionamiento anómalo o dejara de funcionar, este módulo de control tiene la capacidad de reiniciarlos corrigiendo así su funcionamiento. También, en el caso de un mal funcionamiento de los servicios asociados a una Comunidad Inteligente Especializada, este módulo tiene la capacidad de buscar una réplica del servicio que pueda responder a una petición dada. Por otra parte, este módulo controla las comunicaciones realizadas entre las aplicaciones externas, los agentes integrantes de SCODA y los servicios asociados a las Comunidades Inteligentes Especializadas, de forma que si existiese algún tipo de comunicación anómala intenta su corrección reiniciando la misma, o en su caso reinicia todas las entidades implicadas en dicha comunicación.

2.4. Comunidades Inteligentes Especializadas (CIE)

Las Comunidades Inteligentes Especializadas son la esencia de SCODA. En ellas se encuentra la capacidad para distribuir y seleccionar el trabajo de forma inteligente a través de los agentes BDI que la componen. Constituidas por un agente controlador *(ComunityController)*, y un equipo de trabajo formado por un agente planificador y otro ejecutor *(PlannerAgent y ExecutorAgent)*, que son instanciados en tiempo de ejecución cuando son necesarios, y liberados al finalizar su trabajo. La propia arquitectura interna de la comunidad y la filosofía que se persigue en cuanto a la instanciación y liberación de agentes bajo demanda hace que SCODA sea una arquitectura distribuida en cuanto a sus servicios y eficiente en cuanto a la gestión de recursos internos de la plataforma que la ejecuta. Estas Comunidades Inteligentes Especializadas funcionan como sistemas autónomos, de forma que una implementación basa en SCODA puede estar compuesta por una o varias Comunidades Inteligentes Especializadas y hacer uso de los servicios que éstas proveen por separado, o utilizar varias Comunidades Inteligentes Especializadas y sus servicios de forma escalada y coordinada, y así tener la capacidad de resolver problemas mayores de forma colaborativa. A diferencia de otras organizaciones de agentes como las instituciones electrónicas (Arcos et al., 2005), las Comunidades Inteligentes Especializadas no constituyen un sistema multiagente abierto, sino pequeños sistemas multiagentes cerrados con capacidad de crecer de forma escalada y colaborativa en la consecución de objetivos más complejos, que de forma individual no puedan ser abordados, lo que implica una sencillez en el diseño de las mismas, ya que no es necesaria una fuerte normativa como en las instituciones electrónicas.

La especialización que estas entidades ofrecen de forma atómica es importante debido a que pueden ser reutilizadas en diversos desarrollos donde se requieran los servicios que provean.

2.5. Servicios de la Comunidad

Estos servicios representan las funcionalidades que ofrece la arquitectura. Los servicios de las comunidades son el grueso del procesamiento ofrecido por el sistema, los cuales son accedidos de forma ubicua y distribuida liberando a los agentes de SCODA de carga computacional y haciendo de la misma una arquitectura ligera. Los Servicios de la Comunidad se encuentran permanentemente activos para recibir peticiones de su Comunidad Inteligente Especializada correspondientes y están diseñados para su acceso de forma remota a través de sockets, debido a su facilidad de implementación y a su buen resultado en otros trabajos como son (Balaji et al. 2006; Douglas y Pai 2006; Sunwook et al. 2009), y están organizados por categorías en cuanto a su especialización, es decir los servicios relativos a una determinada funcionalidad están agrupados para su acceso por la comunidad especializada que ofrezca estos servicios, de esta forma se pretende buscar la eficiencia basada en la especialización, en forma de redes empresariales.

Los Servicios de la Comunidad, así como otro tipo de servicios distribuidos como los *Servicios Web y los Servicios Orientados a Arquitecturas* (Papazoglou et al. 2007) han de disponer de mecanismos de publicación y descubrimientos de servicios. También han de poseer un directorio actualizado de los mismos de forma que estén accesibles

cuando se requieran (Leymann et al. 2002; Papazoglou et al. 2007), es por lo que SCODA posee un directorio de servicios flexible desde el cuál pueden ser invocados de forma dinámica a través de las Comunidades Inteligentes Especializadas que los proveen. Sin embargo, la inserción, modificación o eliminación de los Servicios de la Comunidad se realiza de forma manual por razones de seguridad (López et al. 2006).

3 Plataforma de Agentes en SCODA

Este módulo es el núcleo de SCODA. Integra los agentes BDI que conforman las Comunidades Inteligentes Especializadas, y los agentes especiales de control y de gestión de las comunicaciones, cada uno de ellos con características especiales y comportamientos determinados. Estos agentes, a través de la inteligencia que poseen, actúan como administradores del sistema, de forma que sustentan el control de las comunicaciones, y administran el comportamiento del sistema.

Una ventaja que proporciona la arquitectura SCODA sobre los desarrollos que la implementan es que, las aplicaciones externas que hacen uso de los servicios que ésta provee pueden ser programadas en cualquier lenguaje de programación, con lo que aumenta la capacidad de utilización desde multitud de dispositivos. De esta forma, y como se muestra en la **Figura 3**, el acceso a los servicios ofrecidos por las Comunidades Inteligentes Especializadas puede realizarse desde dispositivos de sobre mesa, ordenadores portátiles, teléfonos móviles, PDAs, etc

A continuación se describen los diferentes agentes que integran la arquitectura SCODA y sus funcionalidades:

CommunicatorAgent: Este agente se responsabiliza de las comunicaciones externas con la plataforma. Recibe las peticiones de usuarios, aplicaciones o agentes externos y las transmite a las Comunidades Inteligentes Especializadas para su procesamiento, para ello mantiene un directorio de las Comunidades Inteligentes Especializadas y los servicios que estas ofrecen. También es el responsable de proporcionar las respuestas hacia el exterior. Este agente está constantemente en escucha a través de un *Server socket*. Se recibe una petición vía HTTP y es éste agente el que busca la comunidad adecuada y envía la petición a través del protocolo de comunicación interna de la plataforma. El hilo queda a la espera de obtener una respuesta para comunicarla al la entidad que realiza la petición. Paralelamente también se envía la respuesta al *QualityAgent,* el cual realiza las operaciones necesarias para mantener una estadística de funcionamiento, y en su caso, solucionar los problemas que se hayan podido plantear como la caída en el sistema de alguno de los agentes responsables de comunidad o incluso del propio *CommunicatorAgent*. Este agente se pertenece al Módulo de Control comunicándose con el *QualityAgent* para dar información a cerca de las comunicaciones realizadas en ambos sentidos, revisando así los mensajes recibidos, y además tiene la capacidad de inicializar al *QualityAgent* si se detecta un funcionamiento defectuoso del o inactividad en el mismo.

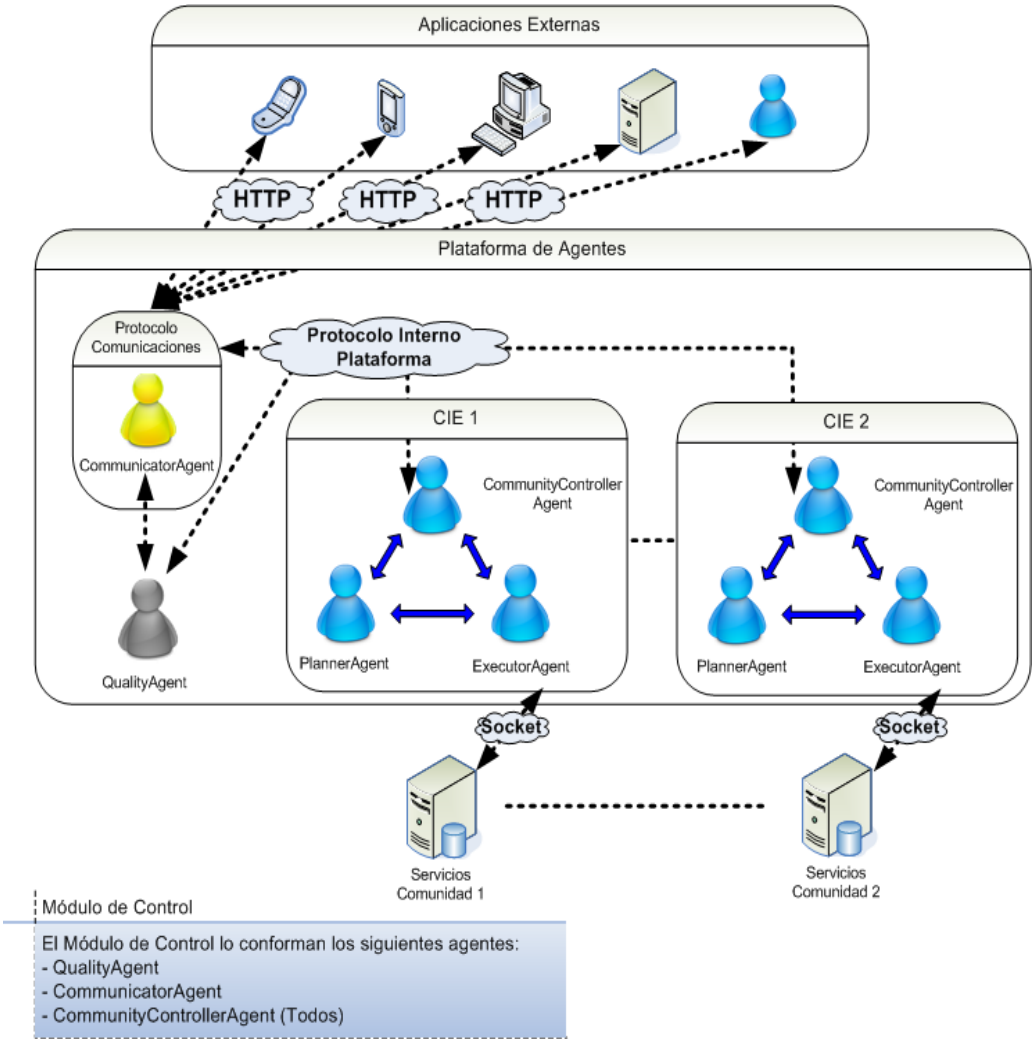

Figura 3 – Arquitectura SCODA

QualityAgent: Se responsabiliza del control del funcionamiento de los demás agentes, así como de las funciones que realizan. Se encuentra en constante comunicación con el *ComunicatorAgent* y con cada uno de los *ComunityControllers*. Tiene la capacidad de inicializar a los demás agentes que conforman SCODA, actuando sobre ellos en caso de un funcionamiento incorrecto. También realiza estadísticas de funcionamiento de las operaciones que se realizan, lo que implica que todos los movimientos operacionales son remitidos a éste agente en tiempo de ejecución.

CommunityControllerAgent: Este agente controla las Comunidades Inteligentes Especializadas. Una vez sido seleccionado por el *CommunicatorAgent* para resolver una petición externa, este agente crea un *"Equipo de Trabajo"* compuesto por un *PlannerAgent* y un *ExecutorAgent*, los cuáles son los responsables de ejecutar en sí

mima la petición. Este agente tiene la capacidad de crear y destruir al equipo de trabajo en tiempo de ejecución de forma que creará tantas instancias de los mismos como peticiones existan, y las irá eliminado una vez hayan finalizado su cometido, o cuando detecta un funcionamiento anómalo por parte de cualquiera de los agentes integrantes del *"Equipo de Trabajo"*. Este agente también pertenece al Módulo de Control, y se comunica con el *QualityAgent* para dar información a cerca de los servicios prestados y su resolución de forma que se pueda mantener un histórico de estadísticas de funcionamiento del sistema, y además tiene la capacidad de inicializar al *QualityAgent* si se detecta un funcionamiento anómalo o inactividad por parte de este agente.

PlannerAgent: Es uno de los componentes del *"Equipo de Trabajo"*, el cual es el responsable directo de la selección y ejecución del servicio adecuado que responda a una petición externa. Este agente es creado en tiempo de ejecución por el *CommunityController* cuando se ha de invocar un Servicio de la Comunidad y recibe la petición del propio *ComunityController* de su comunidad. Tiene la capacidad de seleccionar el servicio concreto de los que provee la comunidad y sus parámetros de forma que se optimice la ejecución del servicio, que posteriormente son comunicados al *ExecutorAgent*, el cual es el responsable de su invocación.

Se mantiene a la espera de un resultado por parte del *ExecutorAgent* lo cual confirma que el servicio requerido ha sido invocado y por lo tanto ha terminado el cometido del *"Equipo de Trabajo"*. Este agente tiene la capacidad de razonamiento que permite detectar el estado de los servicios en cada momento, para así seleccionar un servicio que no esté en ejecución y dotar de mayor agilidad a la Comunidad Inteligente Especializada.

ExecutorAgent: Este agente es el encargado de invocar al servicio pertinente para la resolución de una petición externa. La invocación se realiza a través de sockets, de forma que los servicios de cada Comunidad Inteligente Especializada pueden estar distribuidos de forma remota y por lo tanto la carga computacional repercute en la máquina donde se alberguen los servicios en sí. El *ExecutorAgent* se mantiene a la espera de una respuesta, y una vez recibida, la comunica de forma ascendente para advertir que su trabajo ya ha acabado.

4. Conclusiones

La filosofía seguida por SCODA permite implementar sistemas multiagente de una forma modular y escalada, dotando a los desarrolladores de la posibilidad de reutilizar las Comunidades Inteligentes Especializadas que la integran en diversos sistemas sin necesidad de volver a programar sus funcionalidades.

SCODA cuenta con una potente estructura de control, a partir de la cual se proporciona una completa capacidad en la gestión y recuperación de errores dentro del sistema y los servicios asociados al mismo, manteniendo en todo momento una estadística de funcionamiento, que permite adoptar decisiones y medidas en caso de un funcionamiento defectuoso del sistema y de los servicios asociados al mismo.

Otra de las ventajas que proporciona SCODA es que, junto a la metodología REST que adopta, permite la comunicación de forma estándar desde cualquier dispositivo y

aplicación independientemente del lenguaje de programación en la que esté desarrollada, o sistema operativo sobre la que se ejecute.

Para comprobar la validez del modelo teórico propuesto se ha llevado a cabo el desarrollo de un sistema multiagente, bajo el marco de SCODA, aplicado a la gestión logística de productos alimentarios. Este desarrollo se ha realizado en un entorno real empresarial, dedicado a la distribución de productos congelados. La metodología de trabajo de la empresa engloba tareas de gestión de inventarios, predicción de la demanda de productos y gestión de las rutas comerciales, lo cual hace idónea la filosofía modular que sigue SCODA permitiendo desplegar una Comunidad Inteligente Especializada por cada una de las tareas mencionadas.

Los resultados obtenidos, en cuanto al desarrollo del sistema, han permitido comprobar el correcto funcionamiento de las CIE que lo integran, tanto de forma individual, como de forma colaborativa, dando validez al modelo teórico propuesto. Sin embargo, en el caso de la tarea de predicción de la demanda de productos, los servicios desarrollados para mejorar los procesos seguidos por la empresa no han sido satisfactorios completamente, por lo que es necesario depurarlos con el fin de que el rendimiento del sistema alcance un mayor grado de optimización.

El desarrollo de este sistema se engloba dentro del trabajo de tesis doctoral *"Comunidades Inteligentes para la Construcción y Gestión de Arquitecturas Optimizadas de Sistemas Multiagente"*, que se está llevando a cabo por el autor principal. Una vez comprobada la validez del modelo teórico, se pretende llevar a cabo la creación de un repositorio de CIE de forma que se amplíe el campo de aplicación de SCODA en cuando al desarrollo de sistemas multiagente se refiere, lo que permitirá comprobar y optimizar el funcionamiento de SCODA en otro tipo de problemas. El desarrollo de estas CIE pretende ser abierto a la comunidad científica, de forma que se permita la aportación y recopilación de estas Comunidades Inteligentes Especializadas para su utilización.

Referencias bibliográficas

Arcos, J. L., Esteva, M., Noriega, P., Rodríguez, J. A., Sierra, C. (2005). Engineering open environments with electronic institutions. *Journal on Engineering Applications of Artificial Intelligence*. 18(2), 191-204.

Balaji, P., Bhagvat, S., Jin, H.-W., Panda, D.K. (2006). *Parallel and Distributed Processing Symposium. IPDPS.*

Bellifemine, F., Rimassa, G., Poggi, A. JADE - A FIPA-compliant Agent Framework. *In Proceedings of the 4th International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents, London, 1999.*

Bellifemine F., Poggi A., & Rimassa G. (2001). Developing multi agent systems with a fipa-compliant agent framework. *Software - Practice And Experience*, 31(2), 103–128.

Bellifemine, F., Caire, G., Greenwood, D. (2007). Developing Multi-Agent Systems with JADE. *John Wiley & Sons, NJ.*

Borgatti, S. Foster, P. (2003). The network paradigm in organizational research: A review and typology. *Journal of Management*, *29*(6), 991-1013.

Bratman, M. E., Israel, D., Pollack, M. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence, 4, 349-355.*

Cabus, P. Vanhaverbeke, W. (2006). The territoriality of the network economy and urban networks: Evidence from flanders. *Entrepreneurship & Regional development*, *8*, 25-53.

Camarinha-Matos, L. M., Afsarmanesh, H. (2007). A Comprehensive Modeling Framework for Collaborative Networked Organizations. *Journal of Intelligent Manufacturing ,* 18 (5), 529-542.

Douglas, C., Pai, V.S. (2006). Seekable sockets: a mechanism to reduce copy overheads in TCP-based messaging. *Parallel and Distributed Processing Symposium, pp-6.*

Elrad, T., Filman, R. Bader, A. (2001), Theme Section on Aspect-Oriented Programming, *CACM,* 44(10), 29-32.

Eraydin, A. Armatli-Köroglu, B. (2005). Innovation, networking and the new industrial clusters: the characteristics of networks and local innovation capabilities in the Turkish industrial clusters, *Entrepreneurship and regional development, 17*, 237-266.

Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*, Doctoral Dissertation University of California, Irvine, CA.

FIPA. (2005). *Foundation for Intelligent Physical Agents*. Retrieved 7 14, 2006, from http://www.fipa.org

Galaskiewicz, J. (1979). *Exchange Networks and Community Politics*. Beverly Hills: Sage.

Gay, S., Vasconcelos, V., Ravara, A., Gesbert, N.,Caldeira, A. (2010). Modular session types for distributed object-oriented programming. *In Proceedings of the 37th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 299-312.

Gonzalez-Palacios, J., Luck, M. (2007). Towards compliance of agents in open multi-agent systems. In *Software Engineering for Multi-Agent Systems V, Lecture Notes in Computer Science. Springer, 4408, 132-147.*

Griffin,K., Flanagan, C. (2011). Defining a call control interface for browser-based integrations using representational state transfer. *Computer Communications*. 4(2), 140-149.

Ibarra, H., Kilduff, M. Tsai, W. (2005). Zooming in and out: connecting individuals and collectivities at the frontiers of organizational network research. *Organization Science*, *16*(4), 359-371.

Jensen, M. C., Meckling, W. H. (1992). Knowledge, control and organizational structure. *Current Economics (Blackwell, Oxford). Lars Werin and Hans Wijkander, eds.* , 251-274.

Langtangen, P. (2011). Object-Oriented Programming. *Texts in Computational Science and Engineering*. 6, 437-496.

Leymann, F., Roller, D., Schmidt, M.-T. (2002). Web services and business process management. *IBM Systems Journal* 41(2), 198-211.

López, F., Luck, M., y d'Inverno, M. (2006). A normative framework for agent-based systems. *Computational and Mathematical Organization Theory,* 12,227–250.

Muehlen, M., Nickerson, J., Swenson, K. (2005). Developing Web Services Choreography Standards-The Case of REST vs. SOAP, *Decision Support Systems,* 40(1), 9-29.

Papazoglou, M. P., Traverso, P., Dustdar, S., Leymann, F. (2007). Service-Oriented Computing: State of the Art and Research Challenges. *Computer*. 40(11),38-45.

Pokahr, A., Braubach, L., Lamersdorf, W. (2003). Jadex: Implementing a BDI-Infrastructure for JADE *Agents. In EXP - in search of innovation (Special Issue on JADE),* 76-85.

Pokahr, A., Braubach, L., Walczak, A., Lamersdorf, W. (2007). Jadex - Engineering Goal-Oriented Agents. In Developing Multi-Agent Systems with JADE. *Eds., Wiley & Sons*, 254-258.

Pokahr, A., Braubach, L. (2009). From a Research to an Industrial-Strength Agent Platform: Jadex V2. *9. Int. Tagung Wirtschaftsinformatik*. 769-778.

Pöyhönen, A. Smedlund, A. (2004). Assessing intellectual capital creation in regional clusters. *Journal of Intellectual Capital*, *5*(3), 351-365.

Rao A. S. y Georgeff M. P. (1995) BDI Agents from Theory to Practice. *Proceedings of the First Int. Conference on Multi-Agents Systems. (ICMAS-95)*, 312-319.

Rashid, A., Moreira, A., Araujo, J. (2003). Modularization and composition of aspectual requirements. *In 2nd International Conference on Aspect-Oriented Software Development*, 11–20.

RFC 2616. (1999). Hypertext Transfer Protocol -- HTTP/1.1. *The Internet Society* 1999.

Shen, W., Norrie, D. H. (1998). An agent-based approach for distributed manufacturing and supply chain management. *In G. Jacucci (Ed.), Globalization of manufacturing in the digital communications era of the 21st century,* 579–590. Boston: Kluwer.

Sonquist, J. A., Koenig, T. (1975). Interlocking directorates in the top US corporations: a graph theory approach. *Insurgent Sociology*, *5*, 196-229.

Sunwook K., Chanho P., Seongwoon K., Yongwha C. (2009). The offloading of socket information for TCP/IP offload engine. *In 11th International Conference on Advanced Communication Technology, 1, 826-831.*

Viedma, J. (2004). Social capital benchmarking system: profiting from social capital when building network organizations. *Journal of Intellectual Capital*, *5*(3), 426-442.

Winikoff, M. (2009) Future Directions for Agent-Based Software Engineering. *In International Journal Agent-Oriented Software Engineering*. 3(4), 402-410.

Winikoff, M., Cranefield, S. (2010) On the testability of BDI agents. *8th European Workshop on Multi-Agent Systems (EUMAS2010).*

Zambonelli, F., Jennings, N. R., Wooldridge, M. (2000). Organisational abstractions for the analysis and design of multi-agent systems. *In 1st Int. Workshop on Agent-Oriented Software Engineering*, 127–141.