La memoria titulada **"Comunidades Inteligentes para la Construcción y Gestión de Arquitecturas Optimizadas de Sistemas Multiagente"** que presenta D. Jesús Ángel Román Gallego para optar al Grado de Doctor por la Universidad de Salamanca ha sido realizada bajo la dirección del profesor Dr. D. Juan Manuel Corchado Rodríguez, Catedrático del Departamento de Informática y Automática de la Universidad de Salamanca, y por la profesora Dra. Dª Sara Rodríguez González, Profesora Ayudante Doctor del Departamento de Informática y Automática de la Universidad de Salamanca.

Salamanca, septiembre de 2015

Los Directores

Dr. D. Juan M. Corchado Rodríguez
Catedrático
Informática y Automática
Universidad de Salamanca

Dra. Dª. Sara Rodríguez González
Profesora Ayudante Doctor
Informática y Automática
Universidad de Salamanca

El Doctorando

D. Jesús A. Román Gallego

Jesús Ángel Román Gallego: *Comunidades Inteligentes para la Construcción y Gestión de Arquitecturas Optimizadas de de Sistemas Multiagente*, PhD on Computers and Automation © Sep 2015

Supervisors:

Dr. D. Juan Manuel Corchado Rodríguez
Dra. Dª. Sara Rodríguez González

Location:

Salamanca

Nunca piensas que puede pasar
hasta que pasa, y finalmente,
te das cuenta que lo más importante
es lo que tienes a tu lado …

RESUMEN

El desarrollo de sistemas informáticos es una labor más o menos costosa en función de su complejidad. El hecho de poder reutilizar, parcial o totalmente, trozos de un sistema para otros desarrollos, implica una reducción en el tiempo empleado, una mayor facilidad de implementación y evita la redundancia de funcionalidades.

Este planteamiento llevado a los sistemas multiagente ha de tener en cuenta las características propias de los agentes, para lo cual se requiere que la reutilización pueda llevarse a cabo a partir de pequeños subsistemas de agentes especializados con una organización establecida. Además, para explotar la capacidad de estos pequeños subsistemas de agentes es necesaria una arquitectura que tenga como finalidad la coordinación de los mismos, y que de forma modular y escalada, pueda desarrollarse para lograr objetivos de mayor complejidad.

A lo largo de este trabajo se llevará a cabo un estudio de las características de los agentes y sistemas multiagente, asi como de las organizaciones humanas y su implementación a partir de las organizaciones virtuales, destacando su importancia y efectividad en el desarrollo actual de sistemas multiagente. Llegado este punto se realizará el diseño de SCODA (Distributed and Specialized Agent COmmunities), una nueva arquitectura modular para el desarrollo de sistemas multiagente. Mediante SCODA se permite el desarrollo de sistemas multiagente bajo una filosofía modular especializada, a través de la cual las funcionalidades del sistema puedan ir ampliándose, de forma escalada, en función de las necesidades.

SCODA se compone de pequeños subsistemas de agentes, denominados Comunidades Inteligentes Especializadas (CIE), los cuales proveen las funcionalidades necesarias para resolver las necesidades requeridas a través de servicios distribuidos. Mediante estas CIE se permite una escalabilidad de los sistemas de forma que puedan ser reutilizadas en diferentes desarrollos, independientemente de su finalidad.

La validación de esta arquitectura se realizará a partir de un caso de estudio centrado en tareas principalmente logísticas, debido a la variedad de situaciones que pueden darse en este tipo de ambientes. A partir de este caso de estudio se analizará y evaluará el comportamiento de la arquitectura y podrá llevarse a cabo su validación.

ABSTRACT

Computers systems development is more or less difficult task according to its complexity. The fact of being able to re-use, partially or completely, pieces of a system for other developments, involves a time reduction, a major implementation facility and avoids the functionalities redundancy.

This aim applied to multiagent systems has to bear in mind the own characteristics of the agents, for which it is needed that the re-using could be carried out from small subsystems of specialized agents with an established organization. Also, to improve the capacity of these small subsystems of agents, is necessary an architecture, that has the objective to take the coordination of the same ones, and in a modular and scalable way, could develop to achieve aims with a major complexity.

Throughout this work will be carried out a study of the characteristics of the agents and multiagent systems, as well as of human organizations and its deployment on virtual organizations, highlighting its importance and effectiveness in the current development of multiagent systems. From here it will be developed the design of SCODA (Distributed and Specialized Agent Communities), a new modular architecture for the development of multiagent systems. By means of SCODA, is allowed that multiagent systems could be developed from a specialized modular philosophy, across it the functionalities of the system can be extended in scaled form according to the objectives. SCODA is composed by small subsystems of agents named, Specialized Intelligent Communities (SCI), which provide the necessary functionalities to solve the objectives needed across distributed services. By means of these CIE, scalability of the systems is allowed, so that they could be re-used in different developments, independently of his purpose.

SCODA is integrated by smaller subsystems of agents, called Intelligent Communities Specialized (SCI), which provide the functionality necessary to resolve the aims, using distributed services. These SCI allow a scalability of the systems so that can be reused in different developments, regardless of its purpose.

The validation of this architecture will be realized through a case of study, focused on logistical tasks mainly due to the variety of situations that may arise in this kind of environments. From this case of study, we will analyze and assess the behaviour of the architecture and will carry out its validation.

AGRADECIMIENTOS

Ha sido duro, pero satisfactorio, todo el tiempo empleado en la realización de este trabajo. Es por ello, que quiero expresar mi más sincero agradecimiento a todos aquellos que han hecho posible mi llegada al final de este largo camino, especialmente este último año, tan duro para mí.

En primer lugar, quiero agradecer al Dr. Juan Manuel Corchado Rodríguez su labor como director, su paciencia, sus buenos consejos, y como he visto en él un ejemplo a seguir.

A la Dra. Sara Rodríguez González, de verdad, no sé como exponer en estas líneas todo el agradecimiento por tu labor, tus revisiones, tu ayuda. Siempre me has tendido la mano cuando más lo he necesitado.

Gracias a los miembros del grupo de investigación BISITE por servirme de inspiración en este trabajo.

A mis padres y hermano, Serafín, Virginia y Antonio, siempre habéis estado a mi lado, no importa cuándo ni dónde. Gracias.

A mi mujer, Rebeca. Siempre he encontrado en ti la persona que todos necesitamos en los momentos más difíciles. Gracias por tu paciencia, tu comprensión, y por tu forma de ser.

A mi niñas, Sofía y Julia, que desde que habéis llegado, mi motivación ha sido aún mayor. Vivo, por y para vosotras.

Gracias, Jefe, por haberme dado la perseverancia necesaria, para avanzar día a día, minuto a minuto, y segundo a segundo.

A todos los ausentes que me han ido dejando. Son los buenos recuerdos los que hacen esta vida más fácil, y transforman las adversidades en incentivos para seguir caminando.

Y finalmente, a todas las personas y situaciones que en algún momento de mi vida han puesto obstáculos en mi camino, sin ellas no hubiera aprendido a levantarme una y otra vez.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1.
INTRODUCCIÓN

1. INTRODUCCIÓN

Las personas y las organizaciones tienen la capacidad de solventar los problemas que aparezcan en un momento determinado teniendo en cuenta el cómo se han resuelto otros problemas en situaciones anteriores (Jensen, 1992).

El desarrollo de sistemas cada vez más complejos lleva consigo la necesidad de elaborar componentes con capacidad para ser reutilizados, de forma que las funcionalidades que estos proveen puedan ser utilizadas en otros desarrollos, guardando una compatibilidad entre los mismos. Esta filosofía es la que sigue la tecnología Orientada a Objetos (Lieberherr, 1996; Elrad *et al.,* 2001; Rashid *et al.,* 2003; Kiczales *et al.,* 1997), donde los objetos son encapsulados de forma independiente y pueden ser reutilizados en diferentes desarrollos con finalidades muy distintas, dotando a los desarrolladores de una importante ventaja en cuanto al tiempo empleado en poner en marcha el sistema.

Este planteamiento llevado a los sistemas multiagente implica que el desarrollo de una aplicación, basada en sistemas multiagente, ha de poder ser utilizada en otro tipo de desarrollos, aunque la finalidad global sea muy distinta. Para conseguir llevar a cabo este planteamiento, y atendiendo a la definición de sistema multiagente como un potente modelo computacional, basado en entidades software que cooperan entre sí a través de un enfoque autónomo, cooperativo y distribuido (Amigoni y Fugini 2007), hemos de pensar que el conjunto de agentes que conforman un sistema multiagente ha de guardar cierta estandarización y tamaño para poder reutilizarse. En este sentido, el número de agentes no ha de ser grande ya que en otro caso la complejidad a la hora de integrarlo en diferentes desarrollos podría aumentar. Algo evidente, es que un sistema multiagente con una gran capacidad de generalización, en cuanto a la funcionalidad que ofrece, no sería completamente adaptable a varios desarrollos diferentes ya que existirán funcionalidades que no sean utilizadas en un caso u otro, lo que implica que la especialización es importante a la hora de encajar este tipo de sistemas en este tipo de desarrollos.

El concepto de organización ha sido estudiado ampliamente en ciencias como la economía, la sociología y la psicología, y además, han sido diversos autores los que han aplicado el concepto de organización al desarrollo de sistemas multiagente (Zambonelli *et al.,* 2000; Ferber *et al.,* 2003; Giorgini *et al.,* 2004; Furtado 2005; McCallum 2005; Hubner et al. 2006). La aplicación del concepto de organización a un sistema multiagente aumenta aun más la eficiencia del mismo debido a que existe un control sobre los objetivos a cumplir, tanto individuales como colectivos. Se establecen unas normas de comportamiento entre los agentes y existe de algún modo una división de tareas o especialización, es decir, el funcionamiento del sistema se protocoliza (Zambonelli *et al.,* 2003; Pavón et al. 2003; Pavón et al. 2005; Dignum y Dignum, 2006; Hubner et al. 2006). Si además se desea que un sistema multiagente tenga la capacidad de poder utilizarse en diferentes desarrollos independientemente

del fin global de los mismos, sería necesario definir una nueva arquitectura modular y especializada, basada en algún tipo de organización de tamaño mínimo, que establezca las normas de coordinación entre los agentes. Se entiende por especialización dentro de un grupo como un mecanismo para obtener mayor eficiencia en la consecución de un objetivo determinado. Además la especialización puede darse intragrupo o intergrupos, y consta de diferentes tipos y grados que serán explicados detenidamente en el capítulo 3 de este trabajo. Esta arquitectura tendría un alto grado de especialización a nivel organizacional, debido a que la arquitectura de un sistema multiagente determina la composición del propio sistema, y los mecanismos que utilizan los agentes para interactuar con su entorno (Corchado, 2005).

1.1. Hipótesis y Objetivos de la Investigación

Los desarrollos de sistemas computacionales tienen un fin específico, de forma que proporcionen la solución a uno o varios problemas determinados. La reutilización de estos sistemas no se contempla en otros sistemas con otros fines deferentes ya que la funcionalidad que poseen es inherente a cada sistema, lo que implica un nuevo diseño e implementación. La composición de estos sistemas basada en organizaciones virtuales que se asimilan a las organizaciones humanas, permite una definición más concisa de sus componentes además de controlar sus interacciones de forma que se traduzca en una optimización de estos sistemas. Sin embargo, la especialización inherente en las organizaciones humanas, y entendida como un mecanismo para obtener una mejora en un proceso productivo, no ha sido evaluada ni medida en las organizaciones virtuales, y con esto la posibilidad de optimizar su funcionamiento.

La hipótesis de partida es que: *es posible modelar una arquitectura adecuada para el desarrollo modular y creciente de sistemas especializados, a partir del modelo productivo empresarial, mediante el uso de organizaciones de agentes inteligentes*.

A partir de aquí, el objetivo final que se pretende con este trabajo es el diseño y desarrollo de una arquitectura modular basada en algún modelo de organización que permita especialización, y que a partir de ella se pueda llevar a cabo la construcción y gestión de sistemas multiagente independientemente de sus fines. La arquitectura ha de implementar funcionalidades que puedan ser desplegadas en entornos dinámicos y distribuidos. Esta arquitectura debe proporcionar los mecanismos necesarios para modelar problemas donde intervengan agentes y sistemas multiagente ya que son un pilar fundamental en el desarrollo de la misma. Además, la arquitectura ha de ser estándar de forma que sea independiente del problema a resolver, implementando una especialización de los módulos que la componen. Los componentes de esta arquitectura han de poder reutilizarse en

diferentes desarrollos permitiendo así, una modularidad que facilite el desarrollo de sistemas multiagente basados en esta arquitectura.

Para llevar a cabo este trabajo, se ha llevado a cabo un análisis del estado del arte de la Teoría de la Organización, las organizaciones y la especialización en las mismas, la tecnología de agentes, modelos de razonamiento en agentes, arquitecturas orientadas a servicios, metodologías para el análisis y diseño de sistemas multiagente, técnicas aplicadas a la optimización logística, así como una serie de tecnologías afines a los siguientes subobjetivos:

- Realizar un análisis de los tipos de organizaciones aplicadas a los sistemas multiagente, para llevar a cabo una comparación entre los mismos y poder obtener conclusiones a cerca de su funcionamiento.
- Analizar los beneficios de la especialización dentro de las organizaciones y sus tipos, y comprobar el grado de aplicación dentro de las organizaciones de agentes.
- Llevar a cabo un análisis de los agentes y sistemas multiagente con el fin de buscar mecanismos de razonamiento automáticos, para tomar decisiones de forma dinámica en entornos organizacionales aplicables a este tipo de sistemas.
- Proponer una arquitectura modular para el desarrollo de sistemas multiagente. Esta arquitectura ha de ser estándar, reutilizable, permitir especialización, de fácil implementación y distribuida.
- Llevar a cabo el completo desarrollo de la arquitectura, realizando una formalización del modelo, empleando herramientas para la Ingeniería del Software Orientada a Agentes (Agent Oriented Software Engineering, AOSE), y posteriormente implementar un sistema multiagente, basado en esta arquitectura, para resolver problemas logísticos en un entorno real empresarial.
- Analizar y evaluar de forma empírica, tanto los resultados obtenidos en el desarrollo del sistema para resolver problemas logísticos, basado en la arquitectura propuesta, como el rendimiento obtenido con la implantación del sistema multiagente en el entorno empresarial.

En estos términos se propone SCODA (Distributed and Specialized Agent COmmunities), una nueva arquitectura para el desarrollo de sistemas multiagente. SCODA permite desarrollar sistemas multiagente bajo una filosofía modular y especializada, donde las funcionalidades del sistema puedan ir ampliándose en base a las necesidades. Estas funcionalidades han de tener la capacidad de ser reutilizadas en otros desarrollos, bajo SCODA, de forma que la complejidad de llevar a cabo los mismos se vea reducida.

1.2. MOTIVACIÓN

Un sistema multiagente abierto es el que permite la adhesión de agentes en tiempo de ejecución sin la necesidad de conocerse durante el diseño (González-

Palacios y Luck, 2007), pudiendo utilizar diferentes protocolos de comunicación y cuya implementación puede haber sido realizada sobre diferentes arquitecturas. Este tipo de sistemas son de difícil implementación ya que se requiere de una alta coordinación y una fuerte normalización que permita controlar este flujo de agentes. Hay que tener en cuenta que estos agentes pueden haber sido implementados por diferentes personas que hayan utilizado distintos lenguajes de programación y protocolos de comunicación diferentes, por lo que la mayor parte de los sistemas multiagente implementados se hace sobre un entorno cerrado, es decir, todos sus componentes se conocen en la fase de diseño (Zambonelli *et al.*, 2000), por lo que la versatilidad de los mismos es menor.

Entre estos dos extremos podrían estar los sistemas multiagente cerrados que tengan la capacidad de crecer o disminuir de forma modular en función de las necesidades. Estos se podrían crear con una filosofía de especialización organizacional empresarial y, daría lugar a desarrollos de sistemas multiagente de forma rápida y eficiente. Además no se ha de tener en cuenta la fuerte normalización exigida en los sistemas abiertos y dotarían de la versatilidad necesaria de la que carecen los sistemas cerrados. Otro de los problemas que se resolverían con esta filosofía es la estandarización de los desarrollos, debido a que al ser un sistema cerrado con capacidad de crecimiento modular las, implementaciones que se realicen sobre el mismo han de seguir una serie de patrones de diseño en cuanto a su estructura. Además se ha de resaltar que en muchas ocasiones, los sistemas multiagente cargan computacionalmente, de forma elevada, a uno o varios agentes que lo componen, por lo que se ha de contemplar la posibilidad de distribuir las funcionalidades de cada uno de los agentes, de forma que los agentes no soporten la carga computacional de los servicios necesarios que los dotan de funcionalidad, atendiendo a un desarrollo orientado a servicios (Camarinha-Matos *et al.*, 2007; Tapia y Corchado, 2009).

Es por ello que surge la necesidad de desarrollar la arquitectura SCODA. Esta arquitectura ha de ser modular y especializada, con comportamiento organizacional, para que a partir de ella puedan llevarse a cabo desarrollos de sistemas multiagente independientemente de sus fines. La arquitectura se caracteriza por tener funcionalidades con capacidad para ser desplegadas en entornos dinámicos y distribuidos teniendo así una mayor versatilidad en la interacción con sus usuarios. Para lograr el fin perseguido, se presenta una arquitectura que sigue 5 principios:

- **Estandarización:** Se busca que la estructura de SCODA sea independiente del problema al que se aplique, de forma que a través de la misma se pueda resolver multitud de problemas en los que se quiera adoptar enfoque de agentes. Además las comunicaciones han de implementar los protocolos estándar de la FIPA (FIPA, 2005) de forma que siga la línea marcada por esta institución, debido a la importancia que tiene en cuanto al desarrollo de normas que afectan a sistemas multiagente y a la interoperabilidad de dichas normas con otras tecnologías. Esta organización ha sido aceptada por la IEEE de forma oficial en 2005.

- **Especialización:** Se pretende que los módulos que componen SCODA se especialicen en problemas determinados, de forma que esta especialización se resuma en eficiencia, debido a que la especialización en una tarea determinada resulta más productiva que la dedicación a varias tareas (Maudos *et al.,* 2002). Esto implica que en la organización de SCODA no existirá una departamentalización (Gasser, 2001), en la que los agentes pueden jugar varios roles.
- **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 SCODA han de ser fáciles de implementar. A partir de la propia arquitectura han de ser servicios distribuidos los que provean la funcionalidad, y por lo tanto los componentes necesarios a programar.
- **Reutilización:** Debido a la especialización de servicios que provee SCODA, se busca que una vez implementada una comunidad, pueda ser reutilizada integrándola en otras arquitecturas compuestas por otras comunidades, simples o complejas, siempre y cuando se requiera el servicio ofrecido por la misma.
- **Computación Distribuida:** Este principio conlleva la necesidad de descentralizar, y por tanto distribuir la carga computacional de los agentes que conforman SCODA. Se trata de que los servicios requeridos no los prestarán directamente los agentes pertenecientes a SCODA, sino que se ejecutarán de forma distribuida. Esto hace que la arquitectura sea ligera y no cargue computacionalmente a los agentes que la componen (Tapia y Corchado, 2009).

La implementación de esta arquitectura en un entorno empresarial real, con aplicación a la resolución de problemas logísticos, se convierte en un punto de motivación extra a la hora de llevar a cabo el desarrollo de la misma. Su evaluación se ha realizado en colaboración con la empresa Bahía Príncipe Congelados, S.L. de Zamora, donde se ha desplegado un sistema multiagente implementado bajo el marco de SCODA para la resolución y automatización de problemas logísticos dentro de esta empresa.

SCODA pretende ser una alternativa al diseño e implementación de sistemas multiagente, de forma que el coste de desarrollo de estos sistemas se reduzca debido a la posible reutilización de los componentes modulares que la conforman, además de aumentar la eficiencia, en cuanto al funcionamiento de estos sistemas, a través de la descentralización de los servicios, o funcionalidades que éstas proveen.

1.3. METODOLOGÍA DE TRABAJO

Para alcanzar los objetivos planteados en esta tesis doctoral se ha seguido el siguiente plan de trabajo basado en la metodología investigación-acción (Action-Research) (Burns, 2007):

1. **Definición del problema y descripción de las características que lo componen:** Se presenta la problemática definiendo sus características y proponiendo una hipótesis para resolver de forma total o parcial dicha problemática planteando los objetivos para lograrlo.
2. **Revisión constante y actualización del estado del arte:** Se lleva a cabo un análisis del estado del arte de las áreas, tecnologías y desarrollos relacionados con la presente investigación. A través de este análisis se podrá obtener un marco teórico sostenible.
3. **Desarrollo incremental e iterativo de la propuesta:** Mediante la información obtenida de las actividades anteriores, se diseña y desarrolla un modelo que integre los componentes necesarios para proponer una solución útil y novedosa a la problemática definida, siguiendo los objetivos planteados.
4. **Experimentación e implementación de la solución:** Se lleva a cabo la implantación de la solución de forma iterativa e incremental, de forma que se tenga la posibilidad de realizar pruebas incrementales sobre dicho desarrollo.
5. **Análisis de resultados y formulación de conclusiones:** En este punto se realiza una recopilación de los resultados obtenidos para llevar a cabo un análisis y evaluación de los mismos. Una vez analizados los datos, se podrán extraer las conclusiones necesarias en base a la hipótesis y objetivos planteados.
6. **Difusión de Resultados:** Se lleva a cabo la difusión de los resultados parciales o finales de la investigación, a través de revistas especializadas, congresos, etc.

1.4. ESTRUCTURA DE LA MEMORIA

El proceso de investigación, la experimentación y los resultados obtenidos que integran esta tesis doctoral son recogidos en los siguientes capítulos:

- **Capítulo 2: Organizaciones.** En este capítulo se hace un recorrido por la Teoría de la Organización, y por los diferentes tipos de organizaciones, tanto de humanos, como de agentes, a las que ha dado lugar, con el fin de adecuar un tipo de organización que se ajuste a las necesidades de la arquitectura planteada en esta tesis.

- **Capítulo 3: Especialización.** En este capítulo se hace referencia al concepto de especialización como una mejora en los procesos productivos de un grupo. Esta especialización, se pretende llevar a cabo en los sistemas multiagente con una estructura organizacional en forma de Comunidad Inteligente, estructura organizacional previamente definida en el capítulo 2, con el fin de mejorar su diseño y escalabilidad en su construcción. Para lograr este fin se ha de introducir el concepto de especialización y analizar el comportamiento de esta especialización, en diversos tipos de organizaciones.

- **Capítulo 4: Agentes y Sistemas Multiagente.** A lo largo de este capítulo se realiza una revisión sobre los agentes y sistemas multiagente, donde se realiza un análisis de su tipología y capacidades en función de la misma. Además, se detallan los tipos de arquitecturas que dan lugar a los sistemas multiagente incidiendo en la arquitectura BDI *(Belief Desire Intention)* (Bratman, 1988; Rao y Georgeff, 1995), como base para el desarrollo de SCODA. Finalmente se analizan las diferentes metodologías y plataformas para el desarrollo y ejecución de estos sistemas, en entornos distribuidos.

- **Capítulo 5: SCODA.** En este capítulo se presenta la arquitectura SCODA (*Distributed and Specialized Agent COmmunities*). Para ello se lleva a cabo el análisis y diseño de su estructura y funcionalidades. Se incidirá de forma detallada en las capacidades que ofrece haciendo uso de herramientas AOSE (Agent Oriented Software Engineering), de forma que puedan definir los principales componentes que integran esta arquitectura. Además, se presenta la herramienta, *Community Agent Generator*, para la generación automática de sistemas multiagente basados en SCODA, facilitando así los desarrollos de este tipo de sistemas.

- **Capítulo 6: Caso de estudio.** En este capítulo se presenta el caso de estudio. Se evalúa una aplicación desarrollada a partir de un sistema multiagente baso en SCODA. Este sistema se ha llevado a cabo en un entorno empresarial y tiene la capacidad de resolver problemas relacionados con la logística, tales como optimización de inventarios y optimización de rutas.

- **Capítulo 7: Conclusiones.** En este capítulo se presentan las conclusiones obtenidas de este trabajo, las contribuciones realizadas y las posibles líneas de investigación y desarrollo futuras que se seguirán a partir de esta tesis doctoral.

- **Capítulo 8: Research Overview.** Es el último capítulo del documento, y está formada por un resumen en inglés de todo el trabajo realizado, tanto estudios iniciales, como propuesta, resultados y conclusiones obtenidas.

Finalmente, se presentan las referencias bibliográficas que han servido como base fundamental para el desarrollo de este trabajo de tesis doctoral, y los anexos donde se detallan diferentes fases del desarrollo de la arquitectura SCODA.

Los anexos permiten completar y detallar diferentes aspectos de la memoria que necesitan ser estudiados con mayor nivel de detalle. El anexo A describe la metodología de GAIA y el lenguaje de modelado para agentes (AML). El anexo B detalla el proceso de análisis y diseño de la arquitectura SCODA mediante GAIA y AML. En él se detalla tanto el análisis de la arquitectura como su diseño a nivel estructural y dinámico.

CAPÍTULO 2. ORGANIZACIONES

2. ORGANIZACIONES

A lo largo de este capítulo se hace un recorrido por la Teoría de la Organización, y por los diferentes tipos de organizaciones a los que ha dado lugar, con el fin de adecuar un tipo de organización que se ajuste a las necesidades de la arquitectura planteada en esta tesis y que facilite el desarrollo de sistemas multiagente en los términos planteados.

2.1. INTRODUCCIÓN

El concepto de organización se ha estudiado ampliamente en ciencias como la economía, la sociología y la psicología. Desde la inteligencia artificial distribuida, diversos autores han aplicado el concepto de organización en sus trabajos con el fin de desarrollar sistemas multiagente de forma eficiente y organizada. Ejemplos de estos trabajos los encontramos en (Zambonelli *et al.,* 2000; Ferber *et al.,* 2003; Giorgini *et al.,* 2004; Furtado 2005; McCallum 2005; Hubner et al. 2006) entre otros.

Tomando como referencia que un sistema multiagente está compuesto por un grupo de agentes que interactúan de forma coordinada, y que persiguen un objetivo general, hemos de conocer las características principales de las organizaciones y las reglas que las rigen, con el fin de adaptar un modelo de organización a la arquitectura que se propone en este trabajo, de forma que su funcionamiento sea óptimo y el la complejidad en su desarrollo se decremente de forma importante (Dignun y Dignum, 2012).

Es por ello que en este capítulo se realizará un estudio de las organizaciones humanas y de agentes existentes teniendo en cuenta una comparativa entre las mismas. Se extraerán las características de cada una de ellas sopesando las ventajas y los inconvenientes que presentan, con el fin de encontrar una organización que se ajuste a las necesidades de la arquitectura planteada en esta tesis.

El presente capítulo se estructura de la siguiente forma. En la sección 2.2. se realiza un recorrido por las diferentes teorías de la organización y se introducen las organizaciones humanas. En la sección 2.3 se alude a las organizaciones en los sistemas multiagente. A continuación, en la sección 2.4 se detallan los factores que influyen en las organizaciones y se trata el diseño de las mismas. Posteriormente en la sección 2.5 se detallan los tipos de organizaciones, tanto humanas como de agentes, y se introduce el concepto de comunidad como tipo de organización. En la sección 2.6 se realiza una comparativa entre las comunidades, como tipo de organización, y los demás tipos vistos en la sección anterior. Finalmente, en la sección 2.7 se presentan las conclusiones obtenidas en este capítulo.

2.2. Teoría de la Organización y Organizaciones Humanas

Durante el desarrollo de esta sección se introduce el concepto de organización, y un recorrido por las diferentes teorías de la organización más relevantes a lo largo de la historia. Con el fin de analizar cada una de ellas y comprender mejor su fundamentación, se expondrán las teorías en esta área, para finalmente tener la capacidad de aplicar una adaptación, de una de ellas, a la arquitectura presentada en esta tesis.

El término *organización* ha sido definido por autores con enfoques distintos. A continuación se presentan varias definiciones, desde los puntos de vista de sus autores.

Según J. Massie, la organización es *"un grupo cooperativo de seres humanos que asigna las tareas entre los miembros, identifica las relaciones e integra sus actividades hacia objetivos comunes, de forma estructurada"* (Massie, 1973).

Por su parte, March, J.G. y Simon, H.A. aluden a que *"el concepto de organización no es fácil y que es mucho más sencillo citar ejemplos de organizaciones que dar una exacta definición del término"* (March y Simon, 1981). En esta definición se observa, entre líneas, la diversidad de organizaciones existentes, tanto a nivel de principios organizativos, como por su tipología.

Para I. Guzmán *"la organización es la coordinación de las actividades de todos los individuos que integran una empresa con el propósito de obtener el máximo aprovechamiento posible de elementos materiales, técnicos y humanos, en la realización de los fines que la propia empresa persigue"* (Guzmán, 1983).

J.M. Peiro considera la organización como *"una formación o entidad social con un número de miembros que puede ser precisado y una diferenciación interna de las funciones que son desempeñadas por dichos miembros"* (Peiro, 1991).

Atendiendo a las definiciones anteriores, se pueden definir las organizaciones como unidades sociales que tienen objetivos particulares. Están compuestas por individuos que llevan a cabo tareas específicas. Su estructura está regulada por reglas que permiten una coordinación entre los individuos con el fin de alcanzar unos fines comunes y determinados, siempre teniendo en cuenta que los fines han de ser conocidos por todos los individuos y han de cumplir las funciones de guiar los esfuerzos de los miembros para llevarlos a cabo (Peiro, 1991).

La Teoría de la Organización tiene poco más de un siglo de antigüedad. Es a su vez una disciplina compleja debido a la gran cantidad de enfoques que se han generado a lo largo de estos años. En esta línea Pfeffer afirma que "*el campo de la*

Teoría de la Organización se asemeja cada vez más a un abigarrado matorral en lugar de parecerse a un jardín cuidado con esmero" (Pfeffer, 1989).

A continuación se proponen los enfoques de la Teoría de la Organización más importantes, resumiendo los aspectos más relevantes de cada una de ellas, y teniendo en cuenta la aplicabilidad al objetivo principal de esta investigación.

- **Teoría Científica:** F. Taylor, en (Taylor, 1916), define cuatro principios de trabajo; desarrollo de una ciencia de medición del trabajo de los individuos, selección científica de los individuos y entrenamiento de los mismos, esfuerzo cooperativo de los individuos y la idea de que el trabajo y la responsabilidad son compartidos por los trabajadores y por la dirección.

- **Teoría Funcional:** La aportación de esta teoría dada por H. Fanyol es que se considera como la mejor forma de organización la que se basa en la división de funciones, que a su vez se dividen en subfunciones y procedimientos, que a su vez son desarrollados por uno o más roles (Fanyol, 1949).

- **Teoría Burocrática:** M. Weber, por su parte, sostiene en (Weber, 1947), que la manera más eficaz de organización es parecida a una máquina. Tiene reglas, controles, está jerarquizada y se rige a través de la burocracia.

- **Teoría de los Sistemas Cooperativos:** Esta teoría defendida por Barnard dice que la mejor forma de organización ha de asegurar la cooperación de sus miembros con un trato justo y beneficios recíprocos (Barnard, 1938).

- **Teoría de los Sistemas:** Esta teoría toma la idea de que las organizaciones son sistemas abiertos que están constituidos por subsistemas que se relacionan con el medio ambiente. Defiende que la mejor forma de organización es la que puede coordinar de forma armónica los diferentes subsistemas que se integran dentro de un sistema organizacional (Kast y Rosenzbeig, 1976).

- **Teoría del Comportamiento:** Esta teoría define como la mejor forma de organización la que permite a los empleados de todos los niveles, tomar decisiones y colaborar en el cumplimiento de los objetivos de acuerdo a su nivel de influencia y autoridad (March y Simon, 1981).

- **Teoría de la Contingencia:** Desde esta teoría se defiende que la organización depende de la tecnología, tamaño y medio ambiente (Woodward, 1958) y (Burns y Stalker, 1961). Esta teoría rompe con las teorías universales hasta ahora vistas haciendo hincapié en los recursos de la empresa como factor diferenciador en la definición del tipo de organización.

- **Teoría de los Recursos:** Para Barney, la mejor manera de organización es la que tiene la capacidad de gestionar de forma más racional sus recursos y sus capacidades (Barney, 1991).

- **Teoría de la Población Ecológica:** Resumiendo, esta teoría defiende que la mejor forma de organización es la que de algún modo consigue adaptarse al entorno interactuando con él y opera con eficiencia (Hannan y Freeman, 1989).

- **Teoría de la Agencia:** Esta teoría toma en consideración el comportamiento de cada integrante de la organización sobre la organización. Dice que la mejor forma de organización es la que implementa los mecanismos que previenen y corrigen que integrantes de la organización actúen a favor de sus propios intereses y premia a los mismos si actúan a favor de los intereses de la organización (Rumelt *et al.*, 1991).

- **Teoría de los Sistemas Complejos Adaptativos:** Por su parte S. Kauffman en (Kauffman, 1995) dice que la mejor organización es la que permite constantes ajustes entre los elementos de la organización y con el entorno que la rodea.

- **Teoría de la Autocriticabilidad Organizada:** Esta teoría formulada en (Maturana y Varela, 1980), sostiene que la mejor forma de organización es la que es capaz de crear una red de procesos, que a su vez pueden crear o destruir elementos del mismo sistema, dando respuesta a las interacciones con el entorno.

Una vez conocidos los puntos más importantes, de los diversos enfoques propuestos, sobre la Teoría de la Organización que pueden tener campo de aplicación en este trabajo, existen una serie de características comunes dadas en (Hodge el al., 2003) para las organizaciones humanas. De estas organizaciones hay que destacar que:

- Se componen de personas.
- Persiguen fines concretos, los cuales guían las acciones de sus miembros de forma coordinada y controlada.
- Existe una subdivisión del trabajo a través de la especialización o división de tareas.
- Necesita de una estructura formal donde existen una serie de roles, responsabilidades asignadas a esos roles y las relaciones necesarias entre los miembros de la organización.
- Las actividades realizadas por los diferentes roles han de estar relacionadas con los objetivos globales de la organización.
- Hay que establecer unos límites en cuanto a la pertenencia de los miembros a la organización.

2.2.1. La Organización Como Sistema Abierto

Las organizaciones pueden ser definidas como sistemas abiertos que dependen de organismos externos (clientes, proveedores, accionistas, etc.) para conseguir los recursos energéticos que requieren (trabajo, materiales, capital, etc.) y para enviarles el producto organizacional (tangible o intangible). Esto significa que la organización se dedica constantemente a varios tipos de transacciones ambientales, tales como distribución de productos, abastecimiento de materias primas, reclutamiento de personal, obtención de información, etc. (Aumada, 2001).

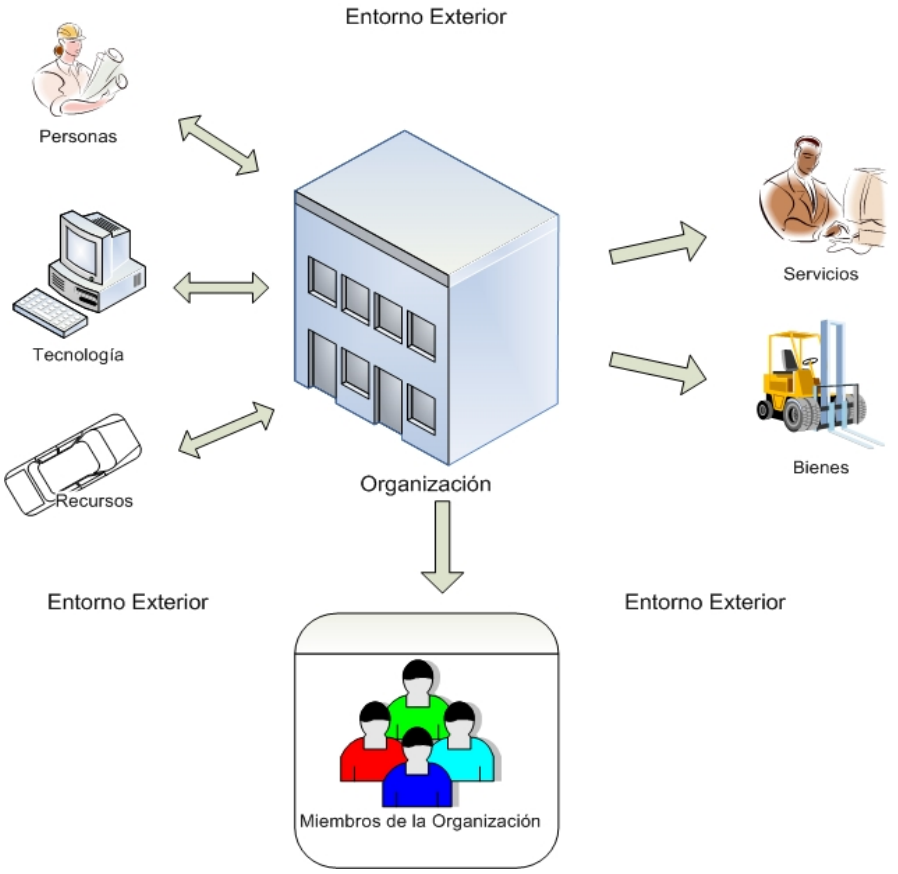


Figura 2.1. Representación gráfica de una organización como sistema abierto

Así pues, partiendo de la concepción de organización como sistema abierto, dada por L. Aumada, se ha de tener en cuenta la dinamicidad que existe dentro de la misma, y fuera con su entorno representada de forma gráfica en la **Figura 2.1.** La organización ha de estar preparada para constantes cambios organizacionales que implican una serie de relaciones entre los elementos de la organización y respuestas precisas a su entorno, lo cual plantea, que por una parte ha de ser capaz de generar la

suficiente variedad interna para hacer frente a la infinita variedad del entorno, y por otra, ha de ser capaz también de diseñar mecanismos reductores de variedad, los cuales han de seleccionar los estímulos relevantes para la organización, dejando de lado los que son irrelevantes, lo que permite aumentar la capacidad de respuesta a las demandas del entorno (Aumada 2001).

2.3. Sistemas Multiagente y Organizaciones

Un sistema multiagente consta de un grupo de agentes los cuales funcionan como algún tipo de organización. La organización se emplea para describir a ese grupo de agentes que se coordinan a través de una serie de patrones de conducta y el establecimiento de unos roles para alcanzar los objetivos del sistema. A continuación se muestran diferentes enfoques del término organización de diversos autores expertos en este campo.

Para L. Gasser (Gasser, 2001), el concepto de organización consiste en un sistema estructurado el cual tiene patrones de actividad, conocimiento, cultura, historia y habilidades distintas y complementarias de cualquier agente particular. Las organizaciones existen a un nivel completamente independiente de los agentes individuales que los constituyan, los cuales pueden ser reemplazables, además de ocupar alguna región del espacio, temporal, simbólico, etc. Es por ello que una organización de agentes dota de un marco de trabajo donde existe una actividad e interacción de los agentes coordinados en base a una serie de roles, expectativas y reglas. En (Zambonelli *et al.*, 2003) se adoptan las organizaciones de agentes como un conjunto de roles relacionados entre sí que interactúan entre ellos de forma sistemática e institucionalizada.

J. Ferber, asume que la organización de agentes proporciona un modo de particionar el sistema en grupos que constituyen la unidad de interacción de los agentes además de basar dicha organización en dos aspectos: estructural y dinámico (Ferber *et al.*, 2003). El aspecto estructural representa la parte estática de la organización, es decir, todo lo que persiste cuando los diferentes componentes entran o abandonan la organización. Define las agrupaciones en unidades organizativas y sus relaciones, además de los roles y reglas necesarias para llevar a cabo los cometidos propios de la organización. El aspecto dinámico tiene en cuenta los modelos de interacción definidos para cada rol describiendo las formas de entrar y abandonar la organización, además de las obligaciones y deberes requeridos a cada rol, y su asignación a los agentes. Así pues en (Rodríguez, 2010) se define una Organización Virtual como un *"sistema abierto formado por la agrupación y la colaboración de entidades heterogéneas y donde hay una separación entre la forma y la función que define su comportamiento"*.

Por su parte R. Snyder y D. MacKenzie (Snyder y MacKenzie, 2004) resaltan que la habilidad que han de poseer los agentes para organizarse en grupos abstractos

denominados comunidades, es una potente herramienta para estructurar las organizaciones de agentes en sistemas multiagente con un gran número de agentes. Estas comunidades interaccionan entre ellas y con el entorno en un tipo de organización denominado sociedad, la cual está enfocada a la consecución de objetivos globales.

En (Pavón *et al.,* 2003; Pavón *et al.,* 2005) la organización describe el entorno en el que agentes, recursos, tareas y objetivos coexisten, y ésta se define por la estructura, la funcionalidad y las relaciones sociales. Estructuralmente la organización se considera como un conjunto de entidades cuya asociación se realiza a través de relaciones de herencia y agregación y es sobre esta estructura donde se definen una serie de relaciones sociales y flujos de trabajo. La organización estructura a los agentes en grupos los cuales pueden estar formados por agentes, roles, recursos y aplicaciones siendo la asignación de los mismos un propósito organizativo que facilite la coordinación de la misma.

V. Dignum, por su parte, defiende la existencia de objetivos globales de la organización además de los objetivos individuales de los agentes, los cuales existen de forma independiente de los agentes (Dignum y Dignum, 2006). Los roles que adoptan los agentes representan posiciones en la organización que tienen la responsabilidad de concluir parte de los objetivos finales de la organización mediante una serie de reglas de interacción predeterminadas. En este caso los agentes pueden adoptar, o no, diversos roles del sistema.

En (Hubner *et al.,* 2006) se enfoca la organización como un conjunto de restricciones de comportamiento que un grupo de agentes impone a los agentes para controlar su autonomía y lograr los objetivos globales de la organización fácilmente.

En (Garijo *et al.,* 2001; Gómez-Sanz *y* Pavón*.,* 2005) se adopta la organización de forma que se identifican los tipos de agentes y recursos, sin embargo, en tiempo de ejecución esos agentes y recursos pueden ser destruidos y creados en función de las necesidades del sistema. Tomando como referencia los enfoques de los autores mencionados se pueden extraer una serie de características comunes para las organizaciones de agentes como son las siguientes:

- Una organización de agentes está compuesta por agentes, roles, y reglas de coordinación e interacción entre dichos roles.
- Persigue un objetivo común y global que no depende de los objetivos individuales particulares de los agentes.
- Existe una división de tareas a través de la asignación de roles a los agentes de forma que exista una especialización de los mismos siempre enfocada a la consecución de los objetivos globales de la organización.
- Divide al sistema en grupos mediante una departamentalización los cuales son la unidad de interacción entre los agentes.
- Define una serie de límites en cuanto a la pertenencia de los agentes a la organización, sus reglas de interacción, la funcionalidad de la misma y los servicios que ofrece.

- La entrada y salida de los agentes de la organización determina la dinamicidad de la misma, pudiendo cambiar de roles en función de los objetivos de la organización.

Si se comparan las características generales de las organizaciones humanas y de agentes, encontramos un alto grado de similitud, lo cual es lógico ya que las organizaciones de agentes nacen a partir de las organizaciones humanas. Si bien, es cierto que analizando las teorías de la organización vistas en la sección anterior, no existe una teoría que se pueda asimilar a las organizaciones de agentes de forma única, ya que en las definiciones y consideraciones sobre las organizaciones de agentes, hechas por los autores mencionados, se toman ideas de todas las teorías de la organización analizadas.

2.3.1. Las Organizaciones de Agentes como Sistemas Abiertos

La mayor parte de los sistemas multiagente son sistemas cerrados, es decir, sistemas en los que no se permite esta entrada y salida de agentes en tiempo de ejecución. En este tipo de sistemas se conoce en su fase de diseño el tipo de agentes y la cantidad de los mismos que formarán parte del sistema, así como los intereses colectivos de los mismos, por lo tanto se asume que los agentes son benevolentes (Zambonelli *et al.*, 2000). Sin embargo, la tecnología de agentes, la cual permite la formación dinámica de organizaciones de agentes, es particularmente adecuada para el desarrollo de sistemas abiertos (Rodríguez *et al.*, 2011).

Una definición de sistema multiagente abierto dada en (Hermoso *et al.*, 2010), es la de un conjunto de agentes desplegado en un entorno abierto, donde estos agentes pueden unirse o abandonar el sistema en cualquier momento a su propia voluntad. Además, los agentes pueden ser desarrollados por diferentes personas, lo que implica posiblemente diferentes objetivos y preferencias. Es por ello que se considera un sistema multiagente abierto con una estructura social, es decir, que no tienen meta global explícita.

Otra definición en la misma línea se da en (González-Palacios y Luck, 2007). Se considera un sistema abierto, al que permite movimientos de entrada y salida entre los componentes del mismo, en fase de ejecución. Es evidente que la concepción de un sistema multiagente como un sistema abierto requiere de una alta coordinación y una fuerte normalización que permita controlar este flujo de agentes. Es por ello que se ha de tener en cuenta que pueden haber sido implementados por diferentes personas que hayan utilizado distintos lenguajes de programación y protocolos de comunicación.

Al igual que en la concepción de organización humana como un sistema abierto, este tipo de sistemas abiertos se encuentran en entornos dinámicos en los que constantemente existe, de forma impredecible, un flujo de entradas y salidas de agentes y servicios que estos ofrecen (Zambonelli *et al.*, 2003). Un enfoque de este

tipo de sistemas se da en (Esteva *et al.*, 2001), a través de la asimilación del funcionamiento de las instituciones en los humanos. Este enfoque defiende la mediación de instituciones electrónicas que regulen el comportamiento de los agentes, de forma que las interacciones establecidas entre los mismos han de ser acordes a una serie de políticas y normas definidas por la institución.

En (Snyder y MacKenzie, 2004) se realiza una revisión de la arquitectura Cougaar. Esta arquitectura no es exactamente planteada como un sistema abierto, sino como una arquitectura escalable y heterogénea que a través de la interacción dinámica y controlada de grupos de agentes denominados, comunidades, emula el comportamiento de un sistema abierto.

La complejidad que existe en el diseño de una organización de agentes que funciona como un sistema abierto es alta. Esta complejidad es debida a factores como la impredecibilidad de las interacciones entre los agentes, las comunicaciones que existen entre los agentes y el comportamiento individual de estos agentes (González-Palacios y Luck, 2007), por lo que el diseño de estas interacciones entre los agentes es algo complejo que requiere una fuerte normalización y un control exhaustivo sobre los miembros de la organización. Es amplio y actual el campo de investigación que abarca este tipo de sistemas teniendo en cuenta las posibilidades que ofrecen (García-Fomes *et al.*, 2011; Argente *et al.*, 2011; Rodríguez *et al.*, 2011; d'Inverno *et al.*, 2012), sin embargo es necesario seguir trabajando en los aspectos anteriormente destacados para conseguir modelizar este tipo de organizaciones basadas en sistemas multiagente abiertos.

2.4. Diseño de las Organizaciones

Hasta ahora se han analizado las características de las organizaciones, tanto humanas como de agentes, en un marco de definición de las mismas y asimilación a las diferentes teorías de la organización existentes. Sin embargo, no se ha tratado el diseño de una organización a partir de los factores que afectan a la organización de forma específica como son: la estructura de la organización, su funcionalidad, el entorno y la evolución de la organización.

La **estructura de la organización** tiene en cuenta todos los aspectos estáticos de la misma independientemente de cuáles sean sus componentes en cada momento. Su análisis viene dado por una serie de dimensiones tomadas de (Pires y Machado, 2005).

La **funcionalidad de la organización**, hace referencia a los objetivos globales de la organización y los recursos de los que dispone para lograrlos.

El **entorno de la organización** comprende en marco de interacción de la organización con el exterior, así pues, tiene en cuenta tanto los servicios y recursos ofrecidos por la propia organización y sus consumidores, como los que necesita de entidades externas para su funcionamiento y sus proveedores.

La **evolución de la organización**, especifica la dinámica de funcionamiento de la organización a lo largo del tiempo, es decir, tiene en cuenta la creación y disolución de grupos, la entrada o salida de miembros en esos grupos y los roles que adoptan dentro de ellos.

A continuación, se analizará cada uno de estos factores en las organizaciones humanas y de agentes.

2.4.1. Estructura de la Organización

En las organizaciones humanas la estructura de la organización define la división, agrupación y coordinación formal de las tareas a realizar. Para ello en (Hodge *et al.*, 2003) se especifica el concepto de *departamentalización,* que consiste en agrupar los trabajos de forma que se puedan coordinar tareas comunes para dar lugar a grupos con características organizativas similares. Para definir el diseño de la estructura de una organización se han de tener en cuenta una serie de dimensiones como son las siguientes (Pires y Machado, 2005):

- **Amplitud de control:** Número de personas supervisadas por un jefe de equipo.
- **Amplitud vertical:** Número de niveles jerárquicos.
- **Autonomía:** Las decisiones que un grupo puede tomar sin consultar a la jerarquía.
- **Centralización:** Grado de concentración de las decisiones.
- **Complejidad**: Grado de diferenciación en profundidad jerárquica, unidades del mismo nivel y dispersión geográfica.
- **Componente** administrativa: Proporción entre el número de gestores, supervisores y personal de apoyo en relación al número total de colaboradores.
- **Diferenciación**: Proceso de división del trabajo.
- **Especialización**: Conjunto de actividades que una persona o grupo realiza de forma diferenciada.
- **Normalización**: Conjunto de normas y acciones definidas para controlar los comportamientos de los miembros de una organización. Este conjunto de normas proporciona los procedimientos necesarios a los miembros de la organización para determinar cómo realizar sus tareas.
- **Formalización**: Grado en que las actividades y sus normas están institucionalizadas.
- **Integración**: Tipo de colaboración que existe entre los diferentes departamentos.
- **Profesionalización**: Uso de códigos de asociaciones profesionales. Cuanto mayor el nivel de profesionalización, menor necesidad de formalización.

Cabe destacar la normalización como un aspecto clave dentro de una organización para obtener una coordinación óptima entre los diferentes grupos

existentes. Se definen cuatro tipos de normalización: *de tareas, de resultados, de habilidades y de comportamientos* (Wagner y Hollenbeck, 2004). La normalización de tareas tiene en cuenta la concreción de las mismas mediante los procedimientos de trabajo que se deben seguir para logar cumplir las responsabilidades de cada miembro. La normalización de resultados define formalmente los objetivos y las metas a alcanzar por los miembros de la organización. En la normalización de habilidades, se definen las habilidades, conocimientos y cualidades necesarias para desarrollar los cometidos de forma óptima. La normalización de comportamientos fija una serie de comportamientos que son compartidos por los miembros de la organización o un grupo dentro de ésta.

Por su parte las organizaciones de agentes definen su estructura en base a roles y grupos, donde los roles representan las funcionalidades de los agentes y los grupos el contexto donde se desarrollan (Dignum y Dignum, 2006), especificando las relaciones existentes entre estos roles. En (Zambonelli *et al.*, 2000) se enfoca la estructura, además de por los roles y sus relaciones, determinando los modelos de interacción entre los miembros y especificando desde donde se ejecuta la autoridad, control y supervisión de las tareas.

2.4.2. Funcionalidad de la Organización

Los propósitos de las organizaciones están reflejados en la misión, la cual describe el motivo de su existencia. Es por ello que determina una serie de objetivos y metas, traducidos a resultados en forma de productos o servicios. Los *objetivos* son los fines generales que persigue la organización, expresados de forma cualitativa, es decir, misiones, propósitos, metas, fines, cuotas y plazos. Por otra parte, las *metas*, son los fines específicos, expresados de forma cuantitativa. La fijación de metas tiene la finalidad de guiar las actividades y decisiones que realiza la organización para evitar las no dirigidas hacia los objetivos.

Los objetivos de la organización han de ser conocidos por todos sus integrantes de forma que guíen los esfuerzos de cada uno de ellos hacia la consecución de los mismos; han de proporcionar una fuente de legitimidad que determine las conductas y acciones adecuadas a la organización; establezcan los niveles mínimos o estándares que han de conseguirse; determinen, al menos de forma parcial, el tipo de estructura de la organización para la consecución de esos fines y proporcionen información sobre las características fundamentales de la propia organización (Peiro, 1991).

Las organizaciones no pueden funcionar con eficacia si el trabajo de sus empleados no tiene un carácter regular. En las empresas la gente tiene que trabajar un número de horas constante. Las actividades deben coordinarse de forma coherente en función del tiempo y del espacio, algo que se encuentra determinado por el entorno físico y por una precisa programación de los horarios (Giddens, 2000). Por otro lado las metas, consideradas como fines específicos, son identificables a partir de los objetivos generales y serán asignadas por la organización a grupos que a su vez asignarán otras metas a grupos más pequeños hasta llegar a metas

individuales, las cuales irán enfocadas a las consecución de los objetivos generales procurando la máxima productividad y eficiencia con el mínimo coste (Peiro, 1991).

En cuanto a la funcionalidad en las organizaciones de agentes sigue los mismos principios vistos para las organizaciones humanas, se definen una serie de objetivos globales de la organización y unas metas particulares, asignadas a los diferentes grupos, que guíen los esfuerzos hacia la consecución de los objetivos globales. Los servicios ofrecidos, como resultado de los objetivos de la organización, se definen como núcleos de funcionalidad que se realizan sirviendo a otras entidades u organizaciones. Detallar los servicios que ofrece una organización permite que los agentes que la integran puedan descubrirlos, invocarlos, monitorizarlos e incluso realizar asociaciones que den lugar a nuevos servicios más complejos.

2.4.3. ENTORNO DE LA ORGANIZACIÓN

El entorno de una organización tiene en cuenta todo aquello que es externo a la organización: clientes, proveedores, competencia, instituciones reguladoras de la actividad de la organización, condiciones económicas, geográficas y políticas (Wagner y Hollenbeck, 2004). Es por ello que el entorno se puede considerar como la fuente de recursos necesarios para sobrevivir, lo que implica que todas las amenazas y oportunidades de éxito provienen del entorno (Hodge *et al.*, 2003).

En las organizaciones de agentes, el entorno se generaliza principalmente con los recursos, aplicaciones y servicios de los que hacen uso los agentes de la organización, así en (Wooldridge *et al.*, 2000) se establecen los modos de acceso a los recursos y los procedimientos necesarios para solicitarlos.

2.4.4. EVOLUCIÓN DE LA ORGANIZACIÓN

La componente dinámica de cualquier organización se enfoca al movimiento de sus miembros en cuanto a la entrada o salida de los mismos en función de determinadas habilidades y determinadas necesidades de la propia organización en un momento dado (Peiro, 1991).

Es por ello, que en las organizaciones de agentes se deben institucionalizar mecanismos de control en base a la entrada o salida de agentes de la organización, además de controlar la asignación de roles entre sus integrantes. Así pues, lo aplicable a cada miembro de la organización, también ha de aplicarse a los diferentes grupos a los que da lugar la departamentalización, o división, que se da en las estas organizaciones.

Todos estos aspectos vistos sobre diseño de las organizaciones y los factores más importantes que se han de tener en cuenta, en su mayor parte, coinciden tanto en organizaciones humanas como en organizaciones de agentes. El diseño de organizaciones de agentes teniendo en cuenta estos factores se desarrolla a través de diferentes metodologías y herramientas de modelado de organizaciones de agentes, como pueden ser INGENIAS (Gómez, 2002), AML (Cervenka y Trencansky, 2007),

AOPOA (Rodríguez *et al.*, 2007), las cuales se amplían detalladamente en el capítulo 4 de esta memoria.

2.5. TIPOS DE ORGANIZACIONES

El diseño de una organización ha de tener en cuenta todos los factores comentados en la sección anterior. Son estos factores los que determinan su tipología en cuanto a su extensión, organigrama, reglas de coordinación, normas a establecidas, etc. El tipo de organización viene influenciada por las teorías vistas en este capítulo, las cuales determinan la importancia de unos factores u otros en función de los objetivos generales y como conseguirlos. En (Argente, 2008) se presenta un completo estudio sobre la taxonomía de las organizaciones tanto de humanos como de agentes, indicando las ventajas y los inconvenientes de cada una de ellas según diferentes expertos de la literatura. En la **Tabla 2.1** se exponen los tipos de organizaciones humanas en base a (Mintzberg, 1989; Robbins, 2004) y se resaltan sus principales ventajas e inconvenientes.

Tabla 2.1 Tipología de las organizaciones humanas

Estructura	Características	Ventajas	Inconvenientes
Simple	-Poca departamentalización -Centralización de decisiones -Gran tramo de control -Estructura plana -Escasa formalización	-Simple -Fácil de mantener -Responsabilidades claras -Comunicaciones rápidas -Decisiones rápidas	-Poca formalización -Elevada centralización -Sobrecarga de información directiva -Dependencia de un individuo
Burocracia	-Funcional y divisional -Tareas operativas rutinarias -Alta especialización -Alta formalización -Alta departamentalización -Autoridad centralizada -Cadena de mando	-Actividades eficaces -Comunicación con los empleados -Decisiones centralizadas	-Muchos niveles de gerencia -Alta reglamentación -Priorización de objetivos departamentales -Relaciones jerárquicas y rígidas -Dificultad de respuesta ante cambios en el entorno
Matricial	-Dos cadenas de mando -Flexibilidad -Priorización de objetivos organizacionales -Ubicación de especialistas	-Mejora la coordinación -Mejora la comunicación	-Puede haber confusión -Existen luchas de poder -Se producen tensiones
Equipos	-Elimina las barreras entre los departamentos - Flexibilidad y consenso en toma de decisiones -Priorización de objetivos globales - División del trabajo en grupos especializados	-Mayor motivación -Incremento de la productividad -Metas que mejoran objetivos generales -Reducción de costes administrativos	-Modelo de trabajo colaborativo -Se ha de proporcionar total información y recursos a los equipos -Se necesita más tiempo en la coordinación de equipos

			-Necesidad de normas
Adhocracia	-Constelaciones de trabajo -Flexibilidad y dinamismo en el sistema -Cada constelación trata objetivos independientes	-Poder descentralizado -Estrategia bien definida -Ejecución de tareas más rápida	-Medios para alcanzar objetivos poco definidos -Necesidad de negociación en la coordinación -Necesidad de normas
Organización Virtual	-Contratación de funciones -Descentralización -Poca departamentalización	-Flexibilidad -Capacidad de adaptación a cambios en el entorno	-Poco control directivo -Complejidad -Control completamente distribuido

La tipología de las organizaciones aplicada a los sistemas multiagente no tiene una correspondencia exacta, así pues, B. Horling, describe los diferentes paradigmas organizacionales empleados en los sistemas multiagente, que incluyen jerarquías, holarquías, coaliciones, equipos, congregaciones, federaciones y organizaciones matriciales (Horling y Lesser, 2004).

Tabla 3.2. Tipología de las organizaciones de agentes

(*) El término Holón representa una entidad que existe de forma simultánea como parte de una entidad mayor y como resultado de la agrupación de entidades subordinadas

Estructura	Características	Ventajas	Inconvenientes
Jerarquías (Yadgar *et al.*, 2003)	-Estructura tipo árbol -Interacción entre entidades conectadas	-Existe un sistema de control -Pueden existir tantos niveles como sea necesario -Se descompone el problema en subproblemas resueltos por agentes simples	-No existe comunicación bidireccional entre agentes en el flujo de datos -Comunicaciones unidireccionales en el flujo de control
Holarquías (Fischer, 1999) (Zhang y Norrie, 1999) (Ulieru *et al.*, 2001) (Giret, 2005)	-Se basan en el concepto de holón(*) -Estructuras jerárquicas y anidadas de holones	-Habilidad para modelar sistemas de negocio y manufactura -Eficaces en la división recursiva de subtareas	-No existe comunicación bidireccional entre agentes en el flujo de datos
Coaliciones (Sheory y Kraus, 1998) (Lerman y Sheory, 2000) (Sho *et al.*, 2003)	-Agrupaciones temporales -Corta duración -Dirigidas a un objetivo concreto -Estructura plana	-Ahorran recursos -Se especializan en un objetivo determinado	-No existe un control constante sobre la coalición que la disuelva en caso de comportamientos negativos de agentes
Equipos (Jennings, 1995) (Tambe, 1997)	-Aborda problemas mayores que un agente individual -Unión de varios agentes de forma cooperativa -Un agente puede jugar varios roles	-Permite aspectos de redundancia -Mayor adaptación a restricciones globales -Mejor aprovechamiento de recursos -Reducción de costes	-Gran número de comunicaciones entre los agentes
Congregaciones (Brooks y Durfee, 2000)	-Grupos de agentes con características similares	-Se reduce la complejidad de búsquedas de colaboradores	-Debe existir un número estable de participantes

(Brooks y Durfee, 2003)	-Se crean para largo plazo -No conllevan la consecución de un objetivo específico -Mecanismo para que poblaciones grandes de agentes se organicen en pequeños grupos	-Los agentes se unen o abandonan la congregación de forma dinámica -Optimizan el proceso de comunicación	-Sus miembros se organizan de forma diferente para facilitar su coordinación
Federaciones (Maturana *et al.*, 1990) (Shen y Norrie, 1998)	-Un agente delegado representa al grupo	-Facilidad en el control y monitorización de la organización	-La comunicación con el entorno se realiza a través del agente delegado -Centralización en un agente
Matriciales (Decker *et al.*, 1995) (Horling *et al.*, 2003)	-Múltiples líneas de autoridad	-Existe mucho control sobre los agentes -Un agente comparte su habilidad con varias líneas de autoridad	-Decisiones tomadas por los agentes supervisores contradictorias

En la **Tabla 2.2** se muestran las características principales, las ventajas e inconvenientes de este tipo de organizaciones de agentes, así como referencias de autores que han hecho uso de las mismas, tomados de (Argente, 2008).

Las similitudes existentes entre las organizaciones humanas y de agentes pone de manifiesto la intencionalidad de simular los procesos de asociación humanos en sistemas multiagente. Todos estos paradigmas organizacionales poseen una serie de ventajas e inconvenientes, los cuales tendrán mayor trascendencia dependiendo de los objetivos globales de la propia organización. Este hecho ya se ha comentado anteriormente cuando hablábamos de las organizaciones en humanos, lo que implica que no existe un tipo de organización de agentes óptima para la resolución de cualquier problema.

2.5.1. Comunidades como Tipo de Organización

El sentido psicológico de comunidad es una experiencia subjetiva de pertenencia a una colectividad mayor, formando parte de una red de relaciones de apoyo mutuo en la que se puede confiar, es decir *"una red de relaciones de apoyo mutuo de la que uno puede depender"* (Sarason, 1974). Los matices que se formulan en esta definición son: la percepción de similitud con otros, el reconocimiento de la interdependencia con los demás, la voluntad de mantener esa interdependencia dando o haciendo por otros lo que uno espera de ellos, y el sentimiento de que uno es parte de una estructura más amplia, estable y fiable (Sánchez Vidal, 2001).

Un autor en referencia a la teoría de la organización no mencionado anteriormente es H. Nicklisch, que elabora su teoría en el marco del concepto de *"Comunidad"* (Nicklisch, 1928). El concepto de comunidad se diferencia del de organización en que es más espontáneo y natural. La comunidad se entiende como algo vivo, dinámico, orgánico que surge por hábitos y costumbres afincados en la

tradición. Para Nicklisch, la comunidad se apoya en los intereses comunes de los que trabajan en un centro, y para llevar a cabo su regulación exige un sistema contractual de cogestión (Nicklisch, 1932; Larsen, 2000).

Según Nicklisch una organización, en forma de comunidad, ha de tener las siguientes características (Nicklisch, 1932):

- **Unidad y articulación**, de forma que una comunidad ha de poseer capacidad de decisión, cohesión de sus miembros y una comunicación y coordinación fluida.
- **División del trabajo y agrupación**, lo cual aporta una especialización de tareas que se traduce en una mejora de la producción.
- **Ley económica**, que exige la obtención del máximo rendimiento en base a un objetivo dado.
- **Configuración del conjunto**, para establecer una metodología de trabajo y las relaciones entre sus miembros.
- **Mantenimiento de dicho conjunto**, que aporta la necesidad de cohesión y sentimiento grupal para la no disolución de la comunidad.

Para Nicklisch el valor social que ha de tener una organización está por encima de cualquier otro valor, haciendo valer por lo tanto, a los integrantes de la organización por encima de cualquier otra cosa, y presuponiendo que estos integrantes son capaces de sentirse miembros de la organización y por lo tanto aunarán esfuerzos para llevar a cabo la consecución de los objetivos de la misma. Esto hace plantearse a Nicklisch, que el camino para organizar hacia adelante no es el marcado por la técnica sino por las necesidades internas de la conciencia que nos llevan a vivir organizadamente en comunidad (Nicklisch, 1920; Larsen, 2000).

2.5.2. Comunidades de Agentes como Tipo de Organización

En los trabajos de (Glinton *et al.*, 2010; Parachuri *et al.*, 2010) el término comunidad se utiliza para definir a un grupo de agentes heterogéneos. Otro enfoque de comunidad de agentes se da en (Snyder y MacKenzie, 2004), en el que se define la estructura de las Comunidades de agentes Cougaar.

Cougaar es una arquitectura basada en el lenguaje de programación Java para la construcción de aplicaciones basadas en sistemas de agentes que integren un gran número de estos (BBN Technologies, 2004). El conjunto global de agentes de esta arquitectura que interactúa para la consecución de un objetivo global, es denominado sociedad. Una comunidad Cougaar es un concepto que denota un grupo de agentes con objetivos comunes y una relación organizacional dentro de la sociedad (Snyder y MacKenzie, 2004).

Las Comunidades Cougaar proveen un modo de organización de las sociedades en diferentes grupos de agentes o comunidades, es decir, una comunidad

pude estar compuesta por agentes integrantes de la sociedad y por otras comunidades, las cuales se definen de forma estática en tiempo de diseño (Snyder y MacKenzie, 2004). Este proceso es transparente para los agentes, ya que la infraestructura de las comunidades integra a los agentes correspondientes en dichas comunidades al inicio. También existe la posibilidad de creación de comunidades de forma dinámica, para lo cual hace falta que un agente con responsabilidad suficiente lo haga de forma explícita.

En las Comunidades Cougaar, los agentes que la integran tienen la posibilidad de adoptar múltiples roles, los cuales representan la funcionalidad de los agentes dentro de la comunidad. El control de los agentes en las comunidades Cougaar se realiza por un agente designado para ello, sin embargo, si la comunidad se crea en tiempo de ejecución, el agente responsable de la creación de dicha comunidad es el responsable de los agentes de la misma y de su correcto funcionamiento.

Finalmente se resaltan las características más importantes de las comunidades Cougaar (Snyder y MacKenzie, 2004).

- La interacción de los miembros de la comunidad es multidireccional.
- Un agente perteneciente a una comunidad puede adoptar múltiples roles.
- La comunidad permite la creación, eliminación de comunidades de forma dinámica, además de operaciones de entrada, salida y modificación de agentes de las comunidades, (la modificación se refiere a los múltiples roles que un agente puede adoptar).
- Las comunidades tienen la capacidad de operar en entornos distribuidos.
- Las comunidades poseen mecanismos de coordinación entre agentes.
- Las comunidades son escalables para la resolución de problemas complejos.
- En las comunidades existe jerarquización.
- En estas comunidades puede haber anidamiento de comunidades.

Teniendo en cuenta la definición de las comunidades Cougaar y sus características, a continuación se presentará la definición de **Comunidades Inteligentes (CI)**. Estas comunidades adoptan características definidas en otro tipo de organizaciones de agentes además de las Cougaar, y tienen en cuenta conceptos claves derivados de las diferentes teorías de la organización introducidas en este capítulo.

Las Comunidades Inteligentes consisten en un grupo reducido de agentes cooperativos que trabajan juntos en la consecución de un objetivo común. Basándonos en la teoría de Nicklisch, un agente miembro de una comunidad ha de tener sentimiento de la misma, es decir, ha de buscar el bien de la comunidad por encima del suyo propio. Es por ello que se ha de mantener una representación explícita de los objetivos, creencias y planes a nivel de comunidad. En la **Figura 2.2.** se representa de forma gráfica una comunidad de agentes.

Las Comunidades Inteligentes se componen de un número pequeño de agentes, de forma que, la comunicación entre los agentes sea multidireccional y no supongan un esfuerzo computacional añadido. Su fundamentación se basa en la de las estructuras organizacionales de equipos, es decir, un grupo de agentes es capaz de abordar problemas de mayor envergadura que si actuasen de forma individual, permitiendo aspectos de redundancia, mayor adaptación al entorno y realizando un mejor aprovechamiento de los recursos. La toma de decisiones está centralizada en cuanto a las decisiones globales de la comunidad y será autónoma para cada agente en las labores que desempeñen individualmente, siempre en beneficio de la comunidad. Para ello existe una formalización de las tareas a realizar dotando así a los agentes de una herramienta para la toma de decisiones individuales.

El papel desempeñado por los agentes dentro de una Comunidad Inteligente, ha de ser estático, es decir, los agentes de una comunidad juegan un rol determinado sin posibilidad de adoptar otro tipo de rol, lo cual dota de una especialización a los agentes. El objetivo global de la comunidad se dividirá en subobjetivos, asignado cada uno de ellos a los agentes integrantes de la comunidad. En el caso de necesidad de alguna funcionalidad determinada se daría entrada a otro agente que jugara el rol necesario para disponer de esa funcionalidad. La especialización también existe a nivel de comunidad de forma que los objetivos perseguidos por la comunidad han de encontrarse en un mismo contexto.

La dinamicidad de la Comunidad Inteligente, en cuanto a la entrada y salida de sus miembros, y su asociación interna, por una parte permite la agrupación interna de sus miembros en pequeños equipos de trabajo, con el fin de coordinar tareas similares, y por otra parte pretende afincar a los agentes, y equipos, sin la posibilidad de dar entrada a otros agentes. Cabe destacar en este aspecto que se permite la escalabilidad de comunidades en el caso de la existencia de grandes problemas globales, funcionando así como un sistema abierto (González-Palacios y Luck, 2007). Además, se ha de tener en cuenta que cuando un miembro de la comunidad no realiza su trabajo correctamente o incumple las normas de la comunidad puede ser expulsado de la misma reemplazándolo por otro miembro de las mismas características.

En una Comunidad Inteligente existirá una jerarquía en dos niveles en cuanto a las labores de control de la comunidad. Uno de los agentes integrantes de la misma ejercerá las labores de control (Dignun y Dignum, 2012) sobre los demás integrantes en base a una serie de reglas comunitarias que aseguren el buen funcionamiento de la misma. Esto es debido a que, aunque Nicklisch presupone benevolentes a los integrantes de la comunidad, en una comunidad de agentes hemos de tener en cuenta las metas individuales y las desviaciones que puedan darse sobre los objetivos globales de la comunidad.

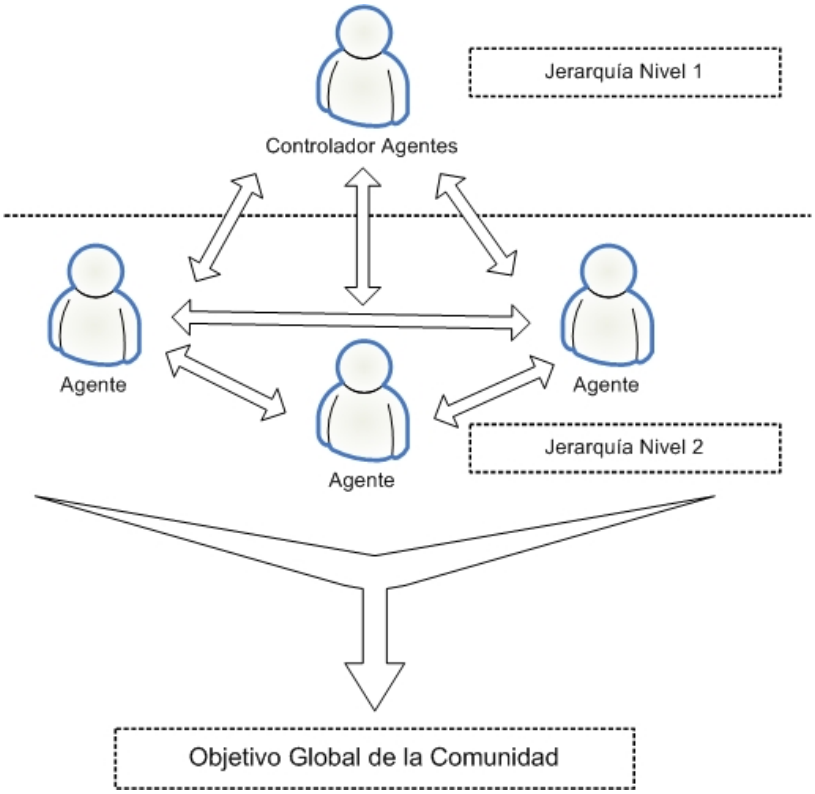


Figura 2.2 Representación gráfica de una Comunidad Inteligente (CI)

Finalmente, las Comunidades Inteligentes han de poseer la capacidad de funcionar de forma distribuida a nivel de comunidad y los servicios que provean no deben estar embebidos en los miembros de la misma, sino que han de ejecutarse de forma distribuida, con el fin de liberar computacionalmente a la estructura comunitaria y a los agentes miembros.

2.6. Comparativa entre Comunidades de Agentes y Tipos de Organizaciones

A continuación se realiza una comparativa entre los distintos tipos de organizaciones de agentes, incluyendo la propuesta de Comunidad Inteligente como tipo de organización, en base varias de las dimensiones estructurales consideradas en (Pires y Machado, 2005), y adaptadas a las organizaciones de agentes, la cual puede verse en la **Tabla 2.3.**

- **Amplitud de control:** Número de agentes supervisados por un jefe de equipo.

- **Amplitud vertical:** Número de niveles jerárquicos.
- **Autonomía:** Las decisiones que un grupo de agentes puede tomar sin consultar a la jerarquía.
- **Centralización:** Grado de concentración de las decisiones organizacionales.
- **Complejidad**: Grado de diferenciación en profundidad jerárquica teniendo en cuenta, las unidades del mismo nivel.
- **Diferenciación**: Proceso de división del trabajo entre grupos de agentes o departamentos.
- **Especialización**: Conjunto de actividades que agente o grupo de agentes realiza de forma diferenciada.

Tabla 2.3 Comparación de las dimensiones estructurales en los tipos de organizaciones de agentes

(*) Las coaliciones en determinadas ocasiones pueden disponer de un agente líder que actúa como representante del grupo.

	Organizaciones de Agentes							
Características	Jerarquía	Holarquía	Coalición	Equipo	Congregación	Federación	Matricial	CI
Amplitud de control	Alta	Alta	Nula(*)	Nula	Media	Alta	Alta	Baja
Amplitud vertical	Variable	Variable	Nula	Nula	Variable	Baja	Alta	Baja
Autonomía	Nula	Nula	Alta	Alta	Variable	Baja	Baja	Alta
Centralización	Alta	Alta	Nula	Nula	Variable	Alta	Alta	Alta
Complejidad	Alta	Alta	Alta	Baja	Alta	Alta	Alta	Baja
Diferenciación	Nula	Alta	Alta	Nula	Alta	Nula	Nula	Alta
Especialización	Alta	Alta	Alta	Alta	Alta	Alta	Alta	Alta
Normalización	Alta	Alta	Baja	Baja	Media	Alta	Alta	Alta
Formalización	Alta	Alta	Baja	Baja	Media	Alta	Alta	Alta
Integración	Nula	Alta	Alta	Nula	Alta	Nula	Nula	Alta
Dinamicidad	Baja	Baja	Alta	Baja	Alta	Baja	Baja	Alta

- **Normalización**: Conjunto de normas y acciones definidas para controlar los comportamientos de los agentes de una organización
- de forma supervisada. Este conjunto de normas proporciona los procedimientos necesarios a los miembros de la organización para determinar cómo realizar sus tareas.
- **Formalización**: Grado en que las actividades y sus normas están institucionalizadas.
- **Integración**: Tipo de colaboración que existe entre los diferentes departamentos.

- **Dinamicidad**: Especifica la capacidad de la organización para gestionar la entrada y salida de agentes de la organización en un momento dado. Esta dimensión no viene dada en (Pires y Machado, 2005), sin embargo es de especial relevancia debido a la necesidad de conocer el funcionamiento y la estabilidad de la organización en función de la entrada y salida de nuevos agentes.

Entre las diversas tipologías de organización de agentes existe cierta similitud en cuanto a las dimensiones propuestas para su comparación. En la adaptación de la comunidad como tipo de organización a los sistemas multiagente, se ha buscado que cumpla con las características necesarias que la doten de flexibilidad, interacción y coordinación entre sus miembros, dinamicidad, especialización y escalabilidad. Estas características dotarán de mayor adaptación de este tipo de organización a los sistemas multiagente. Además, de forma general puede decirse que todas las tipologías de organización tienen un punto en común: la coordinación de la conducta de los agentes integrantes de la organización. Los agentes pueden coordinar sus conocimientos, objetivos, habilidades, tareas y planes de forma conjunta para tomar una acción o resolver una meta global (Rodríguez, 2010).

2.7. Conclusiones

La organización, de forma general, se crea para satisfacer una serie de objetivos generales los cuales han de ser perseguidos por todos los integrantes de la misma. En esta línea, es la organización la que tiene que proveer a sus integrantes de los mecanismos necesarios para lograr estos objetivos generales. Esta visión de organización se da tanto en organizaciones humanas como en organizaciones de agentes, lo cual refleja la similitud que existe entre ambas.

A lo largo de este capítulo se vienen reflejando las diferencias existentes entre los tipos de organizaciones. El estudio de las diferentes teorías de la organización así como sus características, ha permitido comprender mejor estas diferencias a partir de los factores que afectan a las organizaciones humanas, y por lo tanto, trasladarlas a las organizaciones de agentes.

En función de los objetivos que persiga la organización, primarán una serie de factores que determinarán el diseño de la misma. Estos factores son la estructura de la organización, su funcionalidad, el entorno en que desarrollará su actividad y su evolución. Es por ello que no existe un tipo de organización estándar, que tenga la capacidad de distribuir a sus miembros y a sus recursos de manera eficiente para cualquier problema dado, lo que justifica las diferentes visiones de la organización por parte de los autores citados.

La aplicación de un diseño organizacional a los sistemas multiagente resulta de gran utilidad, ya que controla las interacciones de los agentes proporcionando una serie de normas enfocadas a una buena coordinación, y a alcanzar los objetivos generales de la propia organización (Gasser, 2001; Hubner *et al.*, 2006).

Además la organización en los sistemas multiagente proporciona la división de tareas en forma de departamentalización (Ferber *et al.*, 2003). El hecho de descomponer problemas complejos en subproblemas más simples dota de mayor potencia al sistema que tendrá la capacidad de resolver simultáneamente dichos subproblemas optimizando así el tiempo empleado.

Teniendo en cuenta las ideas de Nicklisch sobre su teoría de la organización (Nicklisch, 1928) y la definición de las Comunidades Cougaar, se ha propuesto la *"Comunidad Inteligente"* como tipo de organización. Esta propuesta se debe a que, al no existir una tipología óptima de organización de sistemas multiagente que se adapte a la arquitectura objetivo de este trabajo, su inclusión como organización en el desarrollo de sistemas multiagente, se adecua completamente a la solución buscada, dotando de mayor facilidad su diseño y optimizando su funcionamiento.

CAPÍTULO 3. ESPECIALIZACIÓN

3. ESPECIALIZACIÓN

En este capítulo se hace referencia al concepto de especialización como una mejora en los procesos productivos de un grupo. Esta especialización, se pretende llevar a cabo en los sistemas multiagente con una estructura organizacional en forma de Comunidad Inteligente, con el fin de mejorar su diseño y escalabilidad en su desarrollo. Para alcanzar este objetivo se ha de introducir el concepto de especialización y se ha analizado el comportamiento de esta especialización, en diversos tipos de organizaciones.

3.1. INTRODUCCIÓN

La especialización es una característica que dota a un individuo, perteneciente a un grupo, de una particularidad dentro de ese grupo. Esta especialización se observa dentro de las organizaciones como un mecanismo para conseguir una mayor eficiencia en los objetivos perseguidos por la misma.

Así pues, en los sistemas biológicos, la especialización se ha manifestado de dos formas: en cuanto a la estructura del sistema (Wenseleers *et al.,* 2003) y en cuanto a su comportamiento (Bonabeau y Theraulaz, 1999). Por ejemplo, la especialización estructural puede observarse en las colonias de termitas (Noirot y Pasteels, 1987) y la especialización en cuanto al comportamiento del sistema biológico se da en las abejas, que adaptan su comportamiento de forma dinámica en función de lo que necesiten recolectar, polen, agua, o néctar (Calderone y Page, 1988). El objetivo perseguido con esta especialización es la eficiencia en la consecución de los objetivos globales de las colonias a través de la división de las tareas en grupos especializados.

La especialización en los sistemas biológicos ha sido tomada como referencia en las organizaciones humanas. Según R. Melinkoff, el principio de la especialización parte de que, las actividades de cada uno de los miembros de un grupo organizado deberán confinarse, en todo lo que sea posible, a la ejecución de una sola tarea (Melinkoff, 1969). Una definición de especialización, referida al contexto organizacional humano, la encontramos en el "Diccionario de Economía Política" (Borisov *et al.,* 1975): La especialización es una de las formas de la división social del trabajo, tanto entre distintas ramas de la industria y de la producción agrícola, como en el interior de una rama en los diferentes estadios de elaboración del material. La especialización permite que se organice mejor la gran producción en cadena e incrementa en gran medida la productividad del trabajo social.

La especialización, en el campo de los agentes, ha sido tratada en profundidad por (Theraulaz *et al.,* 1991a; Theraulaz *et al.,* 1991b; Murciano *et al.,* 1997; Li, *et al.,* 2002; Li, *et al.,* 2004; Lei, *et al.,* 2007; Okamoto, *et al.,* 2008) entre otros. Esta

especialización pretende dotar de una mayor eficiencia al sistema multiagente, de forma que exista una mejora en la consecución de sus objetivos globales.

Un aspecto a tener en cuenta es el tipo de especialización necesaria que se adecua a un grupo u organización. Esta tipología ha de implementar una mejora en sus procesos y en los objetivos. Estos tipos de especialización, tanto para organizaciones humanas, como para sistemas multiagente, serán tratados de forma detallada en la sección 3.2. de este capítulo.

Una vez definido el concepto de Comunidad Inteligente**,** en el capítulo 2, como núcleo organizativo que se utilizará para la definición de la arquitectura propuesta en esta tesis, y teniendo en cuenta su tratamiento como organización, parece lógico buscar un tipo de especialización adecuada a la misma, para incrementar su eficiencia y optimizar sus procesos, en la consecución de su objetivo final.

La estructura de este capítulo es la siguiente. En la sección 3.2. se introduce el concepto de especialización, y se realiza un estudio de los tipos de especialización en organizaciones humanas y en sistemas multiagente. A continuación, en la sección 3.3. se detalla la aplicación del concepto de especialización en las Comunidades Inteligentes, previamente definidas en el capítulo 2 de esta memoria. Finalmente, en la sección 3.4. se presentan las conclusiones obtenidas de este capítulo.

3.2. Tipos de Especialización

Cuando decimos que un recurso está más especializado, solemos hacer referencia a que es más idóneo para elaborar un determinado producto o para prestar un cierto servicio, ya que posee ventajas comparativas en su producción (Arruñada, 1990). La especialización, como se ha determinado en la sección anterior, es aplicable a todo tipo de sistemas grupales cooperativos. A lo largo de esta sección se definirán los tipos de especialización aplicables a las organizaciones humanas y a los sistemas multiagente vistos como organizaciones.

3.2.1. Tipos de Especialización en Organizaciones Humanas

La especialización en las organizaciones humanas la podemos ver referida a las tareas que se desarrollan dentro de la misma, o como la definición del objetivo global de la organización en un contexto reducido. Esta última visión da lugar a agrupaciones de organizaciones, denominadas redes empresariales (Becerra, 2008), que en su conjunto buscan lograr un mayor beneficio de forma sinérgica.

3.2.1.1. ESPECIALIZACIÓN DE TAREAS

H. Mintzberg (Mintzberg, 1989) define la especialización de tareas como el número de tareas diferentes que desarrolla un puesto de trabajo y con qué frecuencia se repiten, y al poder de decisión que ejerce el puesto que ocupa el cargo sobre el diseño del mismo. La especialización de tareas, a su vez, se divide en

especialización de tareas horizontal y especialización de tareas vertical. Esta clasificación ha sido tomada de (Mintzberg, 1989).

La especialización de tareas horizontal tiene una estrecha relación con la división del trabajo y denota el número de tareas que ha de realizar un determinado puesto de trabajo siendo su principal objetivo el aumento de la productividad. Esta especialización regula la amplitud del puesto de trabajo y puede ser alta, cuando el puesto de trabajo ha de desarrollar un número pequeño de tareas repetitivas, y baja, si el puesto de trabajo ha de desarrollar muchas tareas no repetitivas. Por otra parte, la especialización de tareas vertical, hace referencia a la diferenciación entre la ejecución de las tareas y, la planificación y control de las mismas. En este caso, también puede considerarse alta cuando un puesto de trabajo solamente ejecuta tareas que han sido planificadas por otra persona, y baja cuando las tareas a ejecutar no son planificadas por otra persona, sino por iniciativa propia.

La relación existente entre estos dos tipos de especialización es muy estrecha, ya que, un puesto de trabajo en el que exista una alta especialización horizontal necesita también de una alta especialización vertical, porque la persona que ejecuta una tarea determinada puede perder la visión global de los objetivos, y por lo tanto es necesario que otra persona, que posea esa visión, planifique, organice y controle dicho trabajo (Mintzberg, 1989).

Así pues, en función de lo expuesto anteriormente podemos medir la especialización de tareas de la siguiente forma (Román et al., 2013):

$$\begin{cases} N_T \neq 0 \;\; número\ de\ tareas\ que\ realiza\ una\ persona \\ P = 1 \;\; puesto\ de\ trabajo\ determinado \end{cases} \tag{3.1}$$

$$E_T = \frac{P}{N_T} \tag{3.2}$$

Donde E_T es el grado de especialización de tareas en un determinado puesto de trabajo. Definiendo E_{TH} como el grado de especialización de tareas horizontal, tendremos que:

$$\begin{cases} Si\ E_T \geq 0.2, \rightarrow \;\; E_{TH}\ es\ muy\ alto \\ Si\ 0.2 > E_T \geq 0.1, \rightarrow \;\; E_{TH}\ es\ alto \\ Si\ 0.1 > E_T \geq 0.05, \rightarrow \;\; E_{TH}\ es\ bajo \\ Si\ E_T < 0.05, \rightarrow \;\; E_{TH}\ es\ muy\ bajo \end{cases} \tag{3.3}$$

Sea N_{TP} el número de tareas que realiza una persona y que han sido planificadas, y son controladas por otra persona, y E_{TV} el grado de especialización vertical en un determinado puesto de trabajo. Tendremos que:

$$\begin{cases} N_{TP} \neq 0; \\ si\ N_{TP} = 0, se\ considera\ que\ no\ existe\ especialización\ vertical \\ N_{TP} \leq N_T \end{cases} \tag{3.4}$$

$$E_{TV} = \frac{P}{N_{TP}} \tag{3.5}$$

Donde obtenemos los siguientes resultados:

$$\begin{cases} Si\ E_{TV} \cong E_T \ \rightarrow \ E_{TV}\ es\ muy\ alto \\ Si\ E_{TV} > E_T \ \rightarrow \ E_{TV}\ es\ \ alto \\ Si\ E_{TV} \gg E_T \ \rightarrow \ E_{TV}\ es\ bajo \end{cases} \tag{3.6}$$

A través de estas ecuaciones podemos medir el grado de especialización de tareas en un determinado puesto de trabajo. Además, se ha tenido en consideración la particularidad de que varios puestos de trabajo realicen una misma tarea, lo cual implicaría una especialización de tareas muy alta ya que la productividad crecería. En este supuesto se ha de resaltar que no todas las tareas en un puesto de trabajo permiten que varias personas puedan realizarla en paralelo. La definición de estos valores umbrales se ha basado en la observación de diferentes puestos de trabajo, desde operarios en una cadena de producción, hasta especialistas. Para ello también se han tenido en cuenta los trabajos (Rushing, 1967; Samuel y Mannheim, 1970; Reimann, 1974) donde se toman los resultados obtenidos y se asimilan a nuestro problema en particular.

3.2.1.2. REDES EMPRESARIALES

El enfoque de especialización organizativa como unidad se puede ver a través de las redes empresariales. En la literatura, el concepto de red empresarial lo encontramos en trabajos de autores como (Sonquist y Koenig, 1975; Galaskiewicz, 1979; Burt, 1980; Walker *et al.*, 1997; Lazer, 2003; Borgatti y Foster, 2003; Tracey y Clark, 2003; Johansson y Quigley, 2004; Viedma, 2004; Pöyhönen y Smedlund, 2004; Ibarra *et al.*, 2005; Eraydin y Armatli-Köroglu, 2005; Cabus y Vanhaverbeke, 2006) citados en (Becerra, 2008), para los 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.

Así pues, una red empresarial puede considerarse como una asociación de empresas que colaboran conjuntamente, de forma que exista una complementación especializada entre ellas, con el objetivo de poder resolver situaciones que individualmente no podrían, de la forma más eficiente posible.

Según (López, 2003) los objetivos que persigue una red empresarial son:

- Elevar la competitividad y la rentabilidad de las empresas de la red.
- Inducir la especialización de las empresas en algunas de las diferentes etapas del proceso productivo.
- Consolidar la presencia en el mercado de las empresas que integran la red.

- Facilitar el acceso de las empresas a servicios que les resultan inaccesibles de manera individual.

C. López, considera primordial el número de integrantes de una red empresarial para que exista eficiencia en las operaciones desarrolladas, es decir, ha de estar constituida por un número limitado de empresas, no más de 15 o 20, y bien definido (López, 2003). Dentro de las redes empresariales existen dos tipos de modalidad en cuanto a su cooperación se refiere: redes horizontales empresariales y redes verticales empresariales (Becerra, 2008) como se muestra en la **Figura 3.1.**

Las redes horizontales son alianzas entre grupos de empresas que ofrecen un mismo servicio o producto. La cooperación entre estas empresas se da en una serie de actividades, sin embargo compiten en el mismo mercado. Un ejemplo de este tipo de red se daría en dos empresas las cuales fabricaran neumáticos. Es evidente que la asociación de las empresas frente a los proveedores de materias primas beneficiará a todas en cuanto al volumen de las mismas y por lo tanto en su precio.

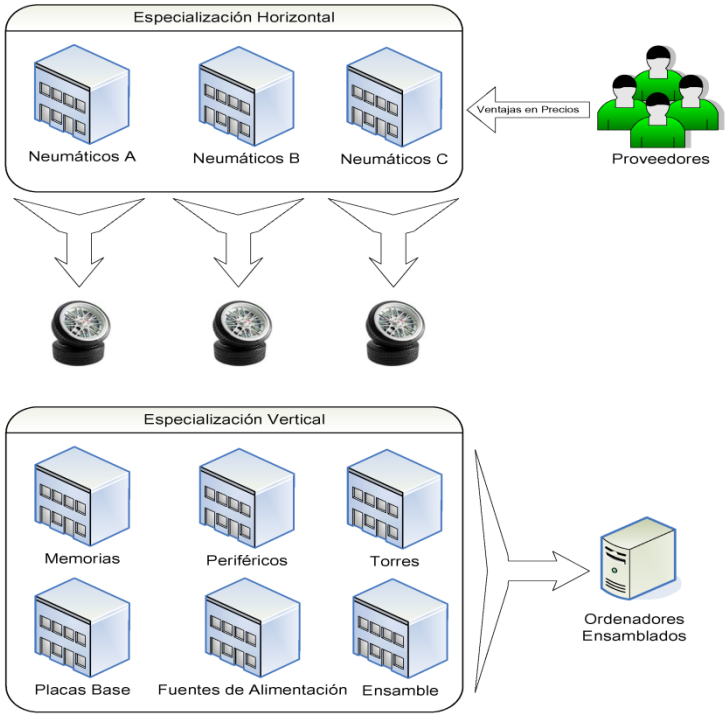


Figura 3.1 Redes Empresariales con especialización vertical y horizontal

Por otra parte, las redes verticales se corresponden a las alianzas entre empresas diferentes que se asocian para alcanzar ventajas competitivas que serían inviables de forma individual. Así pues, un ejemplo lo tendremos en varias empresas

que se dediquen a fabricar componentes informáticos. La unión vertical de estas empresas puede llevar a cabo la comercialización de ordenadores completos y ensamblados, lo cual abre un nuevo mercado que beneficia a todas ellas.

Como en el caso de las especialización de tareas, también podremos medir el grado de especialización en dentro de una organización y, dentro de una red empresarial, el grado de verticalidad y horizontalidad que alcanza la cooperación de sus empresas (Román et al., 2013).

Sea N_{TE} el número de grupos de tareas que se desempeñan en una organización. Entendemos por grupo de tareas, las diferentes labores como pueden ser administración, comercialización, etc., además de las relativas a las diferentes tareas de producción. Y N_E, el número de empleados. Entoces:

$$E_E = \frac{N_{TE}}{N_E} \tag{3.7}$$

Donde E_E es el grado de especialización de empresarial. E_E puede tomar los siguientes valores:

$$\begin{cases} Si\ E_E > 1, \rightarrow\ \ E_E\ es\ muy\ bajo \\ Si\ 1 \geq E_E > 0.75, \rightarrow\ \ E_E\ es\ bajo \\ Si\ E_E \leq 0.75, \rightarrow\ \ E_E\ es\ alto \\ Si\ E_E \cong 0, \rightarrow\ \ E_E\ es\ muy\ alto \end{cases} \tag{3.8}$$

Ahora bien, una vez definido el grado de especialización empresarial, nos interesa cuantificar de algún modo el grado de cooperación, tanto vertical, como horizontal, relativo a un conjunto de empresas que forman una red empresarial. Para ello definimos los siguientes indicadores. Sea G_{CV} el grado de cooperación vertical y G_{CH} el grado de cooperación horizontal dentro de la red empresarial, en función de los términos definidos anteriormente. Teniendo en cuenta lo expuesto en (López, 2003), las redes empresariales no deben sobrepasar un número determinado de integrantes hasta un umbral de 15 o 20. Entonces:

$$\begin{cases} N_{EM} > 2, es\ el\ número\ de\ organizaciones\ que \\ \qquad forman\ la\ red\ empresarial \\ N_{SP} > 0, es\ el\ número\ de\ productos\ o\ \ servicios \\ \qquad que\ ofrecen\ las\ organizaciones \end{cases} \tag{3.9}$$

$$G_C = \frac{N_{SP}}{N_{EM}} \tag{3.10}$$

Donde $G_C > 0$ es el grado de cooperación, a través del cual podremos obtener los grados de cooperación vertical y horizontal de la siguiente forma:

$$\begin{cases} Si\ G_C < 1, \rightarrow\ G_{CH}\ es\ alto \\ Si\ G_C = 1, \rightarrow\ G_{CH}\ es\ medio \\ Si\ G_C > 1, \rightarrow\ G_{CH}\ es\ bajo \end{cases} \quad (3.11)$$

$$\begin{cases} Si\ G_C > 1, \rightarrow\ G_{CV}\ es\ alto \\ Si\ G_C = 1, \rightarrow\ G_{CV}\ es\ medio \\ Si\ G_C < 1, \rightarrow\ G_{CV}\ es\ bajo \end{cases} \quad (3.12)$$

Como se puede observar en las ecuaciones descritas en (3.11) y en (3.12), los grados de cooperación horizontal y vertical son inversos. Dentro de una red empresarial, a medida que existen más servicios ofrecidos por cada uno de sus integrantes a esa red empresarial, mayor será su especialización vertical. Por el contrario, si solamente existe un producto o servicio al que se dedican un grupo de empresas, lo que aumenta es el grado de especialización horizontal. Al igual que en la especialización de tareas los valores umbrales se definen a partir de la observación de diferentes compañías y teniendo en cuenta las conclusiones obtenidas en los trabajos de (Rushing, W. 1967; Samuel, Y. y Mannheim, B. 1970; Reimann, B. 1974).

3.2.2. Tipos de Especialización en Sistemas Multiagente

Una completa clasificación sobre los tipos de especialización en sistemas artificiales cooperativos, y por lo tanto en sistemas multiagente, se da en (Nitschke, *et al.*, 2007), donde se apunta que el estudio de la especialización en sistemas artificiales colaborativos ha sido abordado desde diferentes perspectivas en función del enfoque que los autores hayan dado a sus investigaciones. Ejemplo de estos trabajos son (Nolfi *et al.*, 2003; Campos *et al.*, 2001; Haynes y Sen, 1996; Bongard, 2000; Stone y Veloso, 2002; Bryant y Miikkulainen, 2003; Whiteson *et al.*, 2003; Blumenthal y Parker, 2004), entre otros.

La especialización de estos sistemas, en cuanto a su comportamiento colectivo, puede verse desde dos perspectivas: como una propiedad emergente del sistema o como una programación explícita de los componentes del mismo (Nitschke, *et al.*, 2007):

- **Especialización no emergente:** Es la que está explícitamente predefinida como parte del diseño del sistema y del comportamiento global del mismo. El enfoque practicado es estático, aunque pueden ser utilizados algoritmos de aprendizaje para especificar qué tipo de comportamiento es el adecuado para resolver una tarea determinada (Funes *et al.*, 2003; Arkin y Balch, 1999; Balch, 2002a; Balch, 2002b).

- **Especialización emergente:** Es la que emerge de la interacción de los componentes del sistema en respuesta a tareas dinámicas, que requieren diferentes tipos de especialización, para que sean finalizadas de forma óptima. Este enfoque es el más utilizado en entornos en los cuales no se conoce el grado especialización requerido en un principio (Stanley *et al.,* 2005; Waibel *et al.,* 2006; Gautrais *et al.,* 2002; Potter *et al.,* 2001; Luke y Spector, 1996; Theraulaz *et al.,* 1998; Murciano y Millan, 1997; Murciano *et al.,* 1997).

En el diseño de sistemas especializados emergentes se tiene en cuenta la homogeneidad o heterogeneidad de los componentes del sistema (Nitschke, *et al.,* 2007). Así pues, en el **enfoque homogéneo** de los componentes del sistema, se utiliza un comportamiento para todos los agentes del grupo, que en muchos de los casos se refiere a la definición de una serie de parámetros comunes a todos los agentes (Quinn *et al.,* 2003). Por el contrario, el **enfoque heterogéneo** de los componentes del sistema, define comportamientos diferentes para cada uno de los agentes que lo componen, que según (Campos *et al.,* 2001) consiste en la definición de parámetros diferentes para cada uno de ellos.

En (Nitschke, *et al.,* 2007), se propone una clasificación de la especialización que alude a la morfología de los agentes y a su comportamiento. En cuanto a la **especialización morfológica** se aplica cuando la situación de los agentes de un grupo se estructura de forma específica, para conseguir mejoras en la consecución de sus objetivos (Martinoli *et al.,* 2002; Zhang *et al.,* 2003; Watson *et al.,* 1999; Watson *et al.,* 2002). Por otra parte, la **especialización en función del comportamiento** de los agentes, se utiliza en entornos donde ciertos comportamientos de los agentes frente a tareas dinámicas resultan beneficiosos para el sistema (Li *et al.,* 2002; Bonabeau *et al.,* 1997; Balch, 2002a; Balch, 2002b; Nolfi y Parisi, 1997; Nolfi y Floreano, 2000).

Diversos autores en la literatura están de acuerdo que la división de los objetivos perseguidos por un grupo en subtareas especializadas beneficia la consecución de dichos objetivos (Arkin, 1998; Arkin y Balch, 1999; Balch, 2002a; Balch, 2002b). Sin embargo, la extrapolación de esta especialización, vista a nivel de grupos de agentes que colaboran entre ellos, de forma atómica, en la consecución de objetivos más amplios no está, todavía, bien definida.

Así pues, se pretende llevar a cabo una adaptación del concepto de *red empresarial* a los sistemas multiagente, de forma que sistemas multiagente basados en organizaciones y completamente autónomos, tengan una especialización, y la capacidad de interactuar entre ellos en la consecución de objetivos globales más amplios sin perder su independencia.

3.3. Especialización Aplicada a las Comunidades Inteligentes

Teniendo en cuenta la definición de Comunidades Inteligentes vista en el capítulo 2 de esta memoria, en esta sección se pretende especificar el tipo de especialización, que aplicada sobre la comunidad, mejore la capacidad de coordinación y productividad de la misma, y entre las mismas.

Las Comunidades Inteligentes se componen de un número reducido de agentes los cuales adoptan un rol determinado, sin posibilidad realizar otro tipo de tareas. El enfoque de especialización practicado dentro de las Comunidades Inteligentes se corresponde con el de una especialización no emergente (Funes *et al.,* 2003; Arkin y Balch, 1999; Balch, 2002a; Balch, 2002b), lo que implica que los agentes que componen la comunidad, se especifican explícitamente en el diseño de la misma. Para ello se dota a los agentes del comportamiento adecuado para que realicen su función. Además, en las comunidades de agentes se utiliza un enfoque de especialización que depende del comportamiento de los mismos, ya que en función del de ese comportamiento, se llevará a cabo una mejora en los objetivos globales del sistema (Li *et al.,* 2002; Bonabeau *et al.,* 1997; Balch, 2002a; Balch, 2002b; Nolfi y Parisi, 1997; Nolfi y Floreano, 2000).

Por otra parte, haciendo referencia a la especialización en las organizaciones humanas, las Comunidades Inteligentes practican internamente la especialización horizontal (Mintzberg, 1989), teniendo en cuenta que un agente asume solamente un rol determinado y por lo tanto no ejecutará otro tipo de tareas que no sean las que ese rol le confiera. Además también se focaliza la especialización vertical (Mintzberg, 1989), en el contexto de que existirá un agente que exclusivamente abordará tareas de control de los demás agentes integrantes de la comunidad.

Se ha comentado, en la sección anterior, la posibilidad de colaboración entre organizaciones o sistemas multiagente, asimilando la misma a la que se practica en las redes empresariales. En la comunidad, como forma de organización, se define una especialización de tareas a nivel organizacional, es decir, los objetivos perseguidos por la Comunidad Inteligente han de encontrarse en un mismo contexto. Este hecho permite la asociación de varias Comunidades Inteligentes independientes y con objetivos globales diferentes, las cuales tienen la capacidad de llevar a cabo los mismos de forma independiente, pero en su asociación podrán conseguir objetivos que de forma individual no lograrían.

Así pues, se propone el ejemplo de dos Comunidades Inteligentes independientes y que persiguen objetivos globales diferentes, haciendo referencia al concepto de especialización organizacional. En una de ellas sus agentes cooperan para obtener predicciones de series temporales. En la otra comunidad, sus agentes realizan tareas de cálculo sobre los costes de producción de un determinado producto, en función de la cantidad de materias primas. La asociación de estas dos

Comunidades Inteligentes, como se muestra en la **Figura 3.2.**, tendría la capacidad de determinar, mediante las predicciones de la primera comunidad, el coste óptimo aproximado de producción en función de la cantidad requerida del producto.

La posibilidad de simular una red empresarial entre Comunidades Inteligentes dota de una alta escalabilidad a las mismas, debido a que pueden asociarse tantas comunidades independientes como sean necesarias para resolver un problema global complejo. En estos términos, a la unión de Comunidades Inteligentes como tal, puede asociársele una especialización emergente (Stanley *et al.*, 2005; Waibel *et al.*, 2006; Gautrais *et al.*, 2002; Potter *et al.*, 2001; Luke y Spector, 1996; Theraulaz *et al.*, 1998; Murciano y Millan, 1997; Murciano *et al.*, 1997), ya que es la interacción con el entorno la que hace que sea una u otra comunidad la que tenga la especialización necesaria para llevar a cabo la tarea requerida de forma óptima.

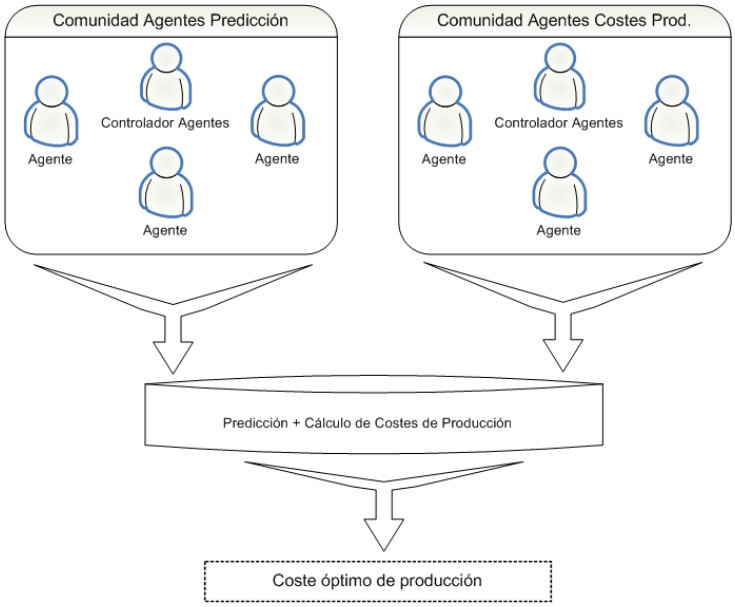


Figura 3.2 Simulación de red empresarial entre comunidades de agentes

Dentro de este enfoque de especialización emergente, asociado a una red de comunidades de agentes, los miembros de esta agrupación están especializados en tareas diferentes. Si tratamos a las comunidades de forma atómica, y se tiene en cuenta esta especialización, podemos afirmar que siguen un enfoque heterogéneo (Campos *et al.*, 2001), ya que cada Comunidad Inteligente, y por tanto cada miembro de la asociación, lleva a cabo tareas especializadas y diferenciadas.

Si hacemos referencia a las redes empresariales de organizaciones humanas, y tenemos en cuenta los tipos de las mismas, podemos asemejar la agrupación de Comunidades Inteligentes con un tipo de red empresarial vertical (Becerra, 2008), ya que cada integrante de esta asociación lleva a cabo tareas, especializadas y

diferenciadas, de forma individual y su unión conlleva la consecución de objetivos de forma conjunta, inalcanzables individualmente.

Tabla 3.1 Características de especialización en las Comunidades Inteligentes

	Especialización en Agentes	**Asimilación de especialización humana**
Agentes miembros de la comunidad	• No emergente • En función del Comportamiento	• Horizontal • Vertical
Agrupación de Comunidades	• Emergente • Heterogéneo	• Vertical

En la **Tabla 3.1** se definen las características de especialización de las Comunidades Inteligentes, y de la agrupación de las mismas. Ahora bien, en cuanto a la asimilación de Comunidades Inteligentes a la especialización humana, han de existir unos valores umbrales para que una agrupación de agentes que formen una Comunidad Inteligente se considere especializada. Para ello tomamos los indicadores definidos en la sección 3.2.1, y los aplicaremos a una Comunidad Inteligente, de forma que se puedan establecer umbrales en cuanto al tipo de especialización que practican.

Tabla 3.2. Grado de especialización empresarial de una Comunidad Inteligente

	Indicadores	**Valor**
Especialización Empresarial	N_{TE}	N_{TE}
	N_E	$0 < N_E \leq 4$
	E_E	$0 < E_E \leq 0.75$

La **Tabla 3.2** muestra los valores que tienen que tomar los diferentes indicadores para que una Comunidad Inteligente pueda considerarse especializada, en cuanto al grado de especialización general de la comunidad.

Tabla 3.3. Grado de especialización de tareas en una Comunidad Inteligente

	Indicadores	**Valor**
Especialización de tareas	N_T	$0 < N_T \leq 10$
	P	1
	E_T	$E_T \geq 0.1$

Teniendo en cuenta los valores establecidos en la **Tabla 3.3** como umbrales para que una Comunidad Inteligente se considere especializada a nivel de tareas, hay que analizar los dos tipos de especialización a nivel de tareas existentes. Viendo la relación existente en las ecuaciones propuestas en (3.3), se observa que el grado de el nivel de especialización horizontal el alto. Ahora bien, como es de esperar por la estructura de la Comunidad Inteligente, la especialización vertical dependerá del nivel jerárquico en el que nos encontremos. Así pues, el controlador de la comunidad no tendrá especialización vertical ya que es el que controla y supervisa a los demás miembros. Sin embargo, los miembros del segundo nivel jerárquico

poseerán cierta especialización vertical ya que están sometidos a tareas de control por parte del controlador de la comunidad.

3.4. CONCLUSIONES

Se ha definido la especialización como una característica que dota a un individuo, perteneciente a un grupo, de una particularidad dentro de ese grupo.

A lo largo de este capítulo se realizado un estudio del concepto de especialización y su aplicación en las organizaciones humanas, así como, en sistemas artificiales cooperativos. A través de este estudio se ha podido determinar el beneficio que tiene la aplicación de especialización dentro de unidades organizacionales, mediante la cual se posibilita una división de tareas especializadas, que coordinadas, llevan a la consecución de objetivos inalcanzables por los miembros de estas unidades organizacionales de forma individual, beneficiando a la organización (Arkin, 1998; Arkin y Balch, 1999; Balch, 2002a; Balch, 2002b). Además se han propuesto una serie de indicadores, y sus fórmulas para calcularlos, que orienten a medir el nivel de especialización dentro de las organizaciones. Estos indicadores se han definido con la intención de aplicarlos a los sistemas multiagente para medir el grado de especialización, y por lo tanto a las Comunidades Inteligentes.

El análisis de los tipos de especialización en sistemas cooperativos artificiales (Nitschke, *et al.*, 2007) y en las organizaciones humanas (Mintzberg, 1989), nos ha permitido conocer sus características y las situaciones en las que su aplicación lleva consigo una mejora en el proceso de realización de tareas. Dentro de los tipos de especialización se ha introducido el concepto de red empresarial y sus principales características (Becerra, 2008), con la finalidad de llevar una adaptación del mismo a los sistemas multiagente organizacionales en forma de Comunidad Inteligente. Esta adaptación del concepto de red empresarial a las Comunidades Inteligentes es clave en el objetivo principal de este trabajo de investigación, ya que a través del mismo, se posibilita la cooperación coordinada de Comunidades Inteligentes, con capacidades especializadas, dotándolas de una mayor capacidad de resolución de problemas y con aplicabilidades tan diversas como se requiera.

Finalmente se ha llevado a cabo una definición de especialización dentro de las Comunidades Inteligentes, introducidas en el capítulo 2 de esta memoria. Se han establecido los valores umbrales, en función de los indicadores dados en la sección 3.2.1, para que una Comunidad Inteligente se defina como especializada. La adaptación del concepto de red empresarial (Becerra, 2008) a las Comunidades Inteligentes dota a las mismas de una gran flexibilidad en cuanto a la posibilidad de crecer de forma escalada y llevar a cabo tareas globales con capacidad de resolver problemas que individualmente no podrían.

CAPÍTULO 4. AGENTES Y SISTEMAS MULTIAGENTE

4. AGENTES Y SISTEMAS MULTIAGENTE

En este capítulo se realiza un análisis del estado del arte en lo referente a los agentes y sistemas multiagente (MAS). Este tipo de entidades presenta unas características y propiedades que tienen la capacidad de modelar problemas computacionales con autonomía y proactividad, por lo que se convierten indispensables en nuestra investigación. Estas capacidades que proveen los agentes pueden ser modeladas emulando comportamientos humanos, lo que implica su idoneidad para cooperar en unidades organizacionales y dotarlos de la especialización necesaria para optimizar su rendimiento.

El grupo de investigación BISITE (Biomedicina, Sistemas Inteligentes y Tecnología Educativa) de la Universidad de Salamanca, cuenta con una amplia experiencia en el área de la Inteligencia Artificial Distribuida y el desarrollo de sistemas multiagente. Por tal motivo, esta tesis doctoral sigue una línea continuista en la investigación y aplicación de este tipo de sistemas, incidiendo en una nueva arquitectura para el desarrollo de los mismos.

El presente capítulo se estructura de la siguiente forma. En la sección 4.2. se realiza un estudio de los diferentes tipos de agentes existentes. En la sección 4.3. se comentan las arquitecturas de sistemas multiagentes y sus principales características. A continuación, en la sección 4.4. se introducen las diferentes metodologías existentes para el desarrollo de sistemas multiagente. En la sección 4.5. se introducen los diferentes tipos de plataformas de agentes y sus principales características. Posteriormente en la sección 4.6. se hace referencia a la distribución de los servicios ofrecidos por en los sistemas multiagente. Finalmente, en la sección 4.7. se presentan las conclusiones obtenidas en este capítulo.

4.1. INTRODUCCIÓN AL CONCEPTO DE AGENTE

Existen muchas definiciones sobre el concepto de agente (Wooldrige y Jennins, 1995; Shoham, 1997; Maes, 1994; Brustolini, 1991; FIPA, 1996; OMG, 1998) A continuación se presentan varias definiciones dadas por autores dedicados a este campo.

Una de las definiciones de agente más aceptadas en la comunidad científica es la de Wooldrige (Wooldrige y Jennins, 1995), que define a un agente como un sistema que tiene las siguientes características:

- **Autonomía:** los agentes deben poder llevar a cabo sus tareas de forma autónoma, sin intervenciones externas, y han de tener el control de sus acciones y de su estado interno.

- **Habilidad social:** los agentes han de poder interactuar con otros agentes y con humanos, mediante el lenguaje apropiado, para llevar a cabo sus tareas o ayudar a otros.

- **Reactividad:** los agentes están situados en un entorno, tienen la capacidad de recibir estímulos de dicho entorno, y son capaces de reaccionar a los mismos.

- **Pro-actividad:** los agentes no actúan solamente como respuesta a estímulos procedentes de su entorno, sino que, además tienen la capacidad de tomar la iniciativa para ejecutar acciones cuando lo crean conveniente.

Además de estas características, se proponen otras que sería deseable que poseyeran:

- **Adaptabilidad:** esta característica representa la capacidad del agente para modificar su comportamiento en función de los estímulos recibidos de su entorno.

- **Movilidad:** capacidad de cambiar de ubicación física cuando sea necesario.

- **Benevolencia:** los agentes no comunican información falsa que pueda corromper los fines perseguidos.

- **Racionalidad:** la persecución de los objetivos del agente hace que actúe de acuerdo al mismo.

Otra definición de agente es dada en (Shoham, 1997) en la que se identifica a un agente como una entidad software que realiza actividades continuas en el tiempo y de forma autónoma, en un entorno determinado cohabitando con otros agentes y procesos. En (Maes, 1994), se define a los agentes como sistemas computacionales que habitan algún entorno dinámico y complejo, perciben y actúan de forma autónoma en ese entorno y, haciendo esto, llevan a cabo un conjunto de metas o tareas para los que han sido diseñados. Por su parte J.C. Brustolini define a los agentes como sistemas capaces de actuar autónomamente y con determinación en el mundo real (Brustolini, 1991). En un informe técnico de la Object Management Group, se define a un agente como un programa de ordenador que actúa

autónomamente en nombre de una persona u organización (OMG, 1998). Finalmente, se toma una definición propuesta por la FIPA (Foundation for Intelligent Physical Agents), la cual determina que un agente es una entidad que reside en entornos donde interpreta datos que reflejan eventos y ejecuta comandos que producen efectos en ese entorno (FIPA, 1996).

Los Sistemas Multiagente (Multi-Agent Systems) se corresponden con potentes modelos computacionales distribuidos, basados en entidades software que cooperan entre sí (Amigoni y Fugini 2007)**.** Los MAS son usualmente caracterizados en términos de comportamientos internos e interacciones externas entre agentes (Wooldridge, 2002). Las propiedades principales para caracterizar el comportamiento interno de los agentes son: el tipo de razonamiento y el funcionamiento del mismo en cuanto a cómo actúa, por ejemplo, reactivos, basados en modelos, basados en metas; su grado de adaptabilidad; su percepción y caracterización del entorno en el que están situados, incluyendo su infraestructura computacional y sus relaciones con otros agentes; y el grado de autonomía en las acciones que realizan (Posland, 2007).

La composición de un sistema multiagente viene determinada por su arquitectura. Esta arquitectura determina qué mecanismos que utiliza un agente para interactuar con su entorno. Las arquitecturas utilizadas para construir este tipo de sistemas, especifican cómo se descomponen los agentes en un conjunto de módulos que interactúan entre sí para lograr la funcionalidad requerida. Uno de los aspectos básicos que diferencia una arquitectura de otra es el método de descomposición del trabajo en tareas particulares (Corchado, 2005). Existen varios tipos de arquitecturas las cuales se verán en la sección 4.2. de este capítulo.

El rápido y constante desarrollo tecnológico lleva consigo la adaptación y construcción de sistemas cada vez más complejos, para lo cual es necesario que exista la capacidad de reutilización de componentes y funcionalidades de los mismos, así como, compatibilidad entre ellos. Los sistemas multiagente tienen la capacidad de lograr estos requisitos, aunque para ello se ha de dotar a estos sistemas de un enfoque distribuido en cuanto a sus funcionalidades se refiere, de forma que se facilite la reutilización e integración de los mismos.

Debido a que el análisis y desarrollo de este tipo de sistemas es complejo, es necesario estudiar las herramientas y metodologías de Ingeniería del Software orientadas a agentes (AOSE - Agent Oriented Software Engineering) (Wooldrige y Jennins, 2000), las cuales facilitan y mejoran dicho desarrollo. Así como también es necesario realizar un acercamiento a las diferentes plataformas de desarrollo y ejecución de sistemas multiagente con el fin de conocer sus características y adoptar las posibilidades que ofrecen.

La computación orientada a servicios (SOC) utiliza servicios que soportan el desarrollo de aplicaciones distribuidas de forma rápida, operable, escalable y masiva (Papazoglou, 2007). La aplicación del paradigma de la computación orientada a servicios en los sistemas multiagente confiere una serie de ventajas a estos sistemas

y cumple con la característica de ejecución de servicios de forma distribuida propuesta para las Comunidades Inteligentes, definidas en el capítulo 2 de esta memoria.

4.2. Tipos de Agentes y Sistemas Multiagente

Existen diversos tipos de agentes que se pueden clasificar en función de sus características individuales y su comportamiento (Franklin *et al.*, 1996; Bradshaw, 1997; Brenner, *et al.*, 1998; Maes, 1994).

En función de sus características individuales, los agentes se clasifican en **reactivos** cuando realizan tareas sencillas reaccionando ante eventos externos sin capacidad de realizar razonamientos a través de mecanismos de representación del conocimiento; y agentes **cognitivos o deliberativos**, los cuales poseen mecanismos de razonamiento y representación explícita del conocimiento. Así pues, los agentes cognitivos perciben estímulos del entorno, razonan una planificación, y actúan de acuerdo a dicha planificación.

En función del modo de interacción de los agentes, existen agentes de **interfaz** los cuales están especializados en interactuar con el entorno y agentes **autónomos** que interaccionan con los recursos y otros agentes decidiendo las modificaciones necesarias en función de los cambios que se puedan producir en el entorno.

Según la organización de los agentes, se pueden diferenciar agentes **individuales,** los cuales no tienen capacidad de cooperación con otros agentes, y agentes **cooperativos** que tienen la capacidad de colaborar con otros agentes en la consecución de tareas conjuntas.

Las arquitecturas utilizadas para construir sistemas multiagente, especifican cómo se combinan los agentes interactuando entre ellos para lograr así el fin requerido. A continuación se presentan las principales arquitecturas en función del tipo de razonamiento que utilizan.

4.2.1. Arquitecturas Reactivas

Las arquitecturas reactivas se caracterizan por carecer de razonamiento simbólico complejo y de conocimiento de su entorno (Maes, 1989; 1990) Esto implica que la comunicación entre sus agentes es básica ya que los mecanismos empleados para ello lo son, así pues, en los sistemas multiagente basados en este tipo de arquitectura, los agentes reciben estímulos del entorno o de otros agentes, y reaccionan ante ellos. La mayor aplicación de este tipo de arquitecturas se ha centrado en el desarrollo de controladores en robótica. Los robots pueden considerarse como agentes reales (no software) que actúan en un entorno cambiante. Precisamente la necesidad de actuar en un entorno impredecible y altamente

cambiante dificulta la adopción de otro tipo de arquitectura ya que las necesidades de replanificación y de continua adaptación del plan a la realidad hacen muy difícil que una arquitectura de este tipo responda con suficiente agilidad (Corchado, 2005). Un ejemplo de la aplicación de este tipo de arquitecturas puede verse en (Carlési, *et al.*, 2011), donde la idea es la mejora y simplificación de la planificación de las misiones de robots submarinos a través de este tipo de arquitectura.

4.2.2. Arquitecturas Deliberativas

Este tipo de arquitecturas utilizan modelos de representación simbólica del conocimiento. Los agentes de esta arquitectura parten de un estado inicial y tienen la capacidad de generar y concluir una serie de planes para alcanzar sus metas (Maes, 1989). En este tipo de sistemas se acepta la idea de que es necesario dotar a los agentes de un sistema de planificación que se encargue de determinar qué pasos han de seguir para alcanzar sus objetivos. Esto implica que, un agente deliberativo (o con una arquitectura deliberativa) es aquel que contiene un modelo simbólico del mundo, explícitamente representado, en donde las decisiones se toman utilizando mecanismos de razonamiento lógico basados en la correspondencia de patrones y la manipulación simbólica, con el propósito de alcanzar los objetivos del agente (Corchado, 2005). En el apartado 4.2.4., se presenta la arquitectura deliberativa BDI *(Belief, Desire, Intention)* como caso especial de este tipo de arquitecturas por ser la más extendida (Rao y Georgeff, 1995; Zato *et al.*, 2011; Rodríguez *et al.*, 2011).

4.2.3. Arquitecturas Híbridas

Este tipo de arquitecturas combinan aspectos de las arquitecturas reactivas y deliberativas, supliendo las carencias que tienen las mismas por separado. Así pues, los sistemas basados en este tipo de arquitectura contendrán una serie de agentes reactivos, los cuales reaccionarán a los eventos que se produzcan en el entorno, y otros deliberativos que tengan la capacidad de generar planes, para lo cual utilicen un modelo simbólico de razonamiento (Corchado, 2005). Se trata de aprovechar las características deseables tanto de las arquitecturas deliberativas como de las no-deliberativas (o reactivas). Los agentes con arquitecturas deliberativas tienen escasa capacidad de respuesta al entorno si los cambios exigen respuestas rápidas. En el caso de los agentes con arquitecturas reactivas, no parece demasiado claro que sean capaces de comportarse con estrategias a largo plazo de manera eficaz. Las arquitecturas híbridas aprovechan de algún modo las ventajas de las arquitecturas deliberativas y reactivas para el diseño de agentes, utilizándolas de forma complementaria (Kefalas y Stamatopoulou, 2011; Birkin y Wu, 2012; Ah-Hwee *et al.*, 2012).

4.2.4. Arquitectura Deliberativa BDI

En la arquitectura deliberativa BDI *(Belief, Desire, Intention)*, los agentes que la implementan tienen asociados una serie de estados mentales, como son, *Creencias*, *Deseos* e *Intenciones* (Bratman, 1988; Rao y Georgeff, 1995). En (Rao y Georgeff, 1995) se definen estas creencias, deseos e intenciones. Las **Creencias** representan el

conocimiento del entorno del sistema y de los estados internos del agente. La información representada es imperfecta, pero fundamental para tener en cuenta eventos pasados y mejorar la interacción con el entorno del sistema pudiendo ser modificadas en función de dicha interacción. Por su parte los **Deseos** representan los objetivos del sistema. Estos objetivos pueden ser un valor o expresión de un estado final deseado. Cada agente puede perseguir varios objetivos lo que implica tener en cuenta información sobre los mismos y la prioridad de cada uno de ellos. Finalmente las **Intenciones**, representan los mecanismos de planificación del sistema. A través de las intenciones ciertas acciones pueden ser modificadas para alcanzar los objetivos, almacenándose para su posterior utilización en futuras situaciones.

Además, para alcanzar los objetivos deseados se definen una serie de planes (Corchado, *et al.*, 2008; Oijen y Dignum, 2011), que se corresponden con las secuencias de acciones necesarias para lograr estos objetivos. Estos planes vinculados a la consecución de un objetivo constituyen las intenciones del agente (Corchado, 2005; Oijen y Dignum, 2011). Una representación de la arquitectura de un agente BDI se propone en la **Figura 4.1.**

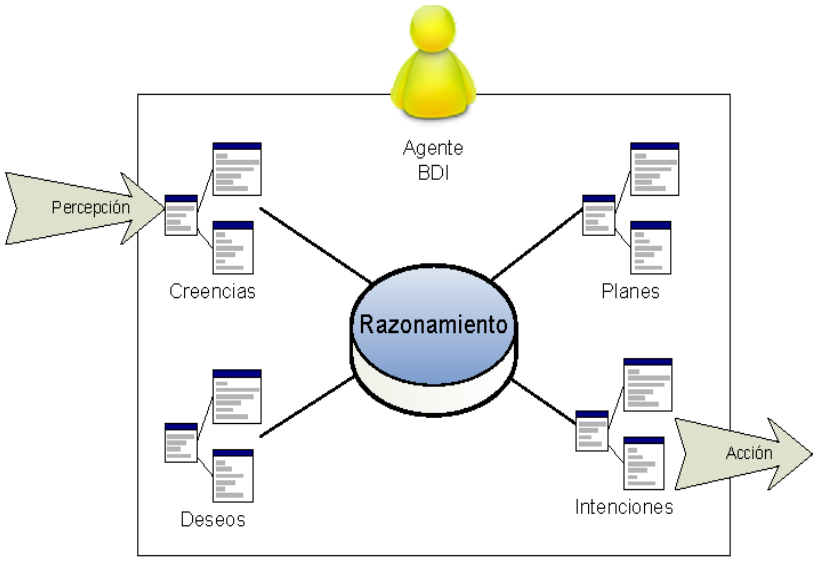


Figura 4.1 Representación gráfica de la arquitectura de un agente BDI

Los agentes deliberativos BDI son modelados a través de una estructura abstracta, basada en la lógica de situaciones (o mundos posibles), denominada árbol temporal con múltiples futuros y un solo pasado (Rao y Georgeff, 1991). Cada nodo de este árbol se corresponde con una situación, y las ramas con las opciones del agente en un momento dado. Para cada situación se definen una serie de nuevas situaciones que el agente puede alcanzar desde el punto de vista de las creencias (situaciones que se consideran posibles), de los deseos (situaciones que se desean alcanzar) y de las intenciones (situaciones que se intentan alcanzar) (Rao, *et al.*, 1995). Para este modelo es necesario que existan ciertas relaciones entre las

creencias, los deseos y las intenciones del agente (Corchado, 2005; Koster *et al.*, 2012; Oijen *et al.*, 2011):

- **Compatibilidad entre creencias y objetivos**. Si el agente adopta el deseo de alcanzar un objetivo, debe creer que dicho objetivo es cierto.
- **Compatibilidad entre objetivos e intenciones.** Anteriormente a que el agente adopte una intención debe haberla adoptado como deseo.
- **Las intenciones conducen a acciones.** Si una de las intenciones es una acción simple, el agente la ejecuta.
- **Relación entre creencias e intenciones.** El agente conoce sus propias intenciones.
- **Relación entre creencias y objetivos.** El agente conoce sus objetivos o deseos.
- **No hay retrasos infinitos.** Cuando un agente adopta una intención, sigue con ella hasta algún momento determinado del futuro.

En función de cómo las intenciones actuales del agente guían o influyen en sus decisiones sobre futuras intenciones se identifican varios tipos de agentes (Corchado, 2005):

- **Ciego.** El agente mantiene sus intenciones hasta que sabe que las ha conseguido, y rechazará las creencias o deseos que sean contradictorios.
- **Firme.** El agente mantiene sus intenciones mientras crea que tiene opciones de alcanzarlas.
- **Imparcial.** El agente mantiene sus intenciones mientras éstas se corresponden con sus deseos.

La implementación de las creencias, deseos e intenciones se realiza almacenándolos en listas de forma separada, y se trabaja con secuencias de eventos en una estructura de datos lineal en forma de cola *(FIFO: First In First Out)*. En la arquitectura BDI se ejecutan una serie de pasos de forma cíclica y finita, durante los cuales se permite interacción con el entorno. Al comienzo del ciclo, se realiza una lectura de cola de eventos y se devuelve una lista de opciones, mediante la cual seleccionan aquellas que se deben ser adoptadas, y se añaden a la cola de intenciones. Posteriormente, se ejecutan todas las intenciones que contengan una acción simple, y se realiza una comprobación de la existencia de nuevos eventos en el entorno los cuales han de ser incorporados a la cola de eventos. Finalmente, el agente BDI, modifica las estructuras de deseo e intención, eliminando los ya satisfechos y los que son imposibles de alcanzar (Rao, *et al.*, 1995).

Los agentes deliberativos BDI se han utilizado de forma exitosa en diversos desarrollos llevados a cabo por el grupo de investigación BISITE (Corchado, *et al.*, 2008, 2010; Bajo, *et al.*, 2009, 2010; Tapia, *et al.*, 2006; 2008; 2009; Rodríguez *et al.*, 2011; Zato *et al.*, 2012). Este tipo de agentes proporciona soluciones en entornos dinámicos, lo cual los hace idóneos en el desarrollo de sistemas en los que se necesita una escalabilidad en función del objetivo global del sistema. Es por ello que

los agentes BDI resultan de vital importancia en el desarrollo de la arquitectura SCODA, la cual se describe detalladamente en el capítulo 5 de esta memoria.

4.3. METODOLOGÍAS PARA EL DESARROLLO DE SISTEMAS MULTIAGENTE

El desarrollo de sistemas multiagente lleva consigo un proceso de ingeniería del software, con una orientación a agentes, que es necesario debido a la complejidad que este tipo de sistemas lleva consigo. Existen diferentes metodologías para el desarrollo de estos sistemas las cuales se muestran en la **Tabla 4.1**.

Tabla 4.1 Diferentes metodologías para el desarrollo de sistemas multiagente

Metodología	Referencias	Descripción
GAIA	(Wooldridge y Jennings, 2000) (Silva *et al.*, 2010) (Castro *et al.*, 2012)	Es una de las metodologías más estudiadas, la cual permite el análisis y diseño de sistemas multiagente en forma de organizaciones de agentes. Estos agentes adoptan roles específicos de la organización y cooperan entre ellos para lograr el objetivo global
TROPOS	(Giunchiglia *et al.*, 2002)	Mediante esta metodología se cubre el proceso completo de desarrollo en cinco fases: toma de requisitos iniciales, requisitos posteriores, diseño de la arquitectura, diseño detallado, e implementación. Para ello utiliza un modelado i*
PROMETHEUS	**(**Padgham y Winikoff, 2004)	En esta metodología se considera el ciclo completo de ingeniería del software para el diseño de sistemas multiagente: especificación de requisitos, análisis, diseño, implementación, pruebas y mantenimiento.
MASE	(Wood, 2000) (García-Ojeda *et al.*, 2007)	Multiagent Systems Engineering, guía el desarrollo de sistemas multiagente desde la especificación inicial del sistema hasta su implementación. El proceso de desarrollo se divide en las fases de análisis y diseño.
MAS-KOMMONCADS	(Iglesias *et al.*, 1996)	Está basada en la metodología CommonKADS para sistemas expertos y amplía nuevos modelos relativos a las fases de análisis y diseño de sistemas multiagente.
MASSIVE	(Lind, 2001)	Multi-Agent Systems Iterative View Engineering, está formada por una serie de vistas del sistema multiagente, las cuales son refinadas iterativamente.

AGR	(Ferber *et al.*, 2003)	Agent Group Role, está basada en los conceptos de agente, grupo y rol. Se considera un agente a una entidad activa y comunicativa que es capaz de asumir ciertos roles dentro de un grupo. El concepto de grupo pasa a ser un conjunto de agentes con características comunes, y finalmente un rol es la representación abstracta de una funcionalidad de un agente perteneciente a un grupo.
ROADMAP	(Juan *et al.*, 2002)	Esta metodología extiende la, ya mencionada, metodología GAIA (Wooldridge *et al.*, 2000) para el desarrollo de sistemas multiagentes abiertos y complejos, heredando de la misma su composición en las fases de análisis y diseño.
MESSAGE	(Caire *et al.*, 2002)	Methodology of Engineering Systems of Software Agents, utiliza la notación UML y el proceso de modelado propuesto por Rational (Jacobson *et al.*, 1999).
INGENIAS	(Gómez, 2002)	Se basa en la metodología MESSAGE (Caire *et al.*, 2002) y utiliza lenguajes para el meta-modelado para la construcción de sus modelos.
ANEMOMA	(Giret, 2005)	En esta metodología se agrega una perspectiva estructural al concepto de agente, extendiendo así, la metodología INGENIAS (Gómez, 2002) con el concepto de agente abstracto. Es empleada en el desarrollo de sistemas multiagente, basados en organizaciones holónicas para fabricación.
AML	(Cervenka *et al.*, 2005; Cervenka y Trencansky, 2007)	El lenguaje de modelado de agentes extiende UML a los agentes, mediante un lenguaje visual y semi-formal. A través de AML se tiene la capacidad de definir sistemas multiagente que serán utilizados en un entorno determinado y estarán formados por entidades que podrán ser agentes, recursos, entornos y unidades organizativas. Este lenguaje de modelado de agentes permite realizar de forma efectiva el análisis y el diseño del sistema.
GaiaEXOA	(Zambonelli *et al.*, 2003)	Esta metodología, Gaia Extended with Organizational Abstractions, es una extensión de la metodología GAIA (Wooldridge *et al.*, 2000), a través de la cual se pretende el desarrollo de sistemas abiertos mediante conceptos organizacionales aplicados a los sistemas multiagente.

La aplicación de la Ingeniería del Software orientada a Agentes sobre los sistemas multiagente facilita el desarrollo de los mismos y su posterior mantenimiento. En este trabajo se opta por utilizar la metodología Gaia (Wooldridge y Jennings, 2000) como metodología de desarrollo y AML (Cervenka y Trencansky, 2007) como lenguaje de modelado de sistemas multiagente. A través de Gaia y AML se obtiene un modelado relativamente cercano a la implementación de una forma eficiente y rápida. La metodología GAIA y el lenguaje de modelado de agentes, AML, se presenta en detalle en el Anexo A de esta memoria.

4.4. Plataformas para el Desarrollo y Ejecución de Sistemas Multiagente

En esta sección se describen y evalúan diversas plataformas de desarrollo y ejecución de agentes mediante las cuales se facilita la implementación y posterior ejecución de los desarrollos basados en sistemas multiagente. A través del estudio de las principales características de las mismas se pretende comprobar el grado de adecuación de la arquitectura propuesta en esta tesis sobre alguna de estas plataformas y llevar a cabo su implementación. La selección de estas plataformas ha sido de alguna forma deliberada, atendiendo a aspectos como la capacidad de ejecución distribuida de servicios, soporte para agentes deliberativos BDI (Bratman, 1988; Rao y Georgeff, 1995), posibilidad de desarrollo de sistemas multiagente organizativos y la facilidad en el proceso de desarrollo de aplicaciones basadas en sistemas multiagente.

ADK (Xu y Shatz, 2003): *Agent Development Kit* es una plataforma comercial para el desarrollo de sistemas multiagente. Está construida en Java y XML a través de una arquitectura modular y tiene capacidad para ofrecer computación distribuida, es flexible, segura y tolerante a fallos. Los componentes más importantes son los ACF *(Agent Foundation Classes)* y el ARE *(Agent Runtime Environment)*. A través de ACF se permite el desarrollo de componentes mediante una API la cual permite a los programadores dotar a dichos componentes de funcionalidad e interactividad. Por su parte el ARE es el encargado de soportar el entorno de ejecución de los agentes. Además ADK tiene la capacidad de soportar Servicios Web, lo cual dota de una mayor flexibilidad en cuanto a sus comunicaciones con el exterior.

Bee-gent (Kawamura *et al.*, 2000a; 2000b): *Bonding and Encapsulation Enhancement Agent*, es una plataforma comunicativa para el desarrollo de sistemas multiagente desarrollada por Toshiba Corporation, con capacidad para soportar agentes cooperativos y facilitar la comunicación entre agentes y servicios distribuidos. Está basada en el modelo de comunicación de CORBA *(Common Object Request Broker Architecture)* (Vinowski, 1997), a través de la cual los agentes invocan los servicios necesarios en un servidor de forma distribuida. Bee-

gent permite a los desarrolladores llevar a cabo sistemas multiagentes distribuidos, abiertos y flexibles que maximicen su eficiencia. Los agentes utilizados en esta plataforma son agentes BDI (Bratman, 1988; Rao y Georgeff, 1995) de forma que posee un motor de razonamiento que dota de mayor robustez los desarrollos.

Cougaar (Snyder y MacKenzie, 2004; BBN Technologies, 2004): *Cognitive Agent Architecture*, es una arquitectura de código libre que provee herramientas para la implementación de sistemas multiagente distribuidos con un gran número de entidades y recursos. Esta arquitectura no sigue las especificaciones FIPA, sin embargo implementa una estructura similar (BBN Technologies, 2004). Cougaar es un producto de DARPA *(Defense Advanced Research Projects Agency).* La plataforma que la soporta y la propia arquitectura están construidas en Java, e implantan un modelo cognitivo de sistema multiagente en cuanto a la percepción e interacción de los agentes en el sistema. Cougaar está enfocada al desarrollo de aplicaciones basadas en sistemas multiagente con un campo de aplicación dentro de la logística militar, mejorando los procesos de planificación e interacción que ésta sigue.

JADE (Bellifemine *et al.*, 1999)**:** *Java Agent DEvelopment Framework*, es una plataforma que simplifica el desarrollo de sistemas multiagente, proporcionando un *middle-ware* que cumple con las especificaciones FIPA (FIPA, 2005) y una serie de herramientas para facilitar su desarrollo y depuración, a través de un entorno gráfico. La filosofía que persigue JADE en cuanto al desarrollo de sistemas multiagente se basa en *contenedores*, de forma que cada agente se corresponde con un contenedor. Existe un contenedor principal que encapsula a todos los demás contenedores, es decir a todos los demás agentes. Cumpliendo las especificaciones FIPA (FIPA, 2005), JADE utiliza el protocolo de comunicación ACL *(Agent Communication Language)* en la comunicación entre los agentes.

RETSINA (Sycara, 2003): Es un sistema multiagente abierto que soporta agrupaciones de agentes heterogéneos. Su implementación sigue la premisa de que los agentes del sistema forman grupos que interaccionan entre ellos, implementándose así una red de tareas jerárquicas cuya función es la de crear secuencias de acciones (planes). RETSINA, no emplea un control centralizado dentro del sistema multiagente sobre los servicios ofrecidos por los mismos, sino que implementa una infraestructura de servicios distribuida que facilita las interacciones entre agentes. Las comunicaciones llevadas a cabo entre los agentes son administradas por un comunicador *(Communicator)* que utiliza para ello el protocolo KML. RETSINA incluye un comunicador *(Communicator)* que administra la comunicación entre los agentes utilizando el protocolo KQML *(Knowledge Query and Manipulation Language).*

ZEUS (Collins y Ndumu, 1999): Es una herramienta para desarrollar sistemas multiagente organizacionales que se caracteriza por ofrecer un entorno integrado para el desarrollo rápido de estos sistemas de forma visual, a través del cual se capturan las especificaciones de los agentes, que posteriormente son utilizadas para

generar su código fuente en Java. ZEUS pretende un desarrollo rápido de aplicaciones basadas en sistemas multiagente.

JADEX (Pokahr *et al.*, 2003; 2007): Es en entorno de desarrollo de sistemas multiagente que sigue el modelo BDI (Bratman, 1988; Rao y Georgeff, 1995). Originalmente fue diseñado para JADE, sin embargo actualmente puede ejecutarse de forma autónoma o con otras plataformas, siendo la principal característica de JADEX su consideración como motor de razonamiento que trata de limitar al máximo sus dependencias con la plataforma que lo ejecuta (García-Montoro, 2007).

A la vista de las plataformas expuestas, y de las características que ofrece cada una de ellas la elección para implementar la arquitectura SCODA se ha decantado por JADEX. Debido a las características de SCODA es necesario que la arquitectura que la implemente haga uso de agentes BDI. Además se busca independencia de la plataforma que la ejecute y la posibilidad de ejecutar servicios externos a la misma. Es por ello que JADEX se ofrece como un motor de razonamiento que ejecuta agentes BDI y que permite la ejecución de servicios externos. Además, JADEX cumple con los requisitos establecidos por la FIPA (FIPA, 2005) y ofrece un diseño modular que dota de una gran sencillez al programar los agentes, definidos en XML, que la componen. JADEX, se encuentra en constante desarrollo desde el año 2003 dando un excelente soporte a los desarrolladores[4.1].

4.5. Distribución de Servicios en Sistemas Multiagente

La computación orientada a servicios (SOC) utiliza servicios que soportan el desarrollo de aplicaciones distribuidas de forma rápida, operable, escalable y masiva (Papazoglou, 2007). En este aspecto, los servicios son autónomos, independientes de la plataforma y débilmente acoplados a la aplicación que los invoca.

La clave para llevar a cabo este paradigma son las Arquitecturas Orientadas a Servicios (Papazoglou, 2007). SOA (Rosen *et al.*, 2008) es el camino para el desarrollo de sistemas compuestos por aplicaciones y servicios distribuidos (Papazoglou, 2007). SOA provee un conjunto de de guías, principios y técnicas, a través de las cuales los procesos de negocio, información y procesos empresariales, se pueden reorganizar y distribuir de forma efectiva elevando el nivel de competitividad interempresarial (Papazoglou, 2003). La definición de servicio es crucial en la efectividad de una arquitectura orientada a servicios, así pues, un servicio puede ser definido como una función claramente formulada, auto-contenida e independiente del contexto en el que se ejecute (Papazoglou, 2006). La implementación de SOA está fuertemente ligada a los Servicios Web, siendo estos una forma de desarrollar estas arquitecturas, pero no la única (Zimmerman, 2005).

4.1 http://jadex-agents.informatik.uni-hamburg.de

El desarrollo de sistemas multiagente basados en computación orientada a servicios, donde los servicios prestados por los agentes son ejecutados de forma distribuida y bajo demanda, dota a estos sistemas de mayores niveles de flexibilidad, escalabilidad y de una mejor distribución de carga computacional, que en sistemas centralizados (Shen y Norrie, 1998; Camarinha-Matos y Afsarmanesh, 2007; Voos, 2006).

Una de las características de las Comunidades Inteligentes, definidas en el capítulo 2 de esta memoria, es la capacidad de ejecutar los servicios ofrecidos por las mismas de forma distribuida, persiguiendo así, una mejora en la ejecución de los servicios, en cuanto a la carga computacional presentada por los agentes. Así pues, el diseño de la arquitectura SCODA, presentada en el capítulo 5 de esta memoria, hace uso del paradigma de la computación orientada a servicios.

Tabla 4.2 Ventajas e inconvenientes de la distribución de servicios en sistemas multiagente

Ventajas	Inconvenientes
• Menor carga computacional sobre los agentes • Posibilidad de ejecutar varios servicios en paralelo • Reutilización de funcionalidades • Independencia del lenguaje de programación • Gestión y control de errores independiente de los agentes	• Necesidad de comunicaciones robustas • Necesidad de conocer la ubicación del servicio

En la **Tabla 4.2** se muestran ventajas e inconvenientes que tiene la distribución de servicios en un sistema multiagente. Se puede observar que tienen un mayor peso las ventajas sobre los inconvenientes, ya que el rendimiento en el sistema multiagente se incrementa al descargar computacionalmente a sus miembros, además de proporcionar una gestión de errores independiente. En cuanta a la necesidad de comunicaciones robustas, es un factor externo al sistema ya que generalmente depende del medio físico en el que se actúe. La necesidad de conocer la ubicación del servicio se solventa a través directorios de publicación de servicios proporcionados por el propio sistema multiagente.

4.6. CONCLUSIONES

La revisión de la teoría de agentes y sistemas multiagente es un punto fundamental en el marco de este trabajo ya que el conocimiento de los mismos permite clarificar la estructura de la arquitectura propuesta en esta tesis.

La capacidad de razonamiento de los agentes deliberativos BDI (Belief, Desire, Intention), hacen que resulten idóneos para su integración en las Comunidades Inteligentes, propuestas en el capítulo 2 de este trabajo, debido a la necesidad de planificación en tiempo real que estas comunidades tienen. Como consecuencia, estos agentes dotan a estas comunidades de la capacidad de aprender de las

experiencias pasadas y reaccionar de manera diferente de acuerdo a las necesidades de los usuarios y las características del contexto en situaciones determinadas.

El desarrollo de sistemas multiagente requiere de un proceso de ingeniería mediante el cual se faciliten sus fases. Las diferentes metodologías de desarrollo de sistemas multiagente están enfocadas desde diferentes perspectivas en función de las necesidades del propio sistema, es por ello que el modelado que se seguirá en el desarrollo de SCODA se corresponde con Gaia y AML, los cuales permiten realizar el análisis y diseño de sistemas multiagente de una forma consistente tanto a nivel de agentes individuales como en forma organizativa. Además, a través de AML se permite modelar los servicios ofrecidos por los agentes, así como, el entorno con el que interactúan.

El conocimiento de las diferentes características de diversas plataformas de desarrollo y ejecución de agentes, permite comprobar el grado de adaptación de la arquitectura propuesta en esta tesis en base a dichas características. La posterior implementación sobre alguna de ellas, permitirá comprobar el funcionamiento de la misma y optimizar así su estructura arquitectónica.

La ejecución de los servicios de forma distribuida en un sistema multiagente permite descarga computacional de los agentes incidiendo así en el rendimiento del sistema. Esta ejecución distribuida de servicios es una característica de las Comunidades Inteligentes, definidas en el capítulo 2 de este trabajo, las cuales son el pilar fundamental de SCODA, la arquitectura propuesta en este trabajo, que se definirá con detalle en el capítulo 5 de esta memoria.

CAPÍTULO 5.
ARQUITECTURA SCODA

5. ARQUITECTURA SCODA

En este capítulo se presenta SCODA *(Distributed and Specialized Agent COmmunities)* como la arquitectura plateada en esta investigación, detallando su estructura y funcionalidades. Además se propone una herramienta para desarrollar y gestionar sistemas multiagente basados en SCODA de forma automática, lo cual facilitará los desarrollos de este tipo de sistemas los que implementen esta arquitectura.

5.1. INTRODUCCIÓN

En general las personas u organizaciones, están acostumbradas a generar soluciones para solventar los problemas que puedan producirse en un momento determinado. En muchos casos ya se han identificado mecanismos para resolver estos problemas con anterioridad (Jensen, 1992). Este hecho traducido a los sistemas computacionales lo encontramos en la tecnología Orientada a Objetos (Lieberherr, 1996; Elrad *et al.*, 2001; Rashid *et al.*, 2003; Kiczales *et al.*, 1997) 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 este trabajo, en el que se busca una arquitectura basada en Comunidades Inteligentes (CI), ya descritas en el capítulo 2 de esta memoria, que puedan ejecutarse de forma independiente, y tengan la capacidad de colaborar de forma atómica con otras Comunidades Inteligentes.

SCODA *(Distributed and Specialized Agent COmmunities)* (Román *et al. 2011*) es una nueva arquitectura que se centra en el desarrollo de sistemas multiagente. Para ello se basa en cinco principios: estandarización (Carlési, et al., 2011), especialización, facilidad de implementación, reutilización y computación distribuida. SCODA integra una o más Comunidades Inteligentes dotadas de especialización, que denominaremos **Comunidades Inteligentes Especializadas (CIE)** (Román *et al. 2011*). Estas CIE tienen la capacidad de funcionar como un sistema multiagente independiente y especializado, donde los servicios que ofrece son ejecutados de forma distribuida, de manera que persiga un objetivo global. Esta estructura permite a diferentes CIE colaborar 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; Burt, 1980; Walker *et al.*, 1997; Lazer, 2003; Borgatti y Foster, 2003; Tracey y Clark, 2003; Johansson y Quigley, 2004; 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. Es precisamente este enfoque, el que hace que SCODA se base

en el concepto de Comunidad Inteligente, descrito en el capítulo 2 de esta memoria, como unidad organizacional y con las características necesarias para llevar a cabo su desarrollo en estos términos.

SCODA es una arquitectura que puede ser implementada en cualquier plataforma para sistemas multiagente, que soporte agentes BDI (Bratman, 1988; Rao y Georgeff, 1995). En el capítulo 4 de esta memoria se introducen un conjunto de plataformas para el desarrollo y ejecución de aplicaciones basadas en sistemas multiagentes, de las cuales se ha seleccionado JADEX (Pokahr *et al.*, 2003; 2007) debido a que es considerado como un motor de razonamiento y puede ser ejecutado de forma independiente.

El presente capítulo se estructura de la siguiente forma. En la sección 5.2 se realiza una descripción detallada de la arquitectura incidiendo en los aspectos más relevantes de la misma. A continuación, en la sección 5.3 se describe la inteligencia que poseen los agentes que componen la arquitectura para llevar a cabo sus tareas de forma óptima. En la sección 5.4 se detalla el modelado de la arquitectura haciendo uso de herramientas de análisis y diseño orientadas a agentes (AOSE). La sección 5.5 describe la herramienta implementada, como contribución a este trabajo de investigación, para desarrollar e integrar Comunidades Inteligentes Especializadas en la arquitectura propuesta. Finalmente, en la sección 5.6. se presentan las conclusiones obtenidas de este capítulo.

5.2. Descripción de la Arquitectura

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. Esto implica, que en la fase de diseño se conoce el conjunto de los agentes que participan, y las interacciones que se producen entre ellos (Zambonelli et al. 2003). 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 (González-Palacios y Luck 2007). Es por ello que la proposición de la arquitectura SCODA surge como alternativa intermedia entre estos dos enfoques, ya que se basa en Comunidades Inteligentes Especializadas que funcionan como sistemas multiagente cerrados, y poseen la capacidad de cooperar de forma escalada con otras Comunidades Inteligentes Especializadas, de forma que permita la incorporación en tiempo de ejecución de las mismas.

La arquitectura SCODA se basa en cinco principios, ya mencionados en el capítulo introductorio de esta memoria, a través de los que se pretende una mayor eficiencia de los sistemas multiagentes desarrollados con la misma. Así pues, en la **Tabla 5.1.** se muestra un esquema de los cinco principios mencionados:

Tabla 5.1 Principios en los que se basa SCODA

Principio	Descripción
Estandarización	Las diferentes Comunidades Inteligentes Especializadas tienen una misma estructura que es independiente de la finalidad que se persiga en el sistema multiagente a implementar (Carlési, et al., 2011).
Especialización	Este principio se basa en la especialización humana y en las *"Redes Empresariales"* (Sonquist y Koenig, 1975; Galaskiewicz, 1979; Burt, 1980; Walker *et al.*, 1997; Lazer, 2003; Borgatti y Foster, 2003; Tracey y Clark, 2003; Johansson y Quigley, 2004; Viedma, 2004; Pöyhönen y Smedlund, 2004; Ibarra *et al.*, 2005; Eraydin y Armatli-Köroglu, 2005; Cabus y Vanhaverbeke, 2006) de forma que exista cooperación entre Comunidades Inteligentes Especializadas.
Facilidad de Implementación	Los sistemas Multiagente, 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, la reutilización de estas Comunidades Inteligentes Especializadas ha de ser viable en cualquier desarrollo basado en SCODA.
Computación Distribuida	Los servicios requeridos no los prestan directamente los agentes integrantes de las Comunidades Inteligentes Especializadas, sino que se ejecutan de forma distribuida para que la carga computacional asociada a los agentes disminuya y la estructura de la arquitectura no tenga variaciones.

SCODA está estructurada de forma modular, de manera que los agentes que la componen administran y coordinan la propia arquitectura y sus funcionalidades. Tal y como se muestra en la **Figura 5.1.,** SCODA se define en seis módulos básicos: Aplicaciones Externas, Protocolo de Comunicaciones, Control, Plataforma de Agentes, Comunidades Inteligentes Especializadas y Servicios de las Comunidades.

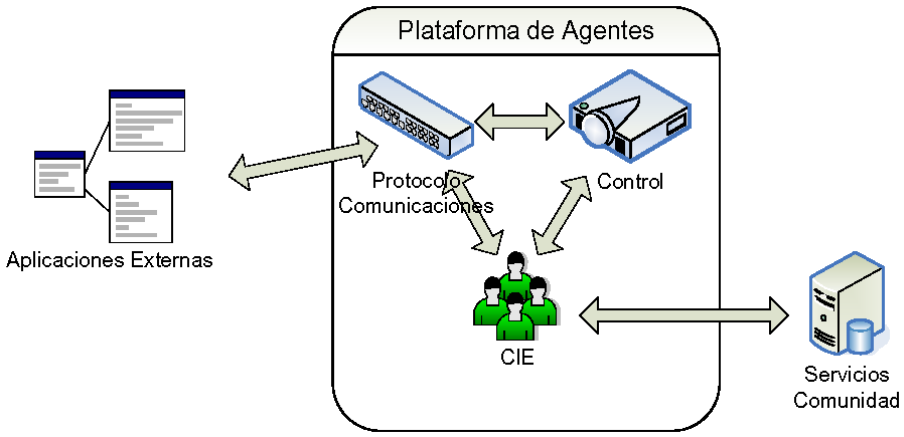


Figura 5.1 Esquema estructural de SCODA

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. Estos seis bloques en los que se estructura SCODA están definidos en las siguientes secciones de este capítulo

Los agentes que componen SCODA siguen el modelo deliberativo BDI (Bratman, 1988; Rao y Georgeff, 1995) y los servicios que ofrecen las Comunidades Inteligentes Especializadas están 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.

Un ejemplo de funcionamiento sobre SCODA, se muestra en la **Figura 5.2.,** en el cual, *"una aplicación externa desea realizar una operación aritmética, es este caso una multiplicación. La aplicación externa realiza una petición a SCODA, esta petición es recibida por el sistema que selecciona la Comunidad Inteligente Especializada necesaria para atender la petición. La CIE seleccionada hace uso de sus servicios, invocando remotamente el servicio de comunidad adecuado con los parámetros de la petición de forma deliberada. Esta petición se procesa y resuelve devolviendo la respuesta a la CIE que lo ha invocado. Esta comprueba su validez a través de mecanismos de razonamiento, y a su vez, lo envía a la aplicación externa que la solicita".*

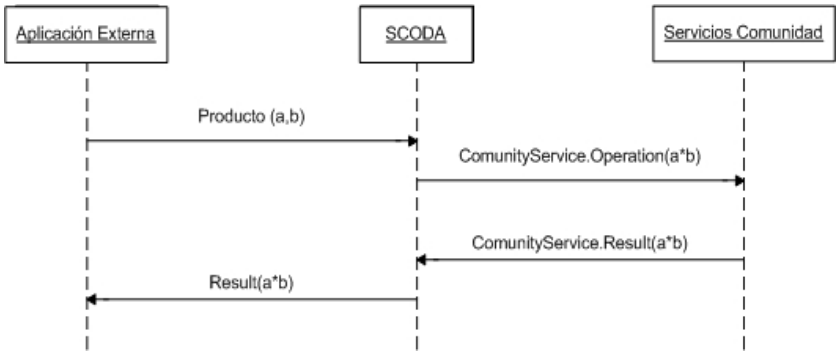


Figura 5.2 Esquema de funcionamiento de una petición sobre SCODA

A través de la **Figura 5.2.** se muestra un pequeño ejemplo del funcionamiento sobre SCODA, en el que una petición de una operación aritmética, lo cual implica una pequeña carga computacional y una implementación del servicio sencilla. Se ha de resaltar que si se requiriese la ejecución de un complejo algoritmo el cual tuviera un peso computacional elevado, la CIE responsable del servicio, así como la arquitectura, no se ven cargadas computacionalmente debido a que estos servicios son ejecutados de forma remota y distribuida reorganizando los procesos, lo que hace de SCODA una arquitectura de agentes ligera en este aspecto.

5.2.1. 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.

5.2.2. Protocolo de Comunicaciones

SCODA implementa un protocolo de comunicaciones basado en REST *(REpresentational State Transfer)* (Fielding, 2000). REST 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 en la **Tabla 5.2.**

Tabla 5.2 Formato de las peticiones HTTPRequest en SCODA

Métodos Utilizados	Descripción
GET/POST	*Request = Simple-Request	Full-Request* *Simple-Request = "GET"|"POST" 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 (Pokahr *et al.*, 2003; 2007) se utiliza *FIPA Agent Communication 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:

Tabla 5.3 Formato de las respuestas HTTPResponse en SCODA

Descripción
*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, JSON (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 5.3.** se muestra un ejemplo de solicitud de un servicio a una SCODA.

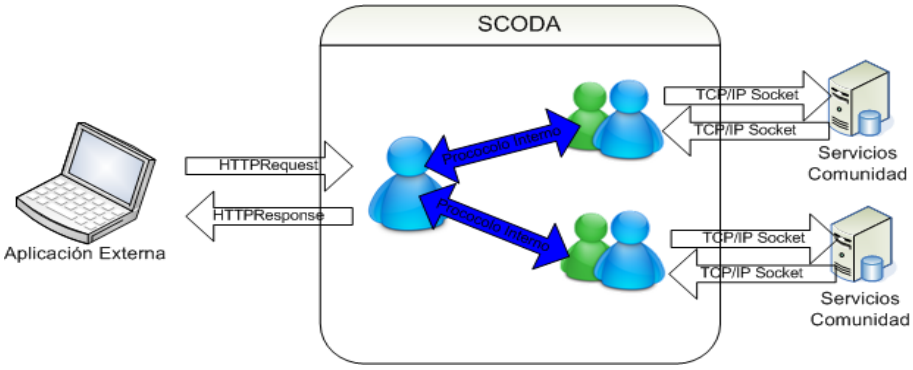


Figura 5.3 Ejemplo de solicitud de servicio a SCODA

En el **Código 5.1** se muestran las fases de la comunicación entre una aplicación externa y SCODA. La aplicación externa solicita a SCODA un servicio a través de un *HTTPRequest*. SCODA recibe la petición y selecciona a la Comunidad Inteligente Especializada que tenga capacidad para atender la petición. Esta comunidad busca el servicio más conveniente para la petición y selecciona los parámetros óptimos para su ejecución. En el caso de que no existiese ningún servicio adecuado a la petición se generaría un error, lo mismo que si la ejecución del servicio resultase errónea. La Comunidad Inteligente Especializada notifica la respuesta y es SCODA quien a través de una *HTTPResponse* envía la respuesta de ejecución a la aplicación externa. Toda la comunicación interna de SCODA se realiza mediante mensajes ACL.

Fases de comunicación externa en SCODA
1. Aplicación Externa → HTTP(Solicitar-Servicio (servicio))
2. SCODA → HTTP-Request(Recepción-Servicio (servicio))
3. SCODA → ACL (Seleccionar-Comunidad(servicio))
4. Mientras (Haya-Servicios)
5. COMUNIDAD → Buscar-Servicio(servicio)
6. Si (Servicio-Disponible)
7. SCODA → TCP/IP (Comunidad(Invocar-Servicio(servicio, parámetros)))
8. SCODA → TCP/IP (Comunidad(Enviar-Respuesta))
9. Fin Si
10. Fin Mientras
11. Si (Servicio-Ejecutado-Éxito)
12. COMUNIDAD → ACL (Enviar-Respuestas(servicio))
13. SCODA → HTTP-Response (Enviar-Respuesta(servicio))
14. Si no
15. COMUNIDAD → ACL (Enviar-Respuesta(error))
16. SCODA → HTTP-Response (Enviar-Respuesta(error))
17. Fin Si

Código 5.1. Fases de comunicación externa con SCODA

5.2.3. 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, para 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.

Figura 5.4 Agentes de SCODA que implementan el módulo de control

La **Figura 5.4** muestra la implementación que hace cada uno de los agentes del módulo de control. Así pues, el *CommunityControllerAgent* ejerce control sobre el *QualityAgent*, y sobre su equipo formado por el *PlannerAgent* y el *ExecutorAgent.* Además controla que los servicios de su comunidad funcionen correctamente. De todo el flujo de información que controla este agente se informa al *QualityAgent*. El *QualityAgent* ejerce control sobre cada *CommunityControllerAgent* existente en SCODA y sobre el *CommunicatorAgent.* Además lleva un control de las comunicaciones internas de la plataforma y gestiona los errores que se puedan producir. Finalmente, el *CommunicatorAgent* ejerce control sobre las comunicaciones externas e informa al *QualityAgent* de los movimientos que controla. Todas las comunicaciones internas de la plataforma se realizan a través de mensajes ACL, siguiendo las especificaciones de la FIPA (FIPA, 2005).

Ejemplo de actuación del módulo de control	
1.	*Aplicación Externa → Solicitar Servicio(servicio)*
2.	*Communicator → RecibirSolicitud(servicio)*
3.	*Communicator → Seleccionar Comunidad(servicio)*
4.	*Communicator → InformarQuality(Inform)*
5.	***Quality → Agente con Funcionamiento defectuoso***
6.	*Communicator → Recibir Confirmación(Quality, ACK)*
7.	*Communicator → Reiniciar Agente(Quality)*

Código 5.2. Ejemplo de actuación del módulo de control

El **Código 5.2** muestra un ejemplo de actuación de varios agentes que implementan el módulo de control. Una aplicación externa solicita un servicio a SCODA. El *CommunicatorAgent* recibe la solicitud, selecciona la comunidad capaz de dar respuesta a la petición e informa al *QualityAgent*. El *CommunicatorAgent* espera confirmación de la recepción de la información por parte del *QualityAgent* (*mensaje "ACK"*), sin embargo no llegará porque el *QualityAgent* tiene problemas de ejecución. Al no recibir confirmación el *CommunicatorAgent* reinicia al *QualityAgent* y vuelve a iniciar el proceso de información para poder continuar con la solicitud del servicio.

5.2.4. 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. Compuestas por un agente controlador *(CommunityController)*, 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 su 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.

Ejemplo Comunidades Inteligentes Especializadas individuales	
1.	*Aplicación Externa → Solicitar Servicio(traducción)*
2.	*Communicator → RecibirSolicitud(traducción)*
3.	*Communicator → Seleccionar Comunidad(traducción)*
4.	*Comunidad Traducción → Ejecutar Servicio (traducción);Enviar Respuestas(traducción)*
5.	*Aplicación Externa → Solicitar Servicio(diccionario)*
6.	*Communicator → RecibirSolicitud(diccionario)*
7.	*Communicator → Seleccionar Comunidad(diccionario)*
8.	*Comunidad Diccionario → Ejecutar Servicio(diccionario); Enviar Respuestas(diccionario)*

Código 5.3. Ejemplo de ejecución de Comunidades Inteligentes Especializadas individuales

La **Figura 5.5** muestra la posibilidad de que las Comunidades Inteligentes Especializadas se ejecuten de forma individual, sin que exista una colaboración entre ellas. Se puede observar que existen N Comunidades Inteligentes Especializadas independientes, y a cada una de ellas se le asocian M servicios. Las peticiones que entren en SCODA, serán atendidas por la Comunidad Inteligente Especializada que esté capacitada para dar una respuesta. Así pues, como ejemplo se propone el **Código 5.3** donde se ilustra de forma sencilla lo que se está exponiendo. Se cuenta con dos Comunidades Inteligentes Especializadas, una que provee servicios de traducción y la otra, servicios de diccionario. Aplicaciones externas solicitan a SCODA servicios de traducción y servicios de diccionario sin ninguna relación, a los que las Comunidades Inteligentes Especializadas correspondientes responden de forma independiente.

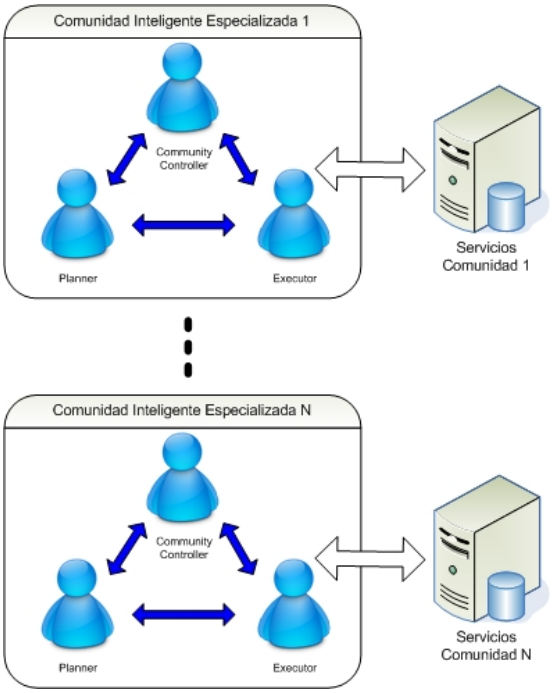


Figura 5.5 Comunidades Inteligentes Especializadas sin colaboración

La **Figura 5.6** muestra el detalle de Comunidades Inteligentes Especializadas que colaboran de forma escalada para obtener resultados que de forma individual no lograrían. Como ejemplo proponemos las Comunidades Inteligentes Especializadas del caso anterior, una que ejecuta servicios de traducción y otra, servicios de diccionario. En el caso que nos ocupa una aplicación externa solicitará un servicio de diccionario y de traducción de forma conjunta, lo que implica que ha de existir una colaboración entre ambas Comunidades Inteligentes Especializadas para lograr atender la petición realizada por la aplicación externa.

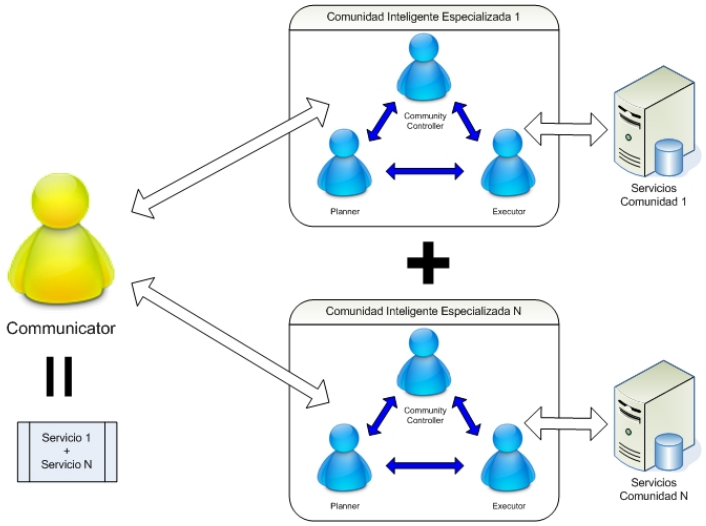


Figura 5.6 Comunidades Inteligentes Especializadas usadas de forma escalada

El **Código 5.4** muestra de forma detallada la colaboración entre las Comunidades Inteligentes Especializadas que ofrecen servicios de traducción y de diccionario. Esta colaboración se realiza de forma escalada y planificada por el CommunicatorAgent, que tiene la capacidad de seleccionar las Comunidades Inteligentes Especializadas necesarias para lograr el objetivo global.

Ejemplo Comunidades Inteligentes Especializadas colaborando
1. *Aplicación Externa → Solicitar Servicio(diccionario+traducción)*
2. *Communicator → RecibirSolicitud(diccionario+traducción)*
3. *Communicator → Seleccionar Comunidad(diccionario)*
4. *Comunidad Diccionario → Ejecutar Servicio (diccionario);Enviar Respuestas(diccionario)*
5. *Communicator → RecibirRespuesta(diccionario)*
6. *Communicator → Seleccionar Comunidad(traducción, Respuesta Diccionario)*
7. *Comunidad Traducción → Ejecutar Servicio(traducción, Respuesta Diccionario); Enviar Respuestas(traducción, Respuesta Diccionario)*
8. *Communicator → Enviar Respuesta (diccionario+traducción)*

Código 5.4. Ejemplo de ejecución de Comunidades Inteligentes Especializadas colaborando

5.2.4.1. GRADO DE ESPECIALIZACIÓN DE LAS COMUNIDADES INTELIGENTES ESPECIALIZADAS EN SCODA

Se han definido en la sección 3.2.1 una serie de indicadores para medir la especialización dentro de las organizaciones. Además, en la sección 3.3, se han establecido los valores umbrales para considerar una Comunidad Inteligente como Especializada. En primer lugar se estimarán los indicadores para calcular el grado de especialización empresarial a nivel de la Comunidad Inteligente y posteriormente se calculará el grado de especialización de tareas.

Tabla 5.4 Grado de especialización empresarial de una Comunidad Inteligente Especializada en SCODA

	Indicadores	Valor
Especialización Empresarial	N_{TE}	1
	N_E	3
	E_E	0.33

La **Tabla 5.4** muestra los valores que toman los indicadores para una Comunidad Inteligente Especializada en SCODA. En función de la composición y del número de tareas generales que realiza la comunidad obtenemos que N_E (número de agentes) es 3, cumpliendo así la restricción de ser menor o igual a 4. La única tarea general que realiza una Comunidad Inteligente Especializada en SCODA es la de ejecutar un servicio, con lo que N_{TE} es 1. Con estos valores se obtiene que, E_E tiene un valor de 0.33, y se encuentra en el umbral establecido para que una Comunidad Inteligente sea especializada a nivel organizacional.

Tabla 5.5 Grado de especialización de tareas en una Comunidad Inteligente Especializada en SCODA

	Indicadores	Valor
Community Controller	N_T	7
	P	1
	E_T	0.14
Planner	N_T	3
	P	1
	E_T	0.33
Executor	N_T	2
	P	1
	E_T	0.5

La **Tabla 5.5** muestra los valores de los indicadores para medir la especialización de tareas para una Comunidad Inteligente Especializada en SCODA. Cada uno de los agentes representa un puesto de trabajo, y cada uno de ellos llevará a cabo un número de tareas repetitivas. Tomando los indicadores umbrales definidos en la sección 3.3, se puede observar que el *PlannerAgent* y el *ExecutorAgent* alcanzan niveles de especialización muy altos. Esto es lógico debido a que son agentes que se limitan a realizar la ejecución de un servicio distribuido, mientras que el *CommunityControllerAgent* alcanza un nivel alto de especialización, ya que lleva a cabo tareas de control y de información. No obstante, todos los agentes a nivel individual se encuentran en el umbral de alta especialización. Este hecho implica que el grado de especialización de las Comunidades Inteligentes Especializadas en SCODA es alto, tanto a nivel de tareas, como a nivel de organización.

5.2.5. 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. A estos sistemas se accede 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 (Sunwook *et al.,* 2009; Balaji *et al.,* 2006; Douglas y Pai 2006). Los servicios 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 (Sonquist y Koenig, 1975; Galaskiewicz, 1979; Burt, 1980; Walker *et al.,* 1997; Lazer, 2003; Borgatti y Foster, 2003; Tracey y Clark, 2003; Johansson y Quigley, 2004; Viedma, 2004; Pöyhönen y Smedlund, 2004; Ibarra *et al.,* 2005; Eraydin y Armatli-Köroglu, 2005; Cabus y Vanhaverbeke, 2006).

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, modificados, añadidos y eliminados 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).

Tabla 5.6 Estructura del directorio de servicios de las Comunidades Inteligentes Especializadas en SCODA

Comunidad	Invocación	Predicado	Host	Puerto	Extra	Descripción
[Predicción]	[demanda]	[type=1; urldatos=data1…]	[community1]	[8998]	[empty]	[predicción de la demanda de un producto]
[EOQ]	[eoq]	[type=3; urldatos=data1…]	[community3]	[8787]	[empty]	[gestión de inventarios]

La **Tabla 5.6** muestra la estructura del directorio de servicios de las Comunidades Inteligentes Especializadas. Este directorio incluye datos sobre el nombre de la comunidad, como se ha de invocar al servicio requerido, el predicado con los parámetros necesarios, el host donde se encuentra el servicio y su puerto de acceso y la descripción del servicio. Además se muestra un campo con la descripción de cada uno de los servicios.

5.2.6. Plataforma de Agentes

Este módulo es el núcleo de SCODA. Integra los agentes 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, de forma que cada agente tiene asociadas una serie de funcionalidades que se determinan en función del rol que adopten. Los agentes BDI conforman la arquitectura que dota a los mismos de la capacidad de actuar como administradores del sistema, de forma que sustentan el control de las comunicaciones, y administran el comportamiento de los demás agentes, incluidos ellos mismos

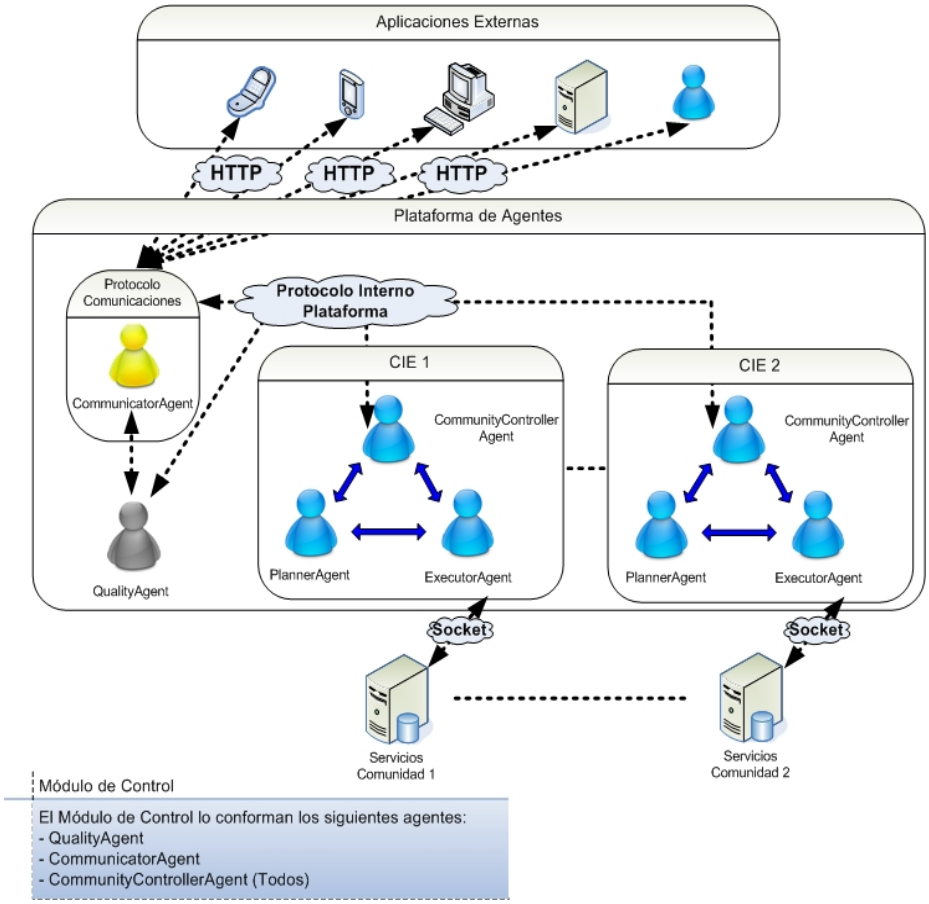


Figura 5.7 Arquitectura SCODA

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 5.7.**, el acceso a los servicios ofrecidos por las Comunidades Inteligentes Especializadas pude realizar 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. La estructura interna de estos agentes, así como la comunicación que proveen se detalla en el Anexo B de esta memoria donde se modelan los agentes y el comportamiento individual y colectivo de cada uno de ellos:

Tabla 5.7 Descripción del Communicator Agent

Communicator Agent	
Descripción	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 a 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.
Rol	Comunicador

Tabla 5.8 Descripción del Quality Agent

Quality Agent	
Descripción	Se responsabiliza del control del funcionamiento de los demás agentes, así como de las funciones que realizan. Se encuentra en constante comunicación con el *CommunicatorAgent* y con cada uno de los *CommunityControllers*. 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. El funcionamiento y mantenimiento de las estadísticas de funcionamiento serán explicadas, en la sección 5.3. de este capítulo.
Rol	Calidad

Tabla 5.9 Descripción de los Community Controller Agents

Community Controller Agent	
Descripción	Este agente controla las Comunidades Inteligentes Especializadas. Una vez haya 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.
Rol	Controlador

Tabla 5.10 Descripción del Planner Agent

Planner Agent	
Descripción	Es uno de los componentes del "Equipo de Trabajo". Es el responsable directo de la selección y ejecución del servicio adecuado que responda a una petición externa. Este agente se crea en tiempo de ejecución por el CommunityController cuando se ha de invocar un Servicio de la Comunidad y recibe la petición del propio CommunityController 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.
Rol	Planificador

Tabla 5.11 Descripción del Executor Agent

Executor Agent	
Descripción	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.
Rol	Ejecutor

5.3. Inteligencia Computacional en SCODA

SCODA utiliza un conjunto de mecanismos de razonamiento en sus agentes, a través de los cuales se dota a la arquitectura de la capacidad de utilizar los recursos y servicios de forma optimizada. El control realizado por SCODA, implementado sobre diversos agentes, dota a la arquitectura de una tolerancia a fallos sobre el funcionamiento de los mismos. Así pues, el *QualityAgent* tiene la capacidad de

reiniciar al *CommunicatorAgent* y a cada uno de los *CommunityControllers* integrantes de la arquitectura, cuando se detecta un funcionamiento erróneo de los mismos o simplemente cuando no están en funcionamiento. Por su parte tanto el *CommunicatorAgent* como cada uno de los *CommunityControllers* pueden instanciar un nuevo *QualityAgent* cuando éste está inactivo. El control sobre estos agentes se realiza de forma periódica a través de mensajes que confirmen el funcionamiento de los mismos como se muestra en la **Figura 5.8.**

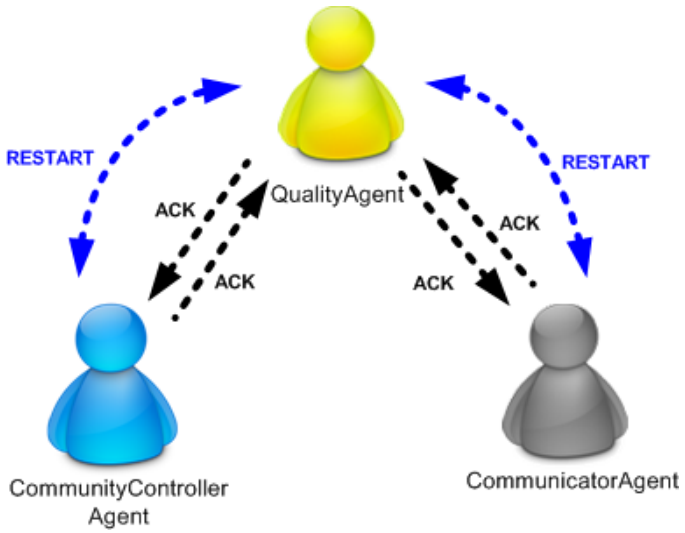


Figura 5.8 Control de Agentes en SCODA

Los mensajes ACK (acknowledge) son mensajes informativos que requieren una confirmación por parte del receptor. Así pues, a través de este tipo de mensajes se determina que los agentes se ejecutan con normalidad dentro del sistema. Los mensajes RESTART informan a su receptor de que será reiniciado, es decir, en primer lugar se eliminará la instancia del agente para posteriormente iniciar una nueva instancia.

Figura 5.9 Ejemplo de funcionamiento del control de agentes en SCODA

La **Figura 5.9** muestra un ejemplo del control ejercido por el CommunicatorAgent sobre el *QualityAgent*. Los mensajes 1 y 2 muestran el funcionamiento correcto del *QualityAgent* y en los mensajes 3 y 4 se puede observar como el *QualityAgent* no contesta al ACK enviado por el *CommunicatorAgent* y este lo reinicia ante un posible problema. A través del *QualityAgent* se lleva un control estadístico sobre las peticiones que se realizan a las Comunidades Inteligentes Especializadas y a cada uno de los servicios que éstas proveen. Estos datos están actualizados constantemente y son los siguientes:

Tabla 5.12 Gestión de funcionamiento de SCODA

Descripción de datos de funcionamiento en SCODA	
Tiempo Máximo de Ejecución de Servicio	Este dato indica el tiempo máximo de ejecución del servicio.
Peticiones Totales	Corresponden al número de veces que se ha invocado el servicio correspondiente.
Fallos Totales	Es el total de fallos que se han producido al invocar el servicio correspondiente.
Aciertos Totales	Es el total de aciertos que se han producido al invocar el servicio correspondiente.
No Parameters	Este dato se corresponde al número de fallos producidos por una invocación errónea del servicio debido a que los parámetros necesarios no son correctos.
Fail Team	Contabiliza los fallos producidos debido a un incidente en la creación o ejecución de cada "Equipo de Trabajo" que se instancia para ejecutar un servicio.
Servicio Inactivo	Corresponde al número de fallos producidos al invocar un servicio que se encuentra inactivo.
No Existe Servicio Adecuado	Indica el número de fallos producidos al solicitar un servicio a la plataforma que no existe o no se da soporte para el mismo.
Problema Ejecución de Servicio	Este dato proporciona los fallos debidos a incidentes en la ejecución del servicio, ya sea por problemas internos de del mismo o por un retardo importante en su ejecución.

Estos datos, permiten al *QualityAgent* realizar funciones de supervisión sobre las Comunidades Inteligentes Especializadas y servicios de SCODA e implementar una alta tolerancia a fallos sobre el sistema. Así pues, en el caso de un fallo en la creación de un "Equipo de Trabajo", este agente ordena al *CommunityControllerAgent* correspondiente volver a ejecutar la petición e instanciar un nuevo "Equipo de Trabajo". Los servicios también son controlados por *QualityAgent*, de forma que si un determinado servicio se encuentra inactivo, este agente tiene la capacidad de reiniciarlo.

Estos datos son muy útiles a la hora de determinar el número de réplicas de servicios necesarios para un funcionamiento óptimo del sistema. Mediante el tiempo máximo de ejecución del servicio y el número de peticiones de un servicio

determinado, el sistema determina dinámicamente el número de servicios óptimo en un periodo de tiempo determinado siguiendo a través de la siguiente fórmula *(5.1)*:

$$N_R = \frac{N_S \times T_{MAX}}{P \times Th} \quad (5.1)$$

Siendo:
N_R, el número de réplicas necesarias.
N_S, el número de servicios solicitados en un periodo de tiempo **P.**
T_{MAX}, el tiempo máximo de ejecución de servicio.
P, un periodo de tiempo determinado.
Th, el número de hilos simultáneos sobre un servicio.

En función del número de réplicas existentes de un determinado servicio, el *PlannerAgent*, tiene la capacidad de seleccionar el servicio que menor carga computacional soporta, optimizando el funcionamiento del sistema. Esta selección se realiza a través de un directorio que cada Comunidad Inteligente Especializada mantiene, en el que se especifican las réplicas de los servicios y su dirección.

5.4. Modelado de SCODA

En esta sección se realiza el modelado de la arquitectura SCODA, como sistema multiagente, teniendo en cuenta su composición basada en las Comunidades Inteligentes Especializadas, para los cual se utilizarán herramientas de análisis y diseño orientadas a agentes (AOSE). En primer lugar se hace uso de la metodología Gaia para obtener un modelo sencillo del sistema y de sus componentes principales. El resultado obtenido a través de Gaia es detallado y complementado mediante el lenguaje de Modelado de Agentes (AML), ambos detallados en el Apéndice A de esta tesis, obteniendo así un modelo de la arquitectura cercano a su implementación. A continuación, se describen las distintas fases del análisis y diseño para la obtención del modelo de SCODA.

5.4.1. Análisis GAIA + AML

Mediante la fase de análisis Gaia se definen el modelo de roles y el modelo de interacción. El modelo de roles se lleva a cabo, a partir de las características y requerimientos del sistema, y el modelo de interacción, a través de las distintas interacción existentes en el sistema y los protocolos considerados en el modelo de roles. En el análisis Gaia también son especificados los protocolos y actividades que deben llevar a cabo cada uno de los roles, así como los permisos que tienen en el sistema (Anexos B.1-5).

En la **Tabla 5.13** se especifican de forma general los roles existentes en SCODA y que implementan sus agentes. También se provee una descripción de

cada uno de ellos, que en líneas generales marcan el funcionamiento de los agentes que los implementen.

Tabla 5.13 Roles desempeñados en SCODA

Descripción de los roles existentes en SCODA	
Rol Comunicaciones	A través de este rol se gestionan las comunicaciones entrantes y salientes al sistema. Lleva a cabo tareas de selección de servicios a través de las Comunidades Inteligentes Especializadas y controla el funcionamiento del rol calidad. Este rol se mantiene constantemente en alerta comprobando la entrada de solicitudes al sistema.
Rol Calidad	Gestiona información estadística del sistema y comprueba el funcionamiento de los demás roles y servicios. Se encuentra en constante comunicación con el rol comunicaciones y con el rol controlador de comunidad, junto a los cuales gestiona la supervisión del sistema.
Rol Controlador	Gestiona las Comunidades Inteligentes Especializadas y dentro de ellas a los equipos de trabajo, compuestos por los roles planificador y ejecutor, creando los mismos y destruyéndolos cuando han finalizado su trabajo. Ejecuta labores de control del sistema junto a los roles calidad y comunicaciones.
Rol Planificador	Gestiona la planificación de los servicios de la Comunidad Inteligente Especializada a la que pertenece seleccionando el servicio y los parámetros que se adecuen de forma óptima a la petición recibida del rol controlador. Envía los parámetros de ejecución de servicio al rol ejecutor para que ejecute el servicio en sí.
Rol Ejecutor	Hace efectivas las peticiones que se realizan a los servicios. Recibe la descripción del servicio a invocar y los parámetros necesarios y crea una comunicación con dicho servicio para atender la petición.

Se ha de tener en cuenta que una vez definidos y descritos de forma genérica los roles implementados en el sistema, hay que describir cada uno de los protocolos y las interacciones que existen entre ellos. Esto nos ayudará a comprender mejor el funcionamiento de SCODA. Para ello se utiliza el Lenguaje de Modelado de Agentes (AML).

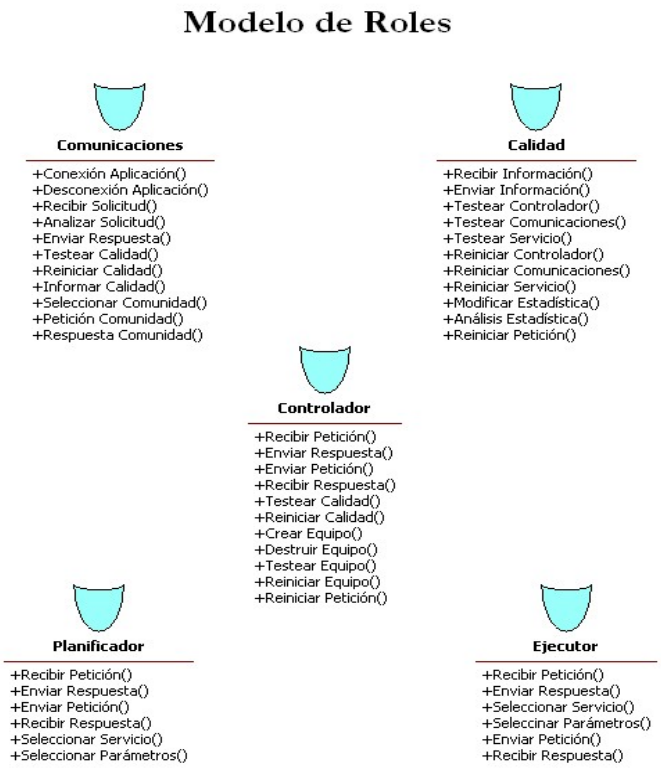


Figura 5.10 Modelo de roles de SCODA utilizando AML

En la **Figura 5.10** se representa el modelo de roles haciendo uso del Lenguaje de Modelado de Agentes. Una vez definidos los roles en el sistema, se describen las interacciones que tienen lugar en el sistema. En las interacciones que intervienen varios roles se necesita una descripción de los protocolos utilizados. Los protocolos definidos son los siguientes (Anexos B.6-32).

Tabla 5.14 Protocolos entre roles

Descripción de los protocolos entre roles existentes en SCODA	
Testear Calidad	Se envía un mensaje de testeo al rol calidad para comprobar su funcionalidad y se recibe confirmación por parte del mismo.
Reiniciar Calidad	En el caso de un mal desempeño por parte del rol calidad se reinicia la entidad que de desempeña este rol.
Informar Calidad	Se envía un mensaje al rol calidad que contenga el resultado de alguna de las actividades realizadas de forma que, este rol calidad pueda llevar a cabo el mantenimiento de la estadística del sistema.

Descripción de los protocolos entre roles existentes en SCODA (Continuación)	
Petición Comunidad	Una vez seleccionada la Comunidad Inteligente Especializada que realizará la ejecución del servicio requerido, se envía una petición por parte del rol comunicaciones a dicha comunidad para que sea resuelta.
Respuesta Comunidad	Cuando se ha finalizado el proceso de ejecución de un servicio determinado por parte de una Comunidad Inteligente Especializada, es el rol controlador de dicha comunidad el que envía la respuesta de su ejecución al rol comunicaciones.
Recibir Información	El rol calidad recibe información de otros roles del sistema. Mediante esta información mantiene la estadística del sistema registrando los movimientos y las acciones que se hayan podido producir.
Enviar Información	El rol calidad envía información de control sobre el funcionamiento a otros roles del sistema cuando exista la necesidad de corregir un determinado comportamiento.
Testear Controlador	Se envía un mensaje de testeo al rol controlador para comprobar su funcionalidad y se recibe confirmación por parte del mismo.
Testear Comunicaciones	Se envía un mensaje de testeo al rol comunicaciones para comprobar su funcionalidad y se recibe confirmación por parte del mismo.
Reiniciar Controlador	En el caso de un mal desempeño por parte del rol controlador se reinicia la entidad que de desempeña este rol.
Reiniciar Comunicaciones	En el caso de un mal desempeño por parte del rol comunicaciones se reinicia la entidad que de desempeña este rol.
Reiniciar Petición	Si existiese un fallo interno en la Comunidad Inteligente Especializada al ejecutar una petición de un servicio, es el rol calidad, el que ordenará volver a ejecutar dicha petición.
Recibir Petición Comunicaciones	El controlador de una determinada Comunidad Inteligente Especializada recibe la petición de servicio sobre la misma.
Enviar Petición Planificador	El rol controlador envía una petición de una solicitud a una instancia de una entidad que implemente el rol planificador.
Enviar Respuesta Comunicaciones	Una vez procesada la respuesta del servicio al que se ha realizado una petición el rol controlador la envía al rol comunicaciones.
Recibir Respuesta Planificador	La respuesta del servicio procesada es enviada por el rol planificador y recibida por el rol controlador de la Comunidad Inteligente Especializada correspondiente.
Crear Equipo	El rol controlador, una vez recibida una solicitud de servicio, instancia un nuevo equipo de forma que crea dos entidades que juegan los roles de planificador y ejecutor, los cuales harán efectiva la petición de servicio.
Destruir Equipo	Una vez procesada la respuesta y comunicada al rol controlador, éste, destruye la instancia del equipo de trabajo que ha ejecutado el servicio, y por consiguiente, ha procesado y comunicado la respuesta.

Descripción de los protocolos entre roles existentes en SCODA (Continuación)	
Testear Equipo	Se envía un mensaje de testeo al los roles planificador y ejecutor para comprobar su funcionalidad y se recibe confirmación por parte del mismo.
Reiniciar Equipo	En el caso de un mal desempeño por parte del equipo correspondiente a una Comunidad Inteligente Especializada, se reinician las entidades que de desempeñan los roles que componen el equipo.
Reiniciar Petición Controlador	El rol controlador, en el caso de recibir una orden del rol calidad de fallo en la comunidad, vuelve a procesar la petición que generó el fallo, llevando a cabo el proceso completo de la misma.
Recibir Petición Controlador	El rol planificador de una determinada Comunidad Inteligente Especializada recibe la petición de servicio del rol controlador.
Enviar Petición Ejecutor	El rol planificador envía una petición de una solicitud a una instancia de una entidad que implemente el rol ejecutor.
Enviar Respuesta Controlador	Una vez procesada la respuesta del servicio al que se ha realizado una petición el rol planificador la envía al rol controlador de la Comunidad Inteligente Especializada a la que está asociado.
Recibir Respuesta Ejecutor	La respuesta del servicio procesada es enviada por el rol ejecutor y recibida por el rol planificador de la Comunidad Inteligente Especializada correspondiente.
Recibir Petición Planificador	El rol ejecutor de una determinada Comunidad Inteligente Especializada recibe la petición de servicio del rol planificador.
Enviar Respuesta Planificador	Una vez procesada la respuesta del servicio al que se ha realizado una petición el rol ejecutor la envía al rol planificador de la Comunidad Inteligente Especializada a la que está asociado.

5.4.2. Diseño GAIA + AML

Finalizada la fase de análisis Gaia, se continúa el desarrollo del modelo con la fase de diseño a través de cual se obtiene el modelo de agentes, los cuales han sido descritos en el apartado 5.2.6. de este capítulo y por ello solamente se tendrá en cuenta el rol que desempeña cada uno de ellos y el número de instancias de los mismos en tiempo de ejecución, el modelo de servicios y el modelo de comunicación o familiaridad.

En cuanto al modelo de agentes, basándonos en las características de las Comunidades Inteligentes, cada agente juega un rol determinado como se describe a continuación en la **Tabla 5.15**, donde se explica el rol desempeñado por cada uno de los agentes que integra SCODA.

Tabla 5.15 Descripción del modelo de agentes para SCODA

Descripción de los roles existentes en SCODA	
CommunicatorAgent	Desempeña el rol comunicaciones, y solamente existe una instancia en tiempo de ejecución de este agente.
QualityAgent	Juega el rol calidad, y existe una instancia únicamente en el sistema en tiempo de ejecución.
ControllerAgent	Juega el rol controlador, y existen tantos agentes de este tipo como Comunidades Inteligentes Especializadas haya en el sistema.
PlannerAgent	Desempeña el rol planificador, y realmente, el número de instancias en tiempo de ejecución tiende a ninguna ya que estos agentes son instanciados y destruidos por el *CommunityControllerAgent* de forma dinámica. Así pues, las instancias de este agente dependen del número de peticiones que se realicen sobre un servicio correspondiente a su Comunidad Inteligente Especializada.
ExecutorAgent	Desempeña el rol ejecutor y tiene el mismo comportamiento en tiempo de ejecución que el rol planificador ya que, junto a éste, conforma el equipo de trabajo que es instanciado por el *CommunityControllerAgent* de su Comunidad Inteligente Especializada.

La **Figura 5.11** muestra la modelización en AML del modelo de agentes. En esta figura la se puede ver la representación de que cada uno de los agentes que integra SCODA juega solamente un rol determinado. Además en la modelización podemos distinguir que el símbolo que une a los agentes con los roles posee un número que representa la cantidad de agentes en tiempo de ejecución. Así pues, solamente habrá un *CommunicatorAgent* y un *QualityAgent*, sin embargo habrá n *CommunityControllerAgent* con sus equipos respectivos.

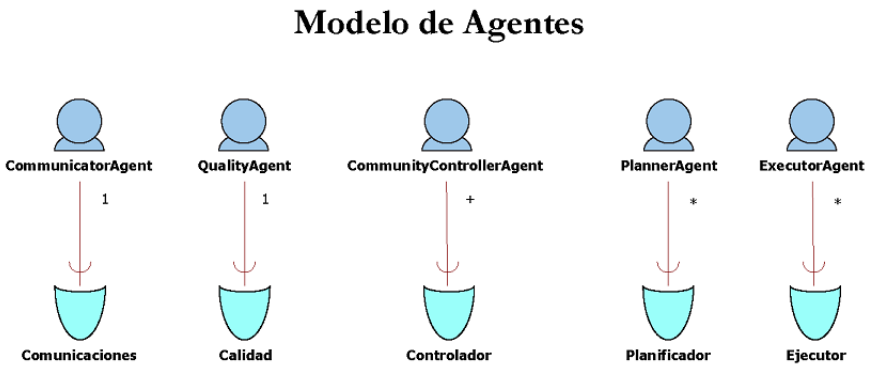


Figura 5.11 Modelo de agentes en SCODA utilizando AML

Por su parte, el modelo de servicios ha de mostrar los servicios asociados a cada agente. Debido a la filosofía que persigue esta arquitectura, los servicios están asociados a las Comunidades Inteligentes Especializadas, siendo estos servicios los que dotan de funcionalidad a las mismas. Así pues, cada Comunidad Inteligente Especializada, que consta de un *CommunityControllerAgent*, que a su vez podrá instanciar tantos equipos de trabajo compuestos por un *PlannerAgent* y un *ExecutorAgent* como sean necesarios, tendrá asociados una serie de servicios especializados, lo que quiere decir que estos servicios se enmarcan en un entorno determinado que no puede ser definido en un modelado genérico. El modelo de servicios se representa en la **Figura 5.12.**

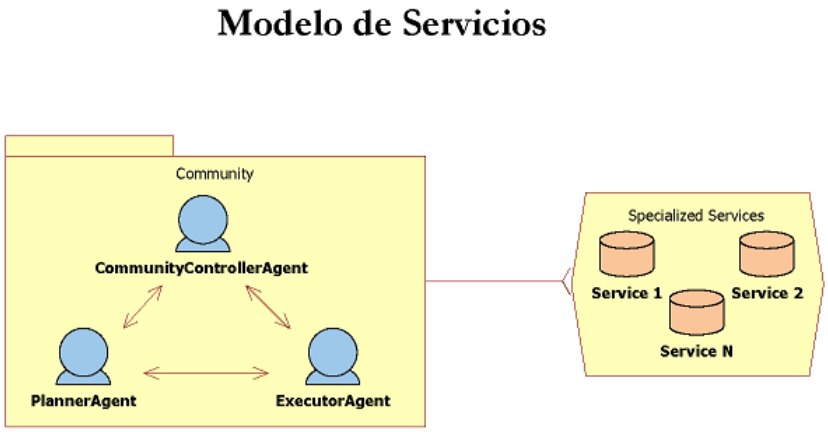


Figura 5.12 Modelo de servicios de SCODA utilizando AML

El modelado de estos servicios no se realiza especificando cada una de sus funciones con sus parámetros ya que su implementación es específica de cada Comunidad inteligente Especializada y por lo tanto un sistema multiagente basado en SCODA contará con tantos servicios especializados como se requiera.

Finalmente, a través del modelo de comunicación se definen los enlaces existentes entre los agentes del sistema. Este modelo tiene como objetivo la definición de las posibles comunicaciones entre los agentes sin entrar en detalles de las mismas (mensajes, formato de mensajes, etc.). El modelo de comunicación definido para SCODA se muestra en la **Figura 5.12.**

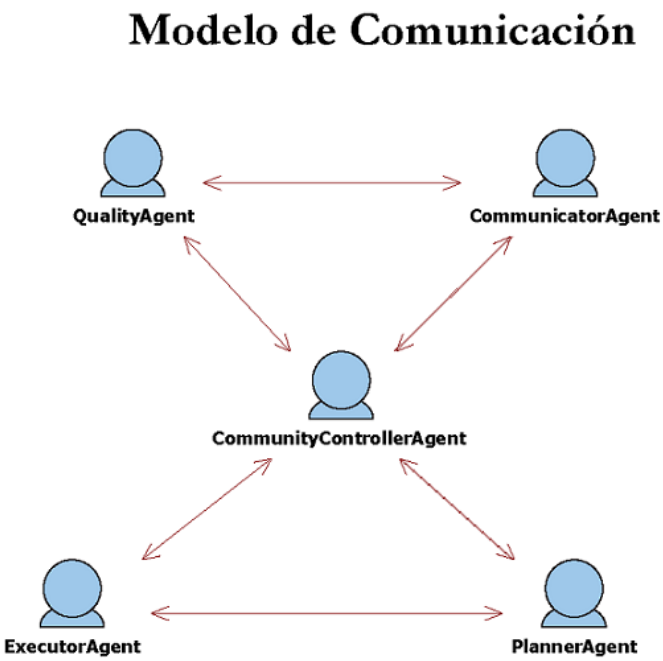


Figura 5.13 Modelo de comunicación de SCODA utilizando AML

5.4.3. MODELADO A TRAVÉS DE AML

Una vez finalizado el análisis y diseño mediante GAIA + AML, existen ciertos aspectos del sistema que no han sido modelados debido a que GAIA provee un análisis y diseño de alto nivel que no capacita al diseñador para ello, para lo cual se hace uso exclusivo del Lenguaje de Modelado de Agentes *(AML)*. A través de GAIA se ha definido la arquitectura del sistema con un alto nivel de abstracción, y se ha ido modelando a través de AML, obteniendo así una representación cercana a la implementación, sin embargo existen aspectos de la arquitectura que no han sido contemplados en las fases de análisis y diseño GAIA, como son el modelado de aspectos sociales, representado en la **Figura 5.14.**, y el modelo de despliegue del sistema representado en la **Figura 5.15.**

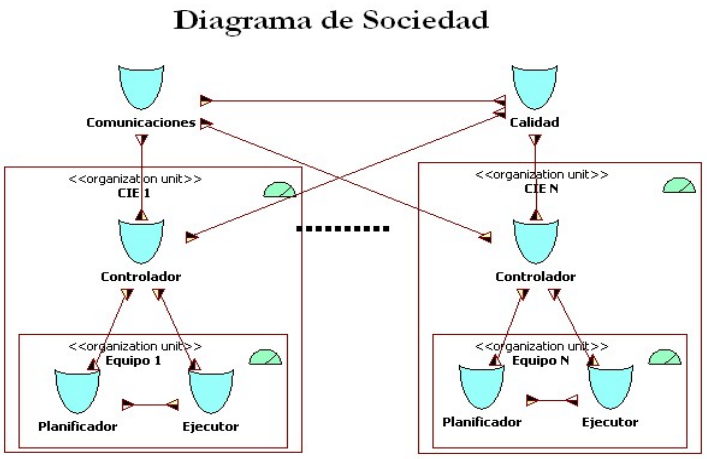


Figura 5.14 Modelado de los aspectos sociales de SCODA utilizando AML

A través del modelado de los aspectos sociales, se obtiene la representación de las relaciones organizacionales del sistema y de su estructura, características sociales de los roles que lo componen y ciertos aspectos de su comportamiento.

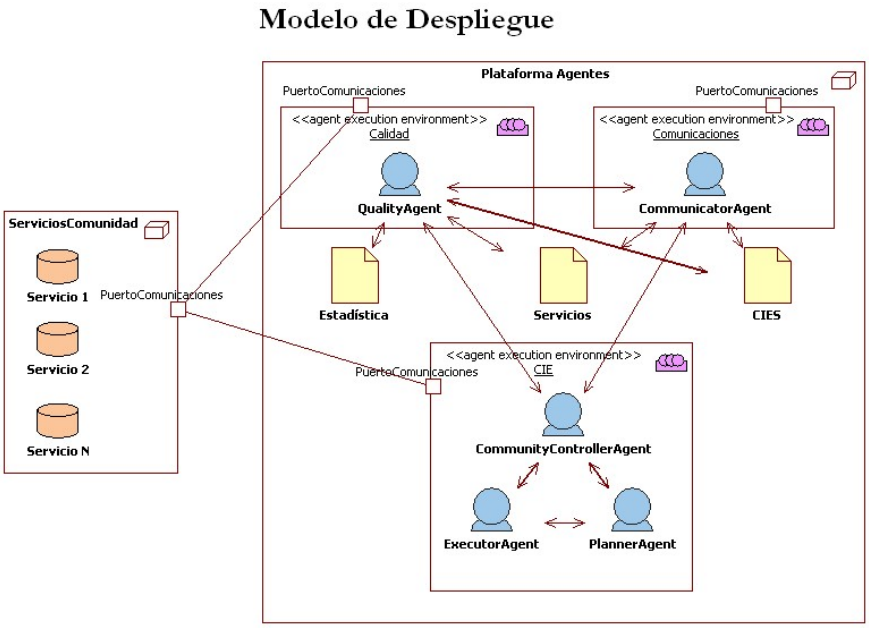


Figura 5.15 Modelado de despliegue de SCODA utilizando AML

Mediante el modelo de despliegue, obtenemos una representación del sistema en un entorno de ejecución con los recursos necesarios y la relación entre los distintos componentes del sistema multiagente.

5.4.3.1. MODELADO DEL COMPORTAMIENTO AML

AML permite reflejar el modelado del comportamiento del sistema y de sus agentes, en cuanto a la capacidad de razonamiento de los mismos, sobre eventos u operaciones a realizar (Anexos B.33-37). Este comportamiento dependerá del rol que cada agente adopte, así pues, se corresponderán con las actividades y protocolos definidos en el análisis Gaia, en el apartado 5.4.1. de este capítulo, para cada uno de dichos roles. Teniendo en cuenta esto, a continuación se definen los comportamientos de cada uno de los agentes que integran SCODA:

- **CommunicatorAgent:** Su comportamiento se atribuye a las capacidades de conexión y desconexión de una aplicación externa, recibir solicitudes de servicio y respuestas del mismo, enviar peticiones de servicio y enviar la respuesta del mismo a la aplicación externa que haya solicitado el servicio. Además, este agente, tiene la capacidad de testear e informar al *QualityAgent* de las actividades realizadas, y de reiniciar la instancia del mismo en el caso de que fuese necesario.

- **QualityAgent:** El comportamiento de este agente se debe a una serie de capacidades que le permiten llevar un control del sistema y de las entidades que lo conforman. Este agente puede recibir y enviar información de control y corrección a otros agentes del sistema, puede testear y reiniciar a los *CommunityControllerAgent* y *CommunicatorAgent*, y a los servicios especializados que proveen las Comunidades Inteligentes Especializadas. Además, lleva un control de las actividades que se realizan en el sistema y analiza las mismas para mejorar el rendimiento de los agentes y del sistema en sí. Finalmente puede volver a realizar una petición de servicio si detecta que ha sido defectuosa en primera instancia.

- **CommunityControllerAgent:** Este agente tiene un comportamiento guiado por una serie de capacidades para gestionar las Comunidades Inteligentes Especializadas. Es capaz de recibir solicitudes de servicio del *CommunicatorAgent* y enviarle la respuesta una vez terminada su ejecución, también tiene la capacidad de testear, informar y reiniciar al *QualityAgent*. En cuanto a la gestión de su Comunidad Inteligente Especializada, puede crear un equipo de trabajo, compuesto por un *PlannerAgent* y un *ExecutorAgent*, enviarle una solicitud de servicio y destruirlos una vez finalizada su ejecución. Además puede ordenar el reinicio de un equipo en caso de un funcionamiento incorrecto del mismo o reiniciar una solicitud de servicio si recibe una orden del *QualityAgent*.

- **PlannerAgent:** El comportamiento de este agente viene dado por las diferentes capacidades de comunicación con el

CommunityControllerAgent de la Comunidad Inteligente Especializada a la que pertenece, y con el ExecutorAgent miembro de su equipo, y las capacidades de planificación y razonamiento para seleccionar un servicio y los parámetros necesarios para su correcta ejecución. Así pues, puede recibir solicitudes de servicio y enviar la respuesta una vez finalizado a su *CommunityControllerAgent*, realizar la planificación y selección del servicio adecuado y de sus parámetros, y finalmente solicitar al *ExecutorAgent*, miembro de su equipo, la ejecución de dicho servicio, manteniéndose a la espera de la respuesta de la ejecución del mismo.

- **ExecutorAgent:** Este agente centra su comportamiento en hacer efectivas las ejecuciones de los servicios solicitados por las aplicaciones externas. Así pues, tiene la capacidad de recibir una solicitud de servicio del PlannerAgent de su equipo, en la que se especifica el servicio que ha de ejecutar y los parámetros para hacerlo, conectar con el servicio necesario, recibir la respuesta de su ejecución, desconectar el servicio una vez finalizada su ejecución y enviar la respuesta al *PlannerAgent*.

5.4.3.2. MODELADO DEL CONTEXTO AML

AML provee la capacidad de modelar el entorno en el que se desenvuelven los agentes. Como se muestra en la **Figura 5.16.,** y de acuerdo a situaciones que puedan ocurrir en tiempo de ejecución, se permite el modelado de situaciones particulares de los agentes en momentos determinados los cuales se denominan contextos. Así pues, los agentes que implementan los roles Planificador y Ejecutor son instanciados en el contexto *CIE*, y a su vez se ejecutan en el contexto *Equipo*.

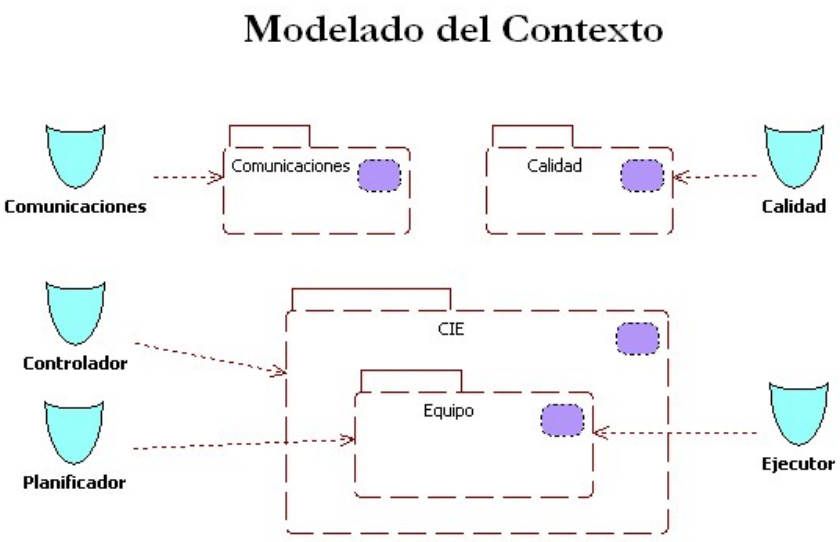


Figura 5.16 Modelo del contexto de SCODA utilizando AML

5.4.3.3. MODELADO DE LAS INTERACCIONES AML

A través de AML es posible modelar las interacciones existentes entre entidades o grupos de entidades a través de protocolos de interacción. Para ello, AML provee una serie de diagramas extendidos de los diagramas de interacción de UML, que permiten modelar las interacciones entre agentes y grupos de agentes a través de mensajes y señales, como son los que se presentan en la **Tabla 5.17**:

Tabla 5.17 Diagramas para modelar las interacciones en AML

Diagramas de Interacción en AML	
Diagramas de Colaboración de Protocolos	Los diagramas de colaboración (Anexos B.38-46), también representan interacciones entre los agentes, sin embargo se centran en la asociación existente entre ellos, representando las relaciones entre los mismos.
Diagramas de Secuencia de Protocolos	Los diagramas de secuencia (Anexos B.47-55) representan las interacciones que existen entre los agentes del sistema, mostrando el intercambio de mensajes entre los mismos.
Diagramas de Estado	Los diagramas de estados (Anexos B.56 – B.60) nos permiten observar el comportamiento a nivel interno de los agentes. Estos diagramas muestran los estados por los que pasa cada agente, y los eventos producidos en los cambios de estados.

A continuación se presenta la explicación de cada uno de los diagramas realizados. Se ha de tener en cuenta que los casos representados en los diagramas de secuencia de protocolos y en los de colaboración de protocolos son los mismos, con lo que la interacción entre los agentes comentada a continuación, también lo es.

En los Anexos B.38 y B.47, se muestra una solicitud exitosa de forma completa. Así pues, el *CommunicatorAgent* recibe una solicitud de servicio, la cual es analizada para seleccionar la Comunidad Inteligente Especializada correspondiente y se informa al *QualityAgent* que existe una solicitud de un servicio para que procese la estadística del sistema. Acto seguido se el *CommunicatorAgent* realiza una petición de servicio al *CommunityControllerAgent* seleccionado, el cual crea un equipo que será el encargado de llevarla a cabo, informando al *QualityAgent* del éxito en la creación del equipo compuesto por un *PlannerAgent* y un *ExecutorAgent*. El *PlannerAgent* realiza una selección del servicio adecuado de los que dispone la comunidad y de los parámetros óptimos para llevar a cabo la solicitud inicial, siendo esta información transmitida al *ExecutorAgent* que ejecuta el servicio adecuado manteniéndose a la espera del resultado. Una vez realizado el proceso de la solicitud inicial el *ExecutorAgent* envía la respuesta al *PlannerAgent*, que a su vez la envía al *CommunityControllerAgent* que creó el equipo, destruyendo al mismo ya que la petición se ha resuelto correctamente. Una vez destruido el equipo, el *QualityAgent* es informado del éxito de la ejecución del servicio y se transmite la misma al *CommunicatorAgent* que mostrará la respuesta en el formato requerido, informando al *QualityAgent* del éxito completo de la ejecución del servicio solicitado inicialmente que actualizará la estadística del sistema.

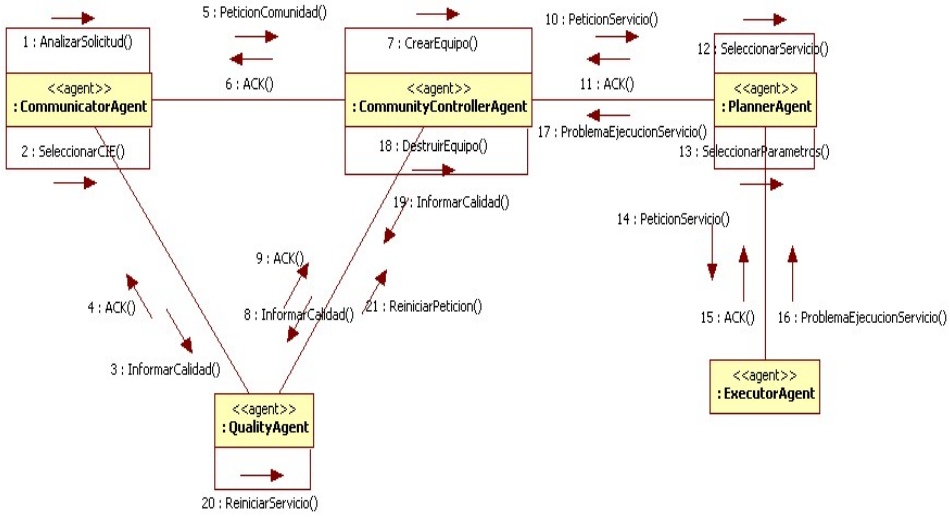


Figura 5.17 Ejemplo de diagrama de Colaboración de Protocolos para una solicitud con error por problema de ejecución del Servicio de una Comunidad Inteligente Especializada

En los Anexos B.39 y B.48, se muestra una solicitud errónea debido a un funcionamiento defectuoso en la selección de parámetros por parte del *PlannerAgent*. Partiendo del diagrama anterior, se realizan los mismos pasos hasta que el *ExecutorAgent* comunica al *PlannerAgent* que el servicio no se ha ejecutado correctamente debido a un error en los parámetros necesarios para el procesamiento del mismo. El *PlannerAgent* comunica este error al *CommunityControllerAgent* que a su vez informa al *QualityAgent* del mismo y destruye el equipo que había llevado a cabo la petición errónea. El *QualityAgent*, reconoce el tipo de error, actualiza la estadística del sistema y envía una orden de reinicio de petición al *CommunityControllerAgent*, que vuelve a crear otro equipo para procesar el servicio requerido inicialmente.

Por su parte, los Anexos B.40 y B.49, muestran una ejecución errónea de un servicio debido a la imposibilidad de creación del equipo que la ejecutará, por parte del *CommunityControllerAgent*. Como en los casos anteriores, el *CommunicatorAgent* recibe una solicitud de un servicio, la cual procesa y selecciona la Comunidad Inteligente Especializada correspondiente, siempre informando al *QualityAgent*. El *CommunityControllerAgent* de dicha comunidad crea un equipo para que procese la petición, sin embargo ocurre un fallo en la creación del mismo. El *QualityAgent* es informado de la incidencia, actualiza la estadística del sistema y ordena al *CommunityControllerAgent* la creación de un nuevo equipo que tenga la capacidad de atender la solicitud inicial.

Los diagramas mostrados en los Anexos B.41 y B.50, hacen referencia a una solicitud de servicio errónea debido a que los servicios de una Comunidad Inteligente Especializada están inactivos. Para ello partimos de los diagramas que

aparecen en los Anexos B.38 y B.47, donde se muestra una solicitud exitosa completa. El proceso de solicitud es el mismo hasta que el *ExecutorAgent* comunica al *PlannerAgent*, del equipo al que pertenece, que el servicio requerido está inactivo. El *PlannerAgent*, por su parte, comunica la incidencia al *CommunityControllerAgent* de la comunidad, que a su vez destruye el equipo e informa al *QualityAgent*. Éste por su parte, analiza la incidencia, actualiza la estadística del sistema y ejecuta el reinicio de los servicios asociados a la Comunidad Inteligente Especializada donde ha surgido la incidencia. Finalmente ordena al *CommunityControllerAgent* de dicha comunidad reiniciar la petición.

En los Anexos B.42 y B.52, se muestra un error en la solicitud de un servicio debido a que no existe ninguna Comunidad Inteligente Especializada que pueda atenderla. El *CommunicatorAgent* recibe la solicitud de un servicio, la cual analiza. Acto seguido selecciona una Comunidad Inteligente Especializada con capacidad para atender la solicitud, no encontrando ninguna. Ante este hecho, informa de la incidencia al *QualityAgent* que actualiza la estadística del sistema y ordena responder a la aplicación externa que solicita el servicio de la imposibilidad de atender la solicitud debido a que no existe ningún servicio adecuado para la misma.

Los Anexos B.43 y B.51 muestran un error provocado por una mala ejecución de un servicio perteneciente a una Comunidad Inteligente Especializada. A partir de los diagramas representados en los Anexos B.38 y B.47, la solicitud del servicio sigue la misma secuencia, sin embargo es el *ExecutorAgent* el que comunica al *PlannerAgent* de su equipo un fallo en la ejecución del servicio que se le ha requerido, informando éste a su vez al *CommunityControllerAgent* que lo ha creado. El *CommunityControllerAgent* destruye el equipo e informa al *QualityAgent* del error provocado por un problema en la ejecución del servicio, el cual, procesa el error actualizando la estadística del sistema y ordenado al *CommunityControllerAgent* que reinicie la petición.

Finalmente, los Anexos B.44-46 y B.53-55 muestran los sendos testaos que se realizan sobre el *QualityAgent*, el *CommunicatorAgent* y sobre los diversos *CommunityControllerAgent* que coexistan en el sistema, de forma que el *QualityAgent* realiza el testeo sobre los *CommunityControllerAgent* y el *CommunicatorAgent*, y cada uno de estos lo hace sobre el *QualityAgent*. Este testeo consiste en un mensaje que confirme el estado del agente que se pretende controlar, y en el caso de que no exista respuesta por su parte se procede al reinicio del mismo.

Una vez finalizada la modelización a través de los Diagramas de Colaboración de Protocolos se da paso a la modelización de los estados por los que pasa cada uno de los agentes integrantes de SCODA. Para ello, en el Anexo B.56 se muestran los estados por los que pasa el *CommunicatorAgent* y las interacciones entre los mismos. Este agente cuenta con cuatro estados principales descritos en la **Tabla 5.18**:

Tabla 5.18 Diagrama de Estados del CommunicatorAgent en AML

Estado	Descripción
Solicitudes y Respuestas	Es el estado que mantiene al agente en espera de nuevas solicitudes o comunicaciones de otros agentes, y permite enviar solicitudes o información a otros agentes.
Análisis de Solicitudes	En este estado el agente analiza una determinada solicitud que le ha llegado por parte de una aplicación externo.
Selección de *CIE*	Una vez analizada la solicitud, se procede a la selección de la Comunidad Inteligente Especializada que tendrá la capacidad de atender una solicitud dada.
Reinicio Entidades	Cuando se posiciona en este estado tiene la capacidad de reiniciar al *QualityAgent* por un funcionamiento defectuoso.

El *QualityAgent* también consta de cuatro estados principales. Estos estados y sus interacciones pueden verse en el Anexo B.57, y están descritos en la **Tabla 5.19**:

Tabla 5.19 Diagrama de Estados del QualityAgent en AML

Estado	Descripción
Información y Respuesta	Es el estado que mantiene al agente en espera de nuevas comunicaciones de otros agentes, y permite enviar información a otros agentes para su correcto funcionamiento.
Reinicio de Entidades	Cuando se posiciona en este estado tiene la capacidad de reiniciar al *CommunicatorAgent*, al *CommunityController* o algún servicio por un funcionamiento defectuoso.
Reinicio Solicitudes	Una vez analizada una solicitud defectuosa, en este estado se procede a ordenar al *CommunityControllerAgent* correspondiente el reinicio de la misma.
Actualización Estadística	En este estado, el *QualityAgent* modifica la estadística de funcionamiento del sistema, y analiza la misma en busca de comportamientos erróneos, por parte de los agentes, para llevar a cabo una corrección de los mismos.

Por su parte, el *CommunityControllerAgent* también posee cuatro estados principales que se muestran en el Anexo B.58. Su descripción pude verse en la **Tabla 5.20.**

Tabla 5.20 Diagrama de Estados del CommunityControllerAgent en AML

Estado	Descripción
Espera Solicitudes	Es el estado que mantiene al agente en espera de nuevas solicitudes o comunicaciones de otros agentes, y permite enviar solicitudes o información a otros agentes.
Creación Equipo	En este estado el agente realiza la instanciación de un *PlannerAgent* y un *ExecutorAgent* que compondrán el equipo responsable de la ejecución del servicio que pueda resolver una solicitud dada.
Destrucción Equipo	Una vez finalizada la ejecución de un servicio, ya sea de forma correcta o incorrecta, el *CommunityControllerAgent* destruye las instancias del *PlannerAgent* y del *ExecutorAgent*, correspondientes al equipo que ha llevado a cabo la ejecución del servicio.
Reinicio Entidades	Cuando se posiciona en este estado tiene la capacidad de reiniciar al *QualityAgent* y a un equipo, compuesto por un *PlannerAgent* y un *ExecutorAgent*, debido a un funcionamiento defectuoso.

Por su parte, el *PlannerAgent* cuenta solamente con tres estados principales como se muestra en el Anexo B.59, y se describen en la **Tabla 5.21**.

Tabla 5.21 Diagrama de Estados del PlannerAgent en AML

Estado	Descripción
Solicitudes y Respuestas	Es el estado que mantiene al agente en espera de nuevas solicitudes o comunicaciones de otros agentes, y permite enviar solicitudes o información a otros agentes.
Solicitud Servicio	En este estado el agente realiza la selección del servicio más adecuado, que se ajuste a la solicitud, de los que posee la Comunidad Inteligente Especializada.
Selección Parámetros	Una vez seleccionado el servicio adecuado, en este estado el *PlannerAgent* procede a seleccionar los parámetros necesarios, en el caso de que existan, para que la ejecución del servicio sea correcta.

Finalmente, el *ExecutorAgent*, también cuenta solamente con tres estados principales. Estos estados se muestran en el Anexo B.60, y se describen en la **Tabla 5.22.**

Tabla 5.22 Diagrama de Estados del ExecutorAgent en AML

Estado	Descripción
Solicitudes y Respuestas	Es el estado que mantiene al agente en espera de nuevas solicitudes o comunicaciones de otros agentes, y permite enviar solicitudes o información a otros agentes.
Conexión Servicio	En este estado el agente se conecta con el servicio seleccionado por el *PlannerAgent* haciéndole llegar los parámetros necesarios para su ejecución.
Desconexión Servicio	Una vez finalizada la ejecución del servicio y recibida la respuesta el *ExecutorAgent* realiza la desconexión del servicio.

5.5. COMMUNITY AGENT GENERATOR

La implementación de sistemas multiagente siguiendo la estructura de SCODA ha de seguir los principios en los que ésta se basa, introducidos en la sección 5.2.: estandarización, especialización, facilidad de implementación, reutilización y computación distribuida. Para ello, se ha desarrollado una herramienta, ***Community Agent Generator***, capaz de llevar a cabo la labor de implementar la estructura de sistemas multiagente basados en SCODA, así como de integrar diferentes Comunidades Inteligentes Especializadas en dicha estructura. La implantación y el despliegue que realiza esta herramienta se hace sobre la plataforma JADEX (Pokahr *et al.*, 2003; 2007), habiendo seleccionado esta plataforma, ya que es considerada como un motor de razonamiento y puede ser ejecutado de forma independiente.

La herramienta *Community Agent Generator* está desarrollada sobre Java en entorno Web, y permite la creación, integración y despliegue de arquitecturas SCODA de forma dinámica. En la **Figura 5.18** se muestra la pantalla principal de esta herramienta, que permite observar las opciones que ofrece la aplicación:

- Exploración de los ficheros necesarios en la implementación de la arquitectura.
- Gestión de las Comunidades Inteligentes Especializadas
- Configuración interna de la aplicación
- Ayuda de utilización de la aplicación para los desarrolladores

La capacidad de la herramienta para generar la estructura de arquitecturas de tipo SCODA, y de las Comunidades Inteligentes Especializadas necesarias para logar una implementación de un sistema multiagente, facilita la labor de los desarrolladores. Esto es debido al principio de estandarización que persigue esta arquitectura, ya que su estructura es la misma en diferentes implementaciones.

Figura 5.18 Pantalla principal de la herramienta Community Agent Generator

Además de generar la estructura de la arquitectura de forma automática, la herramienta Community Agent Generator, ofrece la posibilidad de editar el código generado, tanto de los propios agentes que componen la arquitectura, como de los servicios asociados a las Comunidades Inteligentes Especializadas, como se muestra en la **Figura 5.19** Esta posibilidad, permite llevar a cabo la programación de estos servicios y la modificación de los mismos en el caso de que fuera necesario.

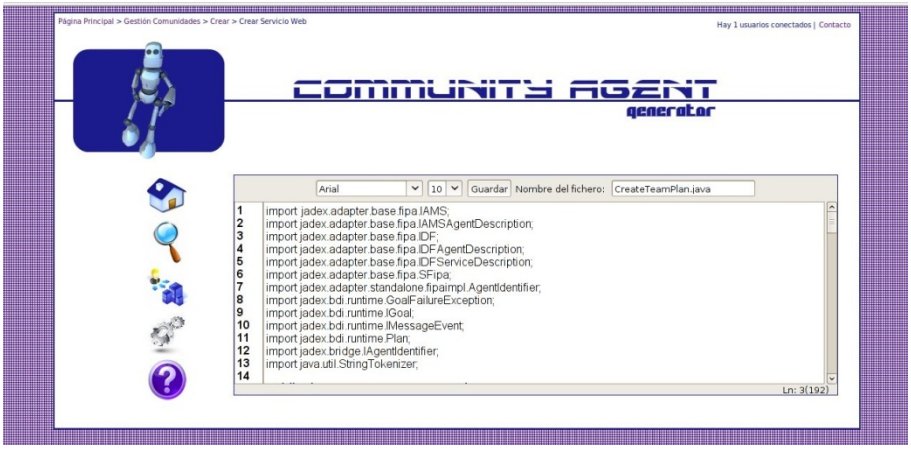


Figura 5.19 Editor de la herramienta Community Agent Generator

La herramienta Community Agent Generator cuenta con un conjunto de opciones para la gestión de arquitecturas SCODA dentro de las que se encuentran la realización de copias de seguridad, parciales o totales, búsquedas sobre servicios y comunidades, descripciones de funcionalidad y la visualización en diversas lenguas. La implementación de esta herramienta sobre un entorno Web permite realizar el

desarrollo de sistemas multiagente sobre SCODA, y su posterior despliegue de forma remota, facilitando así la labor de los desarrolladores y dinamizando el proceso requerido en el mismo.

5.6. CONCLUSIONES

La reutilización de entidades computacionales en el desarrollo de sistemas informáticos es una característica presente en la programación orientada a objetos (Lieberherr, 1996; Elrad *et al.,* 2001; Rashid *et al.,* 2003; Kiczales *et al.,* 1997). Esta característica facilita la labor de los desarrolladores debido a que pueden reutilizar los mismos objetos en multitud de desarrollos, acelerando así el tiempo necesario en su implementación. Este planteamiento llevado a los sistemas multiagente se presenta a través de la arquitectura SCODA, de forma que un grupo reducido de agentes, con la misma estructura y denominado Comunidad Inteligente Especializada, pueda tener la capacidad de ejecutar diferentes tareas, en función de los servicios asociados a este grupo.

La arquitectura SCODA, basada en unidades organizativas como son las Comunidades Inteligentes de agentes, tiene la capacidad de integrar la tecnología de agentes con una filosofía de distribución de los servicios que procesan los mismos utilizando REST *(REpresentational State Transfer)* (Fielding, 2000), lo cual dota de una mayor versatilidad en cuanto a las solicitudes de servicios externas que lleguen a SCODA.

Esta arquitectura ofrece un nuevo marco para el desarrollo de sistemas Multiagente. Aprovechando las características de las Comunidades Inteligentes llevadas a los sistemas de agentes. Para ello se basa en el funcionamiento de las *"Redes Empresariales"* (Sonquist y Koenig, 1975; Galaskiewicz, 1979; Burt, 1980; Walker *et al.,* 1997; Lazer, 2003; Borgatti y Foster, 2003; Tracey y Clark, 2003; Johansson y Quigley, 2004; Viedma, 2004; Pöyhönen y Smedlund, 2004; Ibarra *et al.,* 2005; Eraydin y Armatli-Köroglu, 2005; Cabus y Vanhaverbeke, 2006), dotando así de una mayor facilidad su desarrollo. La capacidad de reutilización de las Comunidades Inteligentes Especializadas, de las que se compone SCODA, la posibilidad de escalado entre las mismas y la distribución de los servicios que ofrecen cada una de ellas, hace de SCODA una arquitectura ligera en cuanto a la forma de distribuir la carga computacional que procesan sus agentes. La hace flexible en cuanto a la posibilidad de crecimiento de las funcionalidades que pueda ofrecer a través de la unión de varias Comunidades Inteligentes Especializadas. Y reutilizable debido a la especialización de estas comunidades, que tendrán asociados una serie de servicios determinados pudiendo ser reutilizados en diversos desarrollos.

SCODA proporciona un exhaustivo control en la gestión y recuperación de errores dentro del sistema y los servicios asociados al mismo. Mantiene en todo momento una estadística de funcionamiento, mediante la que se permite adoptar decisiones y medidas en caso de un funcionamiento defectuoso del sistema y de los servicios asociados al mismo. A partir de protocolo de comunicación REST que

adopta SCODA se 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.

La implementación SCODA se ha realizado sobre la plataforma JADEX, debido a que soporta agentes deliberativos BDI y porque se considera un motor de razonamiento que puede ser ejecutado de forma independiente. Esta implementación sobre JADEX ha permitido comprobar la validez del modelo teórico de la arquitectura hipótesis principal de este trabajo. Este hecho permite continuar con nuevos desarrollos sobre diferentes plataformas, dando continuidad y estabilidad a este modelo teórico.

CAPÍTULO 6. CASO DE ESTUDIO

6. CASO DE ESTUDIO

En el presente capítulo, se describe la aplicación de la arquitectura propuesta en este trabajo, en un entorno empresarial. Se propone la creación de un sistema multiagente, basado en SCODA, compuesto por diversas Comunidades Inteligentes Especializadas (CIE), enfocado a la resolución de problemas logísticos como la gestión de inventarios de productos alimentarios (Zipkin, 2000; Silver *et al.*, 1998; Guide y Srivastava, 1997; Parlarm y Berkin, 1991; Love, 1979; Hadley y Whitin, 1963) y optimización de rutas (Golden *et al.*, 1977; Axhausen y Smith, 1984; Baaj y Mahmassani, 1991; Laporte, 1992; Díaz *et al.*, 1996; Shih *et al.*, 1998; Biswas *et al.*, 2005; Lüer *et al.*, 2009).

Siguiendo la estructura definida en el capítulo 5 de esta memoria, las CIE que componen la arquitectura SCODA, del caso de estudio, están compuestas por un conjunto de agentes inteligentes con capacidad de razonamiento, planificación y cooperación, tanto a nivel interno de cada CIE, como entre ellas. Además cada CIE tiene la capacidad de trabajar de forma autónoma pudiendo prestar sus servicios especializados en otros problemas implementados con esta arquitectura. Es por ello que el desarrollo de las diferentes CIE que conforman el sistema, serán diseñadas de forma independiente, con el fin de que puedan ser reutilizadas, siguiendo las especificaciones y la metodología propuesta en el capítulo 5. Además, debido a la tipología del caso de estudio que está basado en la integración y funcionamiento coordinado del modelo teórico, se pretende llevar a cabo una serie de pruebas de rendimiento de este modelo que demuestre la validez de la hipótesis de partida.

El presente capítulo se divide varias secciones. En la sección 6.1. se realiza una introducción al problema que se abordará. La sección 6.2. muestra el planteamiento del problema introduciendo los conceptos necesarios para la resolución del mismo. En la sección 6.3. se aborda el desarrollo del sistema en los términos propuestos. La sección 6.4. presenta una comparativa de SCODA con un sistema desarrollado para el mismo fin, donde se examinarán los resultados obtenidos por ambos. En la sección 6.5 se analiza la especialización del sistema del caso de estudio atendiendo al sistema de medición propuesto en el capítulo 3 de esta memoria. Finalmente, en la sección 6.6 se lleva a cabo un análisis del caso de estudio y las conclusiones obtenidas en su implementación.

6.1. INTRODUCCIÓN AL CASO DE ESTUDIO

Las empresas dedicadas a la distribución de productos congelados, y más concretamente las que comercializan estos productos a particulares, han de tener en cuenta varios factores para maximizar sus ganancias como son las rutas a seguir, una predicción en las ventas que haga que no exista carencia de productos, y a su vez permita realizar pedidos que ofrezcan algún tipo de descuento y rapidez en los

procedimientos intermedios de abastecimiento de cada medio de transporte de mercancía de la propia empresa entre otros.

El caso que nos ocupa es el de la empresa **Bahía Príncipe Congelados S.L.** situada en la provincia de Zamora. Esta empresa tiene como actividad principal la comercialización y distribución de productos congelados en las provincias de Salamanca, Valladolid, León y Zamora principalmente. Su labor es, por una parte, comercializar los productos directamente en los domicilios de los clientes a través de camiones frigoríficos y congeladores perfectamente adaptados para llevar a cabo esta tarea, y por otra atender a los pedidos que restaurantes, hoteles y entidades de este tipo, realizan de forma directa a la empresa.

La empresa comercializa sus productos de lunes a viernes, comenzando la jornada laboral de sus trabajadores a las 8.30 horas. Cada vendedor es responsable de la carga de su camión y ha de llevar un control de las ventas de cada una de las referencias de los productos y así poder reponer los productos que se hayan agotado o esté a punto de hacerlo. Este es uno de los problemas que abordaremos ya que hay una importante pérdida de tiempo a la hora de comprobar que cantidad deben ser repuestos.

Otro de los problemas existentes, a nivel de los productos, es la reserva que exista en la cámara principal, la cual tiene que abastecer a cada uno de los camiones además de a los pedidos que se realicen. Para ello se ha de tener en cuenta que desde la oficina central han de controlar el stock de cada uno de los productos, pero además predecir de una forma lo más exacta posible las cantidades de productos que saldrán de la cámara para poder abastecerla.

Para ello la empresa dispone de un software realizado a medida que funciona como software de facturación y de control de stock en la cámara principal. Sin embargo este software no tiene la capacidad de realizar una predicción del aprovisionamiento de los productos para cada uno de los camiones, ni permite determinar el lote óptimo de productos que se solicitarán a los proveedores en función de la demanda que exista.

Finalmente otro de los aspectos relevantes que se ha de abordar consiste en el análisis y planteamiento de las rutas que siguen los camiones que comercializan los productos en el domicilio de los clientes. Estos camiones realizan una ruta diferente cada día de la semana, realizando varios cientos de kilómetros diarios, lo cual implica un consumo de combustible importante y su minimización es más que conveniente para contribuir a maximizar el beneficio de la empresa.

A través de SCODA se pretenden mejorar los procedimientos llevados a cabo en esta empresa y con ello maximizar los beneficios, tanto económicos como organizacionales en función del tiempo y el trabajo de los propios integrantes de la empresa. Para ello, la citada empresa ha facilitado datos relativos a sus clientes, productos, rutas, y metodología de trabajo, sin embargo ha incidido en que la

tipificación tanto de sus rutas como de sus productos y clientes, se lleve a cabo con nombres genéricos por cuestiones de competencia.

6.2. PLANTEAMIENTO DEL PROBLEMA

En función de la metodología de trabajo llevada a cabo por esta empresa y de la gran cantidad de productos que comercializan es imprescindible poder estimar de forma precisa las cantidades de productos que se demandarán, y con ello la previsión de stock necesaria en sus cámaras principales. Conocer la demanda de productos implica predecir la cantidad de los mismos que se requerirá satisfaciendo todas las peticiones que pudiera haber por parte de los clientes. Además existen una serie de factores que influyen de forma importante sobre la demanda de producto mostrados en la **Tabla 6.1.**

Tabla 6.1 Factores que influyen en la demanda de productos

Factor	Descripción
Calendario	En función de la época del año, el mes, y el día de la semana, la demanda de este producto varía de forma importante. Se ha de tener en cuenta los días festivos y periodos vacacionales en los que se realza el turismo de interior y por lo tanto la ocupación hotelera de la zona. También se ha de resaltar, sobre la tendencia de particulares que pasan estos periodos en diversas localidades a adquirir este producto, que los días festivos se nota un incremento en la demanda del mismo. Así pues, el calendario es un factor importante a considerar en las estimaciones que se realicen sobre la demanda del producto.
Zona Geográfica	Si bien es cierto, a lo largo de la geografía que esta empresa atiende existe gran variedad en cuanto a la comercialización de productos se refiere. Cada una de estas zonas cuenta con una serie de clientes que por lo general consumen los mismos productos, lo cual hace que ha de existir una previsión de productos en función del día de la semana y de la zona a visitar ya que habrán de pasar siete días hasta que vuelvan a visitar esta zona. Este factor también influye en la cantidad de kilómetros que realiza cada vendedor, lo cual implica que una minimización de los mismos reorganizando las visitas optimizaría el consumo de combustible, el tiempo, y por lo tanto el rendimiento de la empresa.
Espacio Temporal	Este factor se refiere a tiempo que el trabajador pasa conduciendo de una localidad a otra sin que exista comercialización alguna de productos. Minimizar este tiempo está claramente relacionado con las distancias que ha de recorrer el trabajador para desplazarse entre localidades, e implica que una reducción en el mismo, no solamente aumentaría las ventas, sino que repercutiría positivamente en el rendimiento del trabajador.

Por otra parte, respecto de los pedidos realizados por entidades como pueden ser hoteles y residencias, el factor que influye directamente es el periodo de tiempo necesario para que los suministros por parte de los proveedores lleguen al almacén central. La empresa colaboradora ha puesto a nuestra disposición un camión con todos los datos necesarios de los productos que comercializa y las rutas que sigue, los datos referentes a las entidades que realizan pedidos directamente a la empresa, los datos de los proveedores de la empresa y una cámara frigorífica donde se almacenarán los productos anteriormente citados, con el fin de no interferir

directamente en el grueso de la organización y en la comercialización de sus productos.

Teniendo en cuenta estos datos, la implementación del sistema multiagente se realizará en los siguientes términos:

1. Se llevará a cabo la implementación de la Comunidad Inteligente Especializada de predicción, que se tendrá la capacidad de predecir la demanda de productos tanto de los camiones itinerantes como de los pedidos realizados directamente a la empresa.

2. Otra Comunidad Inteligente Especializada que realizará el control de stock con el fin de optimizar el proceso de pedidos y controlar las existencias, basándose en los modelos de gestión de inventario.

3. Finalmente se realizará la implementación de una Comunidad Inteligente Especializada que lleve a cabo la optimización de las rutas del camión itinerante.

A través de la arquitectura SCODA estas CIE anteriormente citadas tendrán la capacidad de comunicarse y funcionar como un sistema único que permita optimizar los procesos de la empresa.

Se ha de resaltar que la implementación de las diferentes CIE, se ha realizado con la herramienta *Community Agent Generator*, desarrolla bajo el marco de este trabajo, y propuesta en el capítulo 5 de la misma.

A continuación se introducen los modelos de inventario, las diversas técnicas de predicción y a la optimización de las rutas que se implementarán dentro de las CIE que conforman el sistema multiagente propuesto.

6.2.1. Introducción a los Modelos de Inventario

El coste mantener un cierto número de unidades de un determinado producto en inventario durante un periodo de tiempo determinado, es importante para una empresa del sector de la alimentación. Independientemente que se trate de una empresa de distribución de productos congelados, ya que sus productos son conservados durante largos periodos de tiempo, la imagen que da la empresa comercializando al día los productos que recibe de sus proveedores es muy importante para los clientes. Además, ajustar los pedidos implica minimizar el desembolso de económico de forma que exista una retroalimentación en el negocio. El objetivo de la Teoría de Inventarios es establecer técnicas para minimizar los costes dados para satisfacer una demanda (Zipkin, 2000; Silver *et al.*, 1998; Guide y Srivastava, 1997; Parlarm y Berkin, 1991; Love, 1979; Hadley y Whitin, 1963).

Los costes más frecuentes asociados un inventario son: **Costes de producción,** que están asociados a efectuar una orden para un cierto producto, o bien a producirlo

internamente. **Coste unitario de compra** que corresponde al coste variable unitario involucrado en la compra de artículos a algún proveedor. **Coste de mantenimiento en inventario**, que implica los gastos en los que se incurre al mantener una unidad en inventario un determinado período de tiempo, por lo que este tipo de coste debe ir necesariamente ligado a un intervalo de tiempo, por ejemplo coste anual, semestral o diario. **Costes por escasez o mantenimiento de órdenes pendientes**, el cual se da cuando la demanda de un comprador no puede ser satisfecha, y entonces, se habla de *stockout.* Si el comprador acepta recibir sus artículos fuera de plazo se habla de órdenes pendientes, y si se acepta el hecho de mantener órdenes pendientes, se habla de escasez planificada (Zipkin, 2000).

En la determinación de los modelos de inventario existen dos tipos, en función de la demanda (Zipkin, 2000), tal es así que se proponen los Modelos Determinísticos si la demanda es conocida, y en el caso contrario los Modelos Probabilísticas. En este trabajo se propone la utilización de Modelos Determinísticos ya que una de las CIE que componen el sistema multiagente objeto de estudio, tiene la capacidad de llevar a cabo la predicción de la demanda de los productos.

Dentro de estos Modelos Determinísticos existen cuatro modelos que determinan su utilización en función de de la situación y valores de las variables implicadas en estos modelos. Estas variables y la situación de las mismas se explican a continuación con los cuatro modelos mencionados, así pues, una explicación detallada de cada uno de los modelos y de su formulación matemática podemos encontrarla en (Zipkin, 2000).

Economic Order Quantity Model (EOQ): Para poder emplear este tipo de modelo se requiere que se den una serie de supuestos como son los siguientes: la demanda es conocida y ocurre a tasa constante, las órdenes realizadas tienen un coste asociado, no existen órdenes pendientes y existe un coste de mantenimiento de productos en inventario. (Schwaller, 1988; Khan *et al., 2011*)

Economic Order Quantity Model con Descuentos: En esta situación el coste de un pedido varía en función de la cantidad de la orden de pedido, lo que implica que el coste de mantener unidades en inventario depende del precio al que se ha comprado. Esto implica que si Q es la cantidad de una orden el modelo de descuento es el siguiente: (Cheng, 1991; Roy y Chaudhuri, 2009)

Si $Q<c_1$ *el coste unitario es* p_1
Si $c_1<Q<c_2$ *el coste unitario es* p_2
Si $c_{k-1}<Q<c_{k=\infty}$ *el coste unitario es* p_k

Donde en los puntos $c_1, c_2, \ldots c_k$ existe una variación en el precio, denominándose puntos de quiebre del precio.

Economic Order Quantity Model con Producción (EPQ): Se requiere que la demanda sea conocida y una tasa constante de pedidos, no existirá escasez y los

productos son fabricados a una tasa constante en un periodo de tiempo determinado. (Cheng, 1991; Roy y Chaudhuri, 2009)

Economic Order Quantity Model con Órdenes Pendientes: En situaciones reales la demanda no puede ser satisfecha en una fecha precisa con lo que se da el caso de escasez, el cual se asocia a costes adicionales, como pérdida de ventas. (Cheng, 1991; Roy y Chaudhuri, 2009)

Ahora bien, dentro de estos cuatro modelos determinísticos, la situación planteada en este capítulo, puede asemejarse a los tres casos siguientes: *Economic Order Quantity Model, Economic Order Quantity Model con Descuentos, o Economic Order Quantity Model con Órdenes Pendientes.* Debido a que la empresa no elabora ninguno de los productos que comercializa, el modelo *Economic Order Quantity Model con Producción* es descartado en su aplicación.

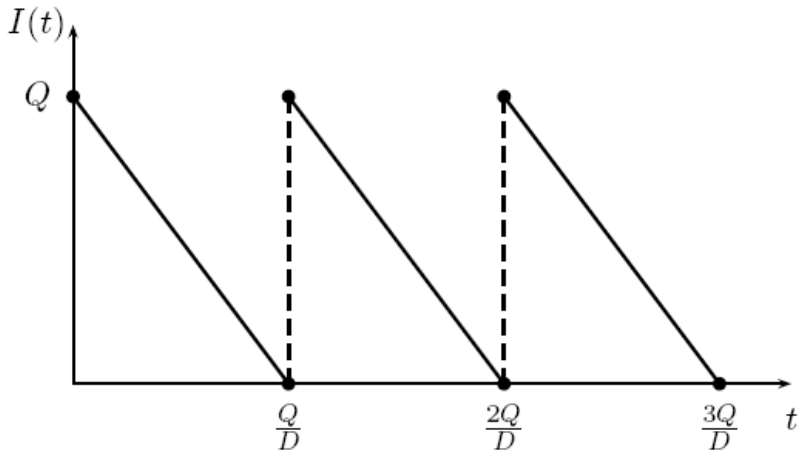


Figura 6.1 Representación de la gráfica correspondiente al modelo EOQ

Dentro de los modelos a tener en cuenta, en nuestro caso, es el *Economic Order Quantity Model (EOQ)* el más común, ya que, las ofertas de productos por comprar una cantidad mayor o los retrasos en la entrega de productos por escasez de los proveedores, no son frecuentes en el modelo de negocio de la empresa colaboradora. Debido a que es el EOQ el modelo más frecuente, en la **Figura 6.1** muestra la representación gráfica de este modelo.

Asumiendo que la primera orden de tamaño **Q** llega exactamente en el instante 0 y considerando que la demanda temporal **D** es conocida (debido a que una de las CIE del sistema realiza la predicción de la misma), el nivel de inventario tomará el valor 0, $\frac{Q}{D}$ veces por periodo de tiempo, en nuestro caso al día. Suponiendo que la demanda ocurre a tasa constante, durante un periodo de longitud **t** (días), se demandarán exactamente **dt** unidades. Luego, el nivel de inventario disminuye linealmente. Cuando el nivel de inventario **I(t)** toma el valor cero, se vuelve a emitir

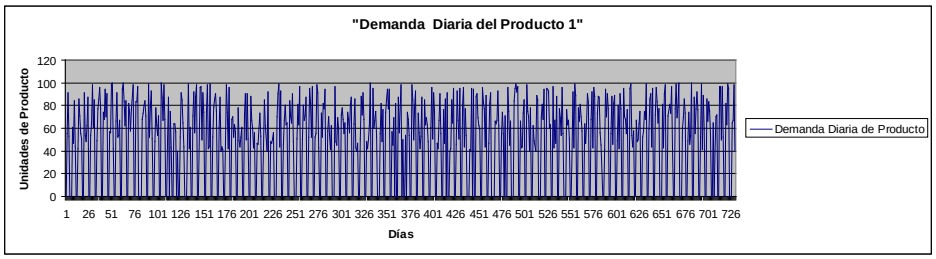
una orden que llega de forma instantánea y comienza un nuevo ciclo. Sin embargo, se ha dicho que los productos solicitados en una orden de pedido tardan un tiempo en ser servidos, lo cual hace que tengamos que considerar una variable de tiempo la cual se ha de tener en cuenta para determinar el punto de reorden, definido como *"nivel de inventario en el cual se ha de emitir una orden de pedido"* (Zipkin, 2000).

Es evidente que la aplicación de estos modelos de inventario al problema planteado en este capítulo rentabilizará y optimizará los recursos empleados hasta ahora para gestionar el almacenaje y comercialización de productos, lo cual generará mayores beneficios. Si bien es cierto, que es necesario conocer la demanda de estos productos de forma estimada la cual ha de acercarse lo máximo posible a la demanda real. Para ello en la siguiente sección se comentarán las técnicas utilizadas para este fin.

6.2.2. Estimación de la Demanda a Través de Técnicas de Predicción

La predicción de valores pertenecientes a series temporales de diversa índole es un problema ampliamente abordado en la literatura (Atiya *et al.*, 1999; Corchado y Lees, 2001; Corchado y Aiken, 2002; Martín-Merino, 2006; Martín-Merino y Román, 2006a; 2006b). Son diversas las técnicas empleadas para predecir series temporales, desde modelos estadísticos tradicionales como son ARIMA y las funciones de transferencia, hasta modelos no lineales basados en redes neuronales artificiales (Velásquez *et al.*, 2010).

La utilización de una u otra técnica tiene que ver con la capacidad que ésta tenga para procesar el tipo de serie temporal seleccionado, así pues, en nuestro caso la serie temporal correspondiente a la demanda de productos congelados, es muy diversa en función del tipo de producto. Si por ejemplo tomamos la serie temporal de los pedidos diarios que se realizan de un producto como puede ser el "*Pulpo*", comprobaremos que la serie tiene una periodicidad en la que sus máximos anuales se alcanzan en Navidad y Semana Santa, sin embargo otros productos no presentan ni estos máximos ni esta periodicidad, por lo que es necesario utilizar alguna técnica que tenga una alta capacidad de generalización de series no lineales. En la **Figura 6.2** se presentan las series de demanda de tres productos diferentes que comercializa la empresa en el período comprendido entre el 01/01/2009 hasta el 30/09/2011.



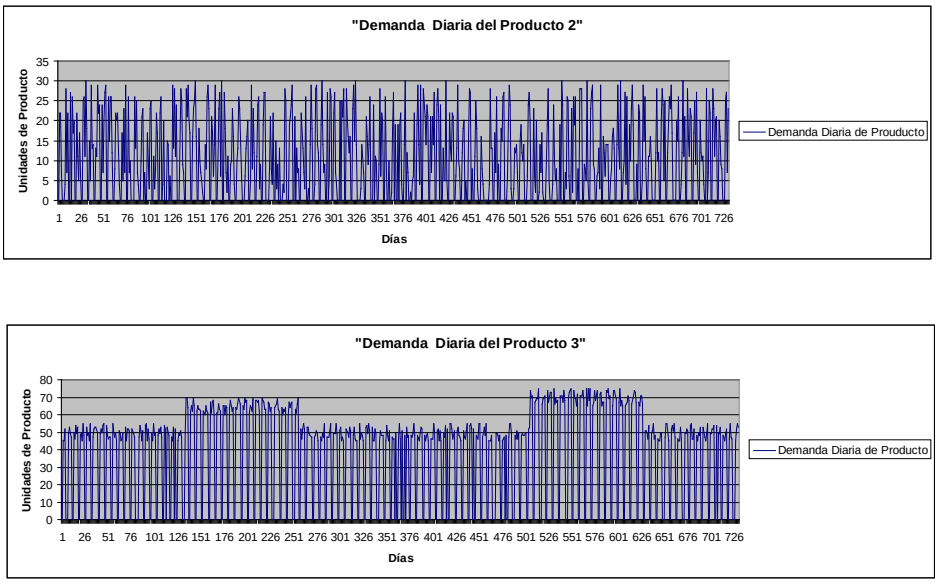


Figura 6.2 Gráfica correspondiente a la demanda diaria de tres productos diferentes

Como se puede observar en la **Figura 6.2** la serie que siguen la demanda de estos tres productos es no lineal. Además, las dos primeras presentan un carácter no periódico y las unidades de producto demandado son completamente diferentes. Se observa en las tres gráficas existen una gran variabilidad de los datos en función del día del año, ya que los fines de semana y días festivos al no trabajarse, la demanda de productos es cero.

En la gráfica correspondiente al *producto 3* se observa una cierta periodicidad que corresponde con los meses de verano, meses en los que la facturación de la empresa asciende ya que las pequeñas localidades se llenan de veraneantes que consumen este producto. Además se observa una variabilidad de los datos muy pequeña, lo que implica que el producto en sí es de consumo diario y común.

A la vista de estas gráficas se puede deducir que existe una gran diferencia entre cada uno de los productos que comercializa la empresa. Además es importante señalar la dificultad que se tendrá a la hora de predecir la demanda de los diferentes productos debido al carácter no lineal y cambiante dichas series, lo que implica que la técnica que seleccionemos ha te tener una alta capacidad de generalización.

Dentro de las técnicas no lineales, el perceptrón multicapa (MLP) ha sido utilizado ampliamente en la literatura para la predicción de de series no lineales, ya que tienen la capacidad de aproximar cualquier función continua definida en un dominio compacto, sin embargo, su especificación se basa fundamentalmente en criterios heurísticos y juicio experto del modelador, de tal forma, que dicho proceso se basa en un conjunto de pasos críticos que afectan al resultado final del modelo, en

términos de su ajuste a los datos históricos y su capacidad de generalización con el fin de realizar predicciones (Velásquez *et al.,* 2010).

Por otra parte, las Máquinas de Vectores Soporte *(SVM, Support Vector Machines)* (Vapnik, 1995; Vapnik *et al.,* 1996) son un tipo de redes neuronales diseñadas en sus orígenes para llevar a cabo la resolución de problemas de clasificación no lineales (Burges, 1998; Belousov *et al.,* 2002). En los últimos años las máquinas de vectores soporte se han aplicado a problemas de regresión y predicción de series temporales (Vapnik, 1995; Vapnik *et al.,* 1996) debido a su gran capacidad de generalización (Vapnik, 1998; Schölkopf y Smola, 2002), la cual se debe a su estructura y de la metodología utilizada para la estimación de sus parámetros.

En (Velásquez *et al.,* 2010) se detallan una serie de ventajas de las SVM sobre otras técnicas no lineales, entre las que se incluye el perceptrón multicapa:

- Su especificación se basa en el principio de minimización del riesgo estructural que equivale a minimizar el límite superior del error de generalización del modelo. Así, la estimación de una SVM busca encontrar tanto la estructura óptima del modelo como los valores óptimos de sus parámetros, permitiendo una mayor capacidad de generalización del modelo. La estimación de los parámetros de otros modelos tradicionales, tales como los MLP, se basa en la aplicación del principio de minimización de riesgo empírico que depende fundamentalmente del ajuste a los datos históricos, tomando la estructura del modelo como un parámetro de entrada.

- La estimación de los parámetros de una SVM es equivalente a la solución de un modelo de programación cuadrática con restricciones lineales; ello implica, que la solución óptima es global y única, lo cual representa una clara ventaja sobre modelos como los MLP que se caracterizan por poseer múltiples mínimos locales.

Sin embargo, hemos de tener en cuenta que hay que realizar el ajuste de una serie de parámetros para abordar cada problema en particular. Esto implica que se convierte en una técnica con un alto coste computacional, ya que el ajuste de los parámetros se realiza por lo general a través de validación cruzada, donde adicionalmente, hay que escoger los criterios de esta misma (Martín-Merino, 2006; Velásquez *et al.,* 2010).

La formulación del modelo matemático, así como, el ajuste de los parámetros de las SVM no son objeto de esta tesis, si bien es cierto que la CIE de predicción se implementará a partir de Support Vector Machines, ya que existe multitud de bibliografía que muestra cómo hacerlo (Stefan, 2000; Mangasarian y Musicant,2001; Collobert y Bengio, 2001; Hsu y Lin, 2002; Chakrabartty, 2003; Hernández y

Salazar, 2006; Hsu *et al.,* 2010; Chang y Lin, 2011), lo cual simplificará la labor al implantar la CIE y permitirá comprobar su funcionamiento.

No obstante, en (Vapnik, 1995; Vapnik *et al.,* 1996; Martín-Merino, 2006; Velásquez *et al.,* 2010) se da una descripción detallada de las SVM, tanto de su formulación matemática como de las posibles aplicaciones que pueden tener. En nuestro caso se adoptará un enfoque de regresión para llevar a cabo las predicciones de los diferentes productos y con ello comprobar el funcionamiento del sistema multiagente y su efectividad, siguiendo las líneas propuestas en la bibliografía anteriormente citada a la hora de su implementación y de estimar sus parámetros.

6.2.3. Optimización de Rutas

El problema de la optimización de rutas es ampliamente abordado en la literatura (Golden *et al.,* 1977; Axhausen y Smith, 1984; Baaj y Mahmassani, 1991; Laporte, 1992; Díaz *et al.,* 1996; Shih *et al.,* 1998; Biswas *et al.,* 2005; Lüer *et al.,* 2009). Este tipo de problemas son una de las partes más importantes en la logística del transporte, ya que se busca la optimización de una serie de rutas a realizar por una flota de vehículos que usualmente deben satisfacer las demandas de mercancías de clientes geográficamente dispersos. En función de una serie de requisitos iniciales estos problemas se corresponden con grafos formados por vértices y aristas que los conectan entre los cuales pueden ser definidas un conjunto de condiciones, que dan lugar a la resolución de este problema de formas diferentes.

Debido a la metodología de trabajo empleada por los camiones itinerantes que comercializan los productos congelados a los clientes directamente, la cantidad de kilómetros recorridos diariamente es muy importante. Este kilometraje se traduce en un gasto de combustible que repercute directamente en los beneficios de la empresa, que en función de los datos suministrados, cada uno de los camiones de venta directa recorre diariamente hasta 20 localidades, entre las cuales varía mucho la distancia existente, y la forma de recorrerla en función de qué localidades son visitadas primero.

En primer lugar son analizados una serie de problemas clásicos de la literatura como son el Problema del Cartero Chino *(Chinese Postman Problem, CPP)* propuesto por el matemático chino Kwan en 1962 y resuelto por Edmons y Johnson en 1973, el cual consiste a grandes rasgos en recorrer todas las aristas de un grafo con el mínimo coste posible (Kwan, 1962; Edmonds y Johnson, 1973; Minieka, 1979; DeArmon, 1981; Assad *et al.,* 1987; Eiselt *et al.,* 1995a; Pearn y Chou, 1999; Ghiani e Improta, 2000). El Problema del Cartero Rural *(Rural Postman Problem, RPP),* que consiste en una generalización del CPP de forma que solamente un subconjunto de aristas han de ser necesariamente recorridas (Eiselt *et al.,* 1995b; Hertz *et al.,* 1999; Benavent, *et al.,* 2005; Benavent *et al.,* 2007), y el Problema del Viajante de Comercio *(Travelling Salesman Problem, TSP)* que consiste en minimizar el coste al recorrer un grafo iniciando el recorrido y finalizándolo en el mismo vértice, pasando solamente una vez por cada vértice (Miller *et al.,* 1960;

Pérez-Delgado, 2001; Toulouse y Wolfer, 2009; Petersen *et al.*, 2010; Lusby *et al.*, 2010).

Una vez analizados estos problemas clásicos de la literatura vemos que existen similitudes con el problema real que se nos plantea pero que sin embargo no se corresponde ninguna de estas situaciones con la situación real:

"Un vendedor ha de partir de Zamora y recorrer una serie de localidades con el menor coste posible. Estas localidades se pueden visitar una sola vez, sin embargo, se pueden atravesar todas las veces que sea necesario siempre y cuando el coste del recorrido sea el menor"

Para solventar el problema se ha propuesto un sencillo algoritmo que se mostrará en el apartado 6.3.4., mediante el cual se minimiza el coste de la ruta que cada vendedor ha de realizar en los términos anteriormente expuestos.

6.3. Desarrollo del Sistema

Uno de los fines perseguidos en este trabajo es que el desarrollo de sistemas multiagente pueda ser realizado de forma rápida y eficaz. A través de la arquitectura SCODA podemos desarrollar Comunidades Inteligentes Especializadas de forma independiente que puedan comunicarse y trabajar de forma conjunta bajo un mismo marco de acción.

La aplicación práctica para la resolución de los diferentes problemas logísticos que se presentan en este caso de estudio es llevada a cabo mediante la implementación de la arquitectura SCODA, que consta de tres Comunidades Inteligentes Especializadas, desarrolladas de forma independiente, siguiendo la metodología de desarrollo propuesta en el capítulo 5 de esta tesis y utilizando el software *Community Agent Generator* propuesto en ese mismo capítulo.

La CIE de Predicción tiene la capacidad de realizar las predicciones de la demanda de productos, utilizando SVM implementadas como un servicio externo a la propia CIE, siguiendo así los principios en los que se basa la arquitectura SCODA. En cuanto a la CIE de Inventarios, es capaz de realizar los cálculos necesarios para gestionar el stock de productos llevando a cabo una adaptación de la teoría de inventarios en todas sus variantes en las que la demanda es determinística. Finalmente, la CIE responsable de optimizar las rutas, generará un itinerario que minimice el coste asociado de los camiones y maximicen los beneficios de la empresa.

6.3.1. Comunidad Inteligente Especializada de Predicción

Como se ha comentado en la sección anterior cada CIE se desarrolla, utilizando el software *Community Agent Generator*, de forma independiente. La

CIE de predicción tiene la labor de realizar la predicción de la demanda diaria de productos. Para ello se ha hecho uso de la librería *LibSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)* a través de la cual se ha podido implementar un modelo de regresión basado en SVM, capaz de realizar dichas predicciones.

Para realizar el entrenamiento de la SVM se ha dispuesto de los datos relativos a la demanda de una serie de productos comercializados tanto al por menor, como al por mayor. Además estos productos son de diferente procedencia, de forma que pueda evaluarse la influencia temporal en la demanda de los mismos. Los datos proporcionados por la empresa corresponden a un periodo de tiempo comprendido entre el *01/01/2009 y el 30/09/2011*, y su tipificación ha sido tomada de (Martin-Merino, 2006).

La elección de los parámetros para el entrenamiento se ha realizado a través de validación cruzada (Martín-Merino, 2006; Velásquez *et al.*, 2010) lo cual ha permitido generar diferentes modelos, dado que cada producto evaluado cuenta con una serie de demanda temporal diferente. Si bien es cierto que existen productos con series temporales similares, lo cual hace que se hayan agrupado para emplear con ellos un mismo modelo.

Una vez obtenidos los modelos para realizar las predicciones diarias de cada producto, el sistema de predicción se ha implementado como una serie de servicios de la CIE de predicción los cuales son desplegados en un servidor diferente, que siguiendo la filosofía de SCODA permitirá un modelo distribuido descargando computacionalmente el servidor donde se alojan las CIE, como se representa en la **Figura 6.3.**

.

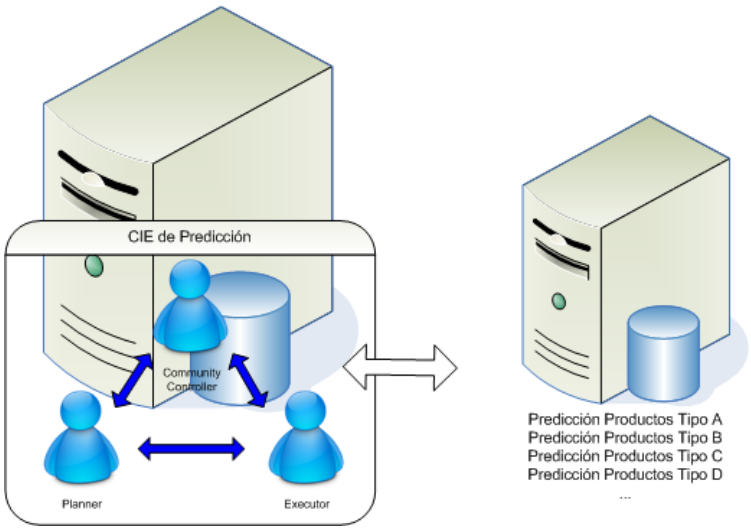


Figura 6.3 Comunidad Inteligente Especializada de Predicción

Para la evaluación de estos modelos de predicción, se ha decidido utilizar tres medidas de error que son complementarias y nos permiten tener una visión de las predicciones realizadas y su comparación con la realidad.

1. **Error Absoluto Medio en Porcentaje (Mean Absolute Porcentaje Error, MAPE),** es la media absoluta entre el pronóstico y los valores observados expresado como porcentaje de los valores observados y se define como:

$$MAPE = 100 \frac{\sum_{i=1}^{n} \left| \frac{L(i) - \hat{L}(i)}{L(i)} \right|}{n} \tag{8}$$

Donde *L(i)* es la demanda de producto media, $\hat{L}(i)$ es la demanda estimada y *n* el tamaño de la muestra de datos.

2. **Error Cuadrático Medio (Root Mean Square Error, RMSE):** Los errores grandes son de gran importancia ya que si ese error es por defecto se induce una pérdida de venta de mercancía puntual muy importante, por ello se incluye este error ya que la función cuadrática da más peso a los errores grandes. Este error se define como:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \left(\frac{L(i) - \hat{L}(i)}{L(i)} \right)^2}{n}} \tag{9}$$

3. **Error Máximo (Maximum Error, ME):** Esta medida complementa a las anteriores y evalúa la máxima desviación entre el valor predicho y el valor real. A través del mismo podemos evaluar que tipo de producto es que mayor error arroja en una estimación dada. Se define matemáticamente como:

$$ME = \max_{i} \left| L(i) - \hat{L}(i) \right| \qquad i = 1, ...,n \tag{10}$$

Ya en producción y de forma individual, las predicciones que realiza la CIE obtienen unos resultados que varían de forma significativa en función del producto. Como se había comentado anteriormente, era de esperar ya que existen productos cuya serie temporal no sigue ningún patrón y su predicción es muy costosa. Los resultados obtenidos se presentan a continuación en la **Tabla 6.1.**, y corresponden a los test realizados para el mes de septiembre de 2011.

Tabla 6.2 Errores de predicción de la demanda diaria de productos

1. La tipología hace referencia a conjunto de productos con una serie temporal de demanda similar.

Producto	Tipología[1]	MAPE	ME	RMSE
Producto 1	A	17,15%	33,46	14,60
Producto 2	C	4,23%	12,51	4,26
Producto 3	C	3,16%	11,39	3,78
Producto 4	A	17,12%	31,91	14,86
Producto 5	B	81,51%	15,74	6,81
Producto 6	A	16,24%	35,91	13,99
Producto 7	B	29,79%	21,32	10,65
Producto 8	B	37,23%	12,11	5,94
Producto 9	B	32,39%	21,53	12,52
Producto 10	C	3,94%	12,52	4,30
Producto 11	A	16,67%	34,03	14,39

Teniendo en cuenta lo anterior, se han agrupado en varias tipologías los productos que presentan características similares en sus series de demanda. Así pues se han realizado los siguientes grupos:

A: *Productos con una demanda diaria de hasta 100 unidades y que su serie no presenta periodicidad.*

B: *Productos con una demanda diaria de hasta 30 unidades y que su serie no presenta periodicidad.*

C: *Productos con una demanda diaria de hasta 80 unidades y que su serie presenta periodicidad estacional.*

A la vista de los resultados mostrados en la **Tabla 6.2.** podemos observar que las series de productos que presentan un periodicidad estacional, es decir que son consumidos de forma masiva en determinadas épocas del año, son estimados con un error MAPE mucho más bajo que las otras. Además dentro de las series que no presentan estacionalidad el porcentaje de error MAPE aumenta de forma significativa cuando la demanda de productos oscila entre números más bajos. A continuación se muestran tres gráficas correspondientes a uno de los productos por tipología, donde se puede observar las diferentes curvas de demanda y de predicción y los errores cometidos. A través de estas gráficas podemos observar de una forma más cómoda los errores máximos de predicción, que si nos fijamos en la **Figura 6.5.** vemos que son más acentuados debido a que las cantidades con las que tratamos en esta serie son menores que en los otros dos casos.

Son los picos de las series los que los modelos de predicción implementados no son capaces de detectar con exactitud, ya que los días que no se trabaja, y por lo tanto no existe demanda de productos, son detectados perfectamente por el modelo. La detección de estos picos es una tarea interesante sobre la cual se trabajará, ya que se pretende desarrollar un repositorio de CIE en un futuro y por lo tanto se realizarán pruebas con otros parámetros e incluso con otro tipo de técnicas de predicción.

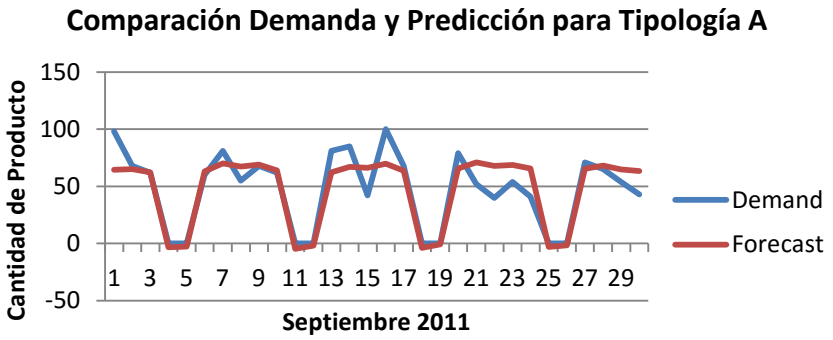


Figura 6.4 Comparación entre demanda y predicción para productos de tipología A

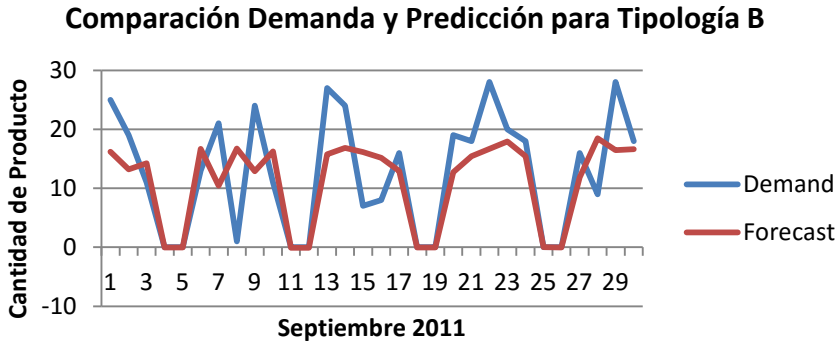


Figura 6.5 Comparación entre demanda y predicción para productos de tipología B

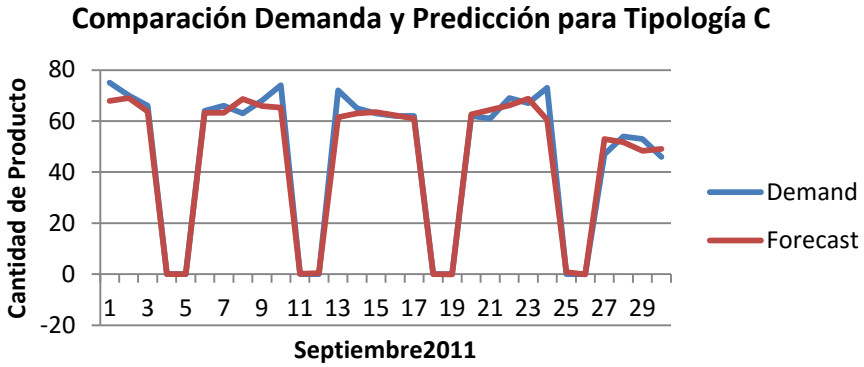


Figura 6.6 Comparación entre demanda y predicción para productos de tipología C

Una vez comentados los resultados obtenidos con los modelos de predicción resaltar, como se muestra en la **Figura 6.7.,** es la propia CIE, a través del *PlannerAgent*, el que se encarga de seleccionar el modelo de predicción en función de la tipología del producto lo cual agiliza la elección del servicio necesario de predicción, para que finalmente el *ExecutorAgent*, con los datos proporcionados por el *PlannerAgent*, haga efectiva la petición del servicio correspondiente.

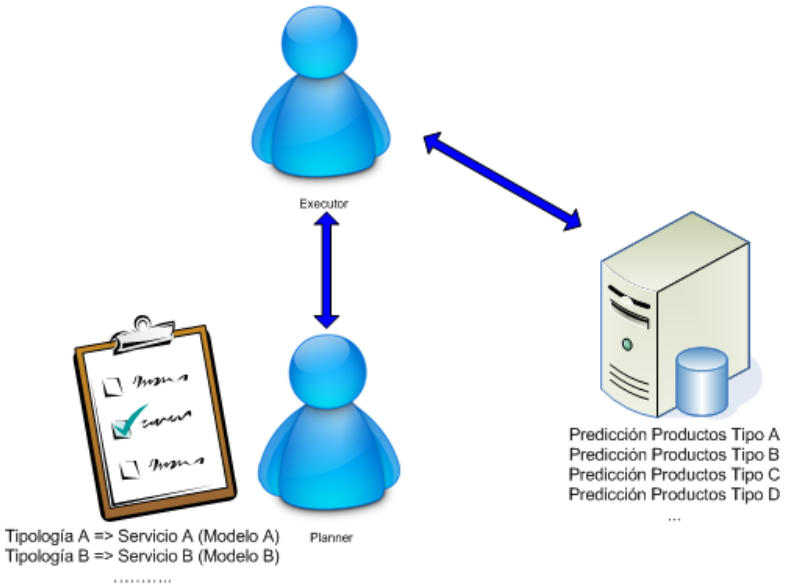


Figura 6.7 Selección del modelo de predicción adecuado por el PlannerAgent

6.3.2. Comunidad Inteligente Especializada para la Gestión de Inventarios

Esta comunidad va a permitir controlar las existencias que existen en la cámara, de forma que cuando detecta que un producto no tiene el stock suficiente genera una orden de pedido que se enviará por mail al responsable de la cámara. Esta orden es orientativa ya que es el responsable quien la confirma o la modifica. Esta CIE por si sola tiene capacidad para acceder a una base de datos, implementada para el presente desarrollo, donde se almacenan el stock de los productos, los descuentos que los proveedores hacen en función de la cantidad de producto solicitado, y el tiempo que tarda el producto en llegar.

Así pues, esta CIE por sí sola no tiene capacidad para estimar los datos de la demanda futura, y hasta que no se modifique la base de datos de existencias no podrá actuar. Si bien es cierto, que en el apartado 6.3.4. se especificará el funcionamiento del sistema completo, de forma que sean las CIE las que trabajen de

forma autónoma y coordinada para llevar a cabo el desempeño global del sistema, lo cual es uno de los fines pretendidos con la arquitectura SCODA.

Dentro de los modelos de inventario vistos en el apartado 6.2.1., en nuestro caso particular, y como se había comentado en la sección anterior, la CIE responsable de gestionar los inventarios hará uso de tres de estos modelos para realizar las órdenes de pedidos: *Economic Order Quantity Model, Economic Order Quantity Model con Descuentos y Economic Order Quantity Model con Órdenes Pendientes*, los cuales serán implementados como servicios de la CIE de inventarios, y podrán ser accedidos por la misma.

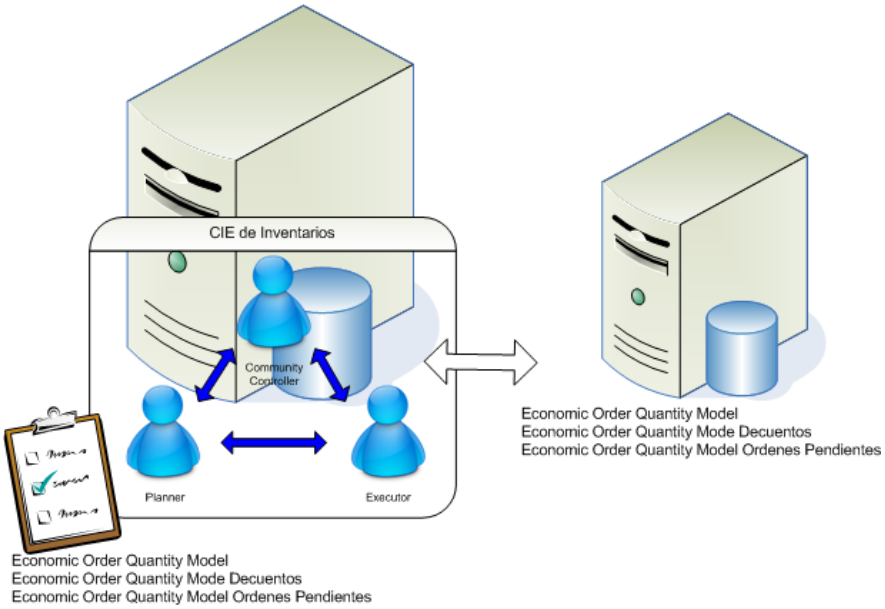


Figura 6.8 Representación de la Comunidad Inteligente Especializada de inventarios

En la **Figura 6.8.** se muestra un esquema de la CIE de inventarios. El *PlannerAgent* recibe la orden, de su *CommunityControllerAgent*, de ejecutar un análisis del inventario. A continuación accede a la base de datos donde realiza una comprobación del stock de cada producto, accede a los datos de los proveedores donde se encuentra información sobre los descuentos y el tiempo que estos proveedores tardan en servir los productos, y en teniendo en cuenta estos datos, determina de forma dinámica el punto de reorden de cada uno de ellos en función de la demanda de los mismos, para finalmente poder seleccionar el tipo de modelo de inventario más acorde y generar una orden de pedido. Una vez obtenidos los parámetros necesarios respecto al tipo de modelo, es el *ExecutorAgent* el que a través de los servicios de la CIE hace efectiva la petición.

A continuación se expone un ejemplo de dos productos con características de demanda diferentes, sobre los que la CIE responsable de inventario trabajará. Para ello se han de considerar la demanda total semanal del producto (D), los gastos de envío (C_O), los gastos de ordenar el pedido, los cuales se calculan a través del tiempo empleado y del salario por hora del trabajador responsable (C_H), y el tiempo que tarda en llegar el pedido a la empresa (L_D).

Producto A: D (445 unidades); C_O (0 €); C_H ($\frac{6(€) \times 5(días)}{445(unidades)}$); L_D (3 días)

Producto B: D (169 unidades); C_O (0 €); C_H ($\frac{3(€) \times 5(días)}{169(unidades)}$); L_D (2 días)

Tabla 6.3 Resultados de aplicación del modelo EOQ para los productos A y B

Producto	Cantidad de pedido óptima	Punto de Reorden
Producto A	115,25	190,46
Producto B	100,16	127,27

A la vista de los resultados mostrados en la **Tabla 6.3,** y teniendo en cuenta las características de cada uno de los productos, observamos que para el *Producto A* el punto de reorden, es decir, la cantidad de producto que ha de existir para que no haya escasez donde se realiza una orden de pedido es de 190, 46 y su orden de pedido óptima es de 115, 25. Es decir, cuando las existencias del *Producto A* lleguen a 191 unidades nuestro sistema ha de emitir una orden de pedido de 116 unidades. Lo mismo que ocurre con el *Producto B*, de esta forma nos aseguraríamos de que no existe escasez de existencias.

La importancia que tiene el determinar la orden óptima de pedido de un producto viene dada entre otros factores por la calidad de los mismos en cuanto a la fecha de caducidad, y además por el dinero a desembolsar por la empresa, ya que si los pedidos son escalonados y de menor magnitud (siempre y cuando no existan descuentos por cantidad) la empresa tiene mayor capacidad de recuperación al existir ingresos de venta diaria, lo cual dota de un margen importante a la hora de hacer frente a los pagos.

6.3.3. Comunidad Inteligente Especializada en la Optimización de rutas

Esta CIE no estaba prevista dentro del sistema global a implementar, sin embargo, se ha incluido para mostrar la eficiencia de la arquitectura, su versatilidad y su capacidad para aumentar en tamaño y funcionalidad. De esta forma se podría comprobar si variando el recorrido a lo largo de las localidades a visitar puede existir un ahorro en la distancia y así, un ahorro de combustible y de horas de trabajo.

Para ello se incluye dentro del sistema una CIE que gestione la optimización de las rutas, de la forma propuesta en el apartado 6.2.3. Una vez analizadas las posibilidades para conseguir los fines propuestos se diseña un sencillo algoritmo que recorra todas las localidades de cada ruta con el menor coste, siendo los datos necesarios que necesita la CIE responsable de optimizar las rutas las distancias existentes entre cada una de las localidades, las cuales han sido obtenidas a partir de ***Google Maps***. El algoritmo propuesto se muestra a continuación:

inicio
cargar matriz *(distancias)*
distancia.total=0
ruta=null
almacenar.ruta *(punto de partida, punto de regreso)*
para cada *(localidad no visitada)*
buscar *(localidad más cercana)*
almacenar.ruta *(localidad visitada)*
distancia.total=distancia.total+distancia.actual
si (todas localidades visitadas)
distancia.total=distancia.total+distancia.vertice.inicial
fin para
devolver (ruta+distancia.total)
fin

Algoritmo 6.1. Algoritmo para optimización de rutas (Elaboración Propia)

Una vez implementado el algoritmo como servicio de la CIE de optimización de rutas, se despliega y se comprueba su funcionamiento sobre las rutas actuales. Se ha de resaltar que esta CIE tiene la capacidad de acceso a la base de datos donde se almacenan las localidades a visitar y las rutas actuales, de forma que pueda reorganizar las rutas siempre que exista un cambio en las mismas.

Como se muestra en la **Figura 6.9.,** es el *PlannerAgent*, una vez recibida la orden de su *CommunityControllerAgent*, accede a la base de datos donde se almacena las diferentes rutas y distancias entre localidades. El *PlannerAgent* envía al *ExecutorAgent* los diferentes parámetros necesarios para ejecutar el servicio de optimización de rutas para que se haga efectiva la petición, y responde al *PlannerAgent* con el resultado obtenido. En caso de que una de las rutas haya cambiado o que uno de los recorridos se realice de forma que exista menos distancia, es el *PlannerAgent* quien modifica la base de datos introduciendo la nueva ruta.

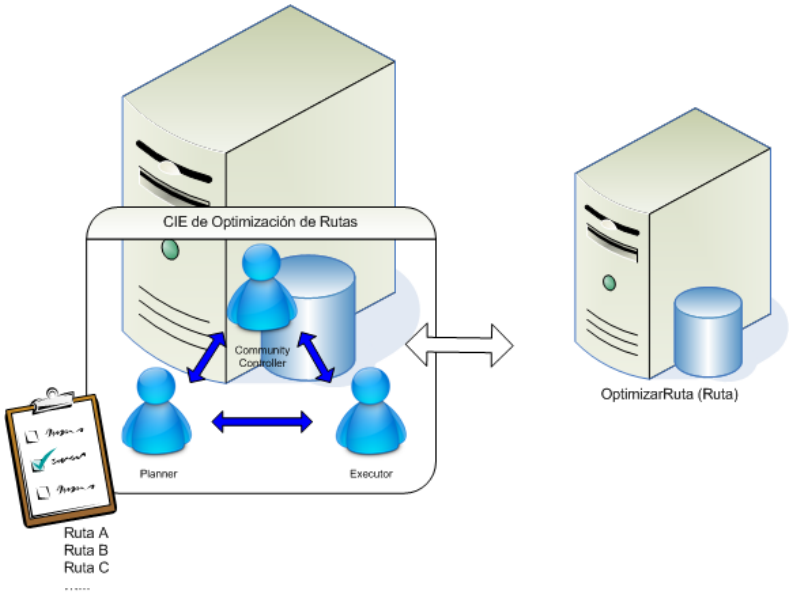


Figura 6.9 Representación de la Comunidad Inteligente Especializada de optimización de rutas

Para finalizar esta sección, en la **Tabla 6.4.** se muestran los resultados obtenidos al aplicar la optimización de rutas a través de la CIE responsable de ello. Esta optimización se ha aplicado al camión que la empresa ha cedido para nuestro estudio, el cual realiza cinco rutas diferentes a la semana, visitando localidades a lo largo de las provincias de Salamanca, Valladolid y Zamora, en el momento de aplicación de la optimización.

Tabla 6.4 Resultados de aplicación de la optimización de rutas para cinco rutas diferentes

Ruta	Km Actuales	Km Optimizados	Mejora en Km
Ruta A	390,4	383,9	6,5
Ruta B	329,8	283,7	46,1
Ruta C	147,9	113,8	34,1
Ruta D	89,8	89,8	0
Ruta E	356,1	279,2	76,9

A la vista de los resultados que arroja la **Tabla 6.4.** se puede deducir que la optimización de rutas es necesaria para los diferentes vehículos de la empresa. Solamente para el vehículo analizado, en tres de sus rutas existen diferencias muy importantes en cuanto a la distancia recorrida entre la ruta actual y la optimizada. Teniendo en cuenta que estas rutas se realizan cuatro o cinco veces al mes, la cantidad de kilómetros recorridos es muy importante lo cual implica un ahorro de combustible y tiempo que mejoraría los beneficios de la empresa, teniendo en cuenta una flota entre 10 y 15 vehículos.

6.3.4. Funcionamiento del Sistema Implementado bajo SCODA

Una vez visto el cometido de cada una de las Comunidades Inteligentes Especializadas y detalladas cuáles son sus funciones, en esta sección se procede a explicar el funcionamiento del sistema completo. Se explicará la relación y coordinación de cada CIE así como los pasos que siguen para lograr sus cometidos bajo la arquitectura SCODA.

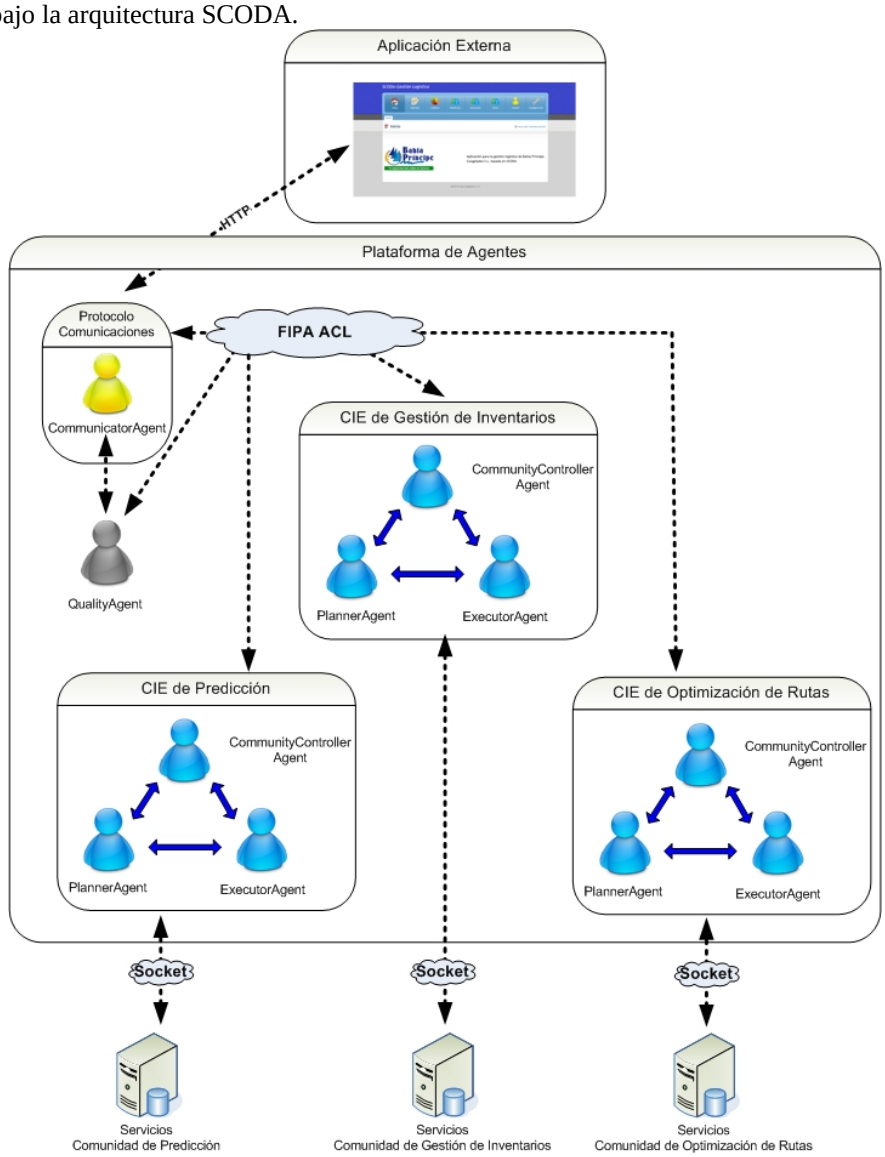


Figura 6.10 Representación de la Arquitectura SCODA para el caso de estudio

En la **Figura 6.10.** se representa la arquitectura SCODA desplegada para el caso de estudio. Su proceso de comunicación se describe a continuación y se representa en la **Figura 6.12**:

1. Una aplicación externa tiene la capacidad de acceder a al sistema de forma que desde la misma se pueden visualizar los ficheros de control generados por el *QualityAgent* los cuales muestran el funcionamiento del sistema en tiempo real. Además se permite visualizar el funcionamiento de cada una de las CIE y comprobar la información manejada por las mismas. También se permite solicitar los servicios que las CIE ofrecen de forma independiente ya que puede surgir la necesidad de realizar alguna operación sobre las mismas externa a su ciclo de funcionamiento. En la **Figura 6.11.** se muestra un detalle de acceso a uno de los registros de la base de datos. En este caso se muestra la predicción de demanda de un determinado producto a lo largo de una semana.

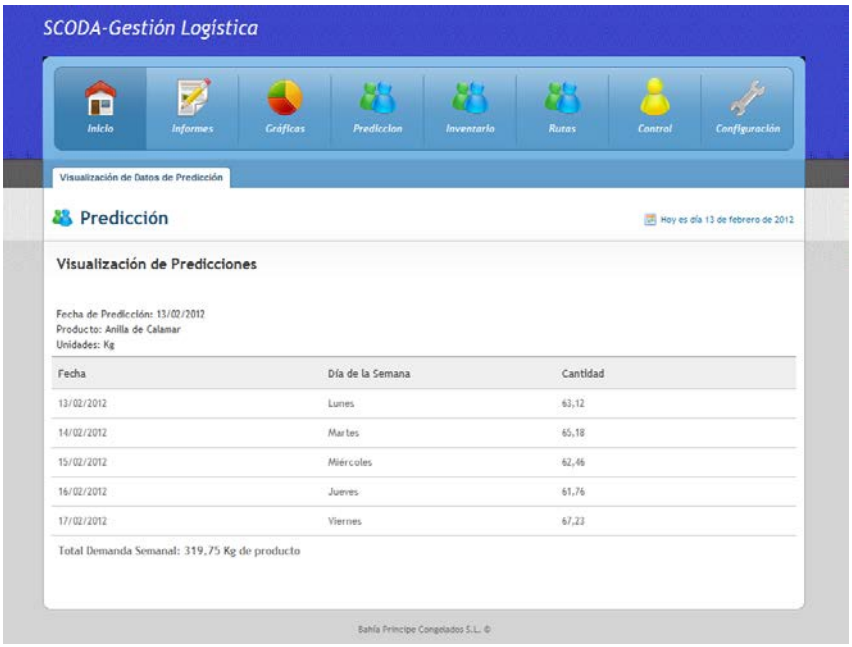


Figura 6.11 Visualización de la predicción de demanda de un determinado producto

2. A través de la aplicación externa, todos los lunes a las 6.00 a.m. se ejecuta una petición de predicción de la demanda que existirá a lo largo de la semana. Se ha de resaltar que a la vista de los resultados obtenidos en el apartado 6.3.1. respecto de la predicción de demanda de los diferentes productos, se decide que el funcionamiento del sistema se realice sobre los productos con *Tipología C* ya que los errores de predicción obtenidos sobre este tipo de productos son aceptables para probar el sistema. Esta petición

es recogida por el *CommunicatorAgent*, que selecciona la CIE correspondiente y solicita una ejecución de los servicios de predicción al *CommunityControllerAgent* correspondiente. Una vez ejecutada la petición con los resultados y se devuelven los resultados obtenidos al *CommunicatorAgent* que, de forma automática, envía una petición a la CIE de gestión de inventarios, para obtener la hoja de ruta de pedidos en la presente semana.

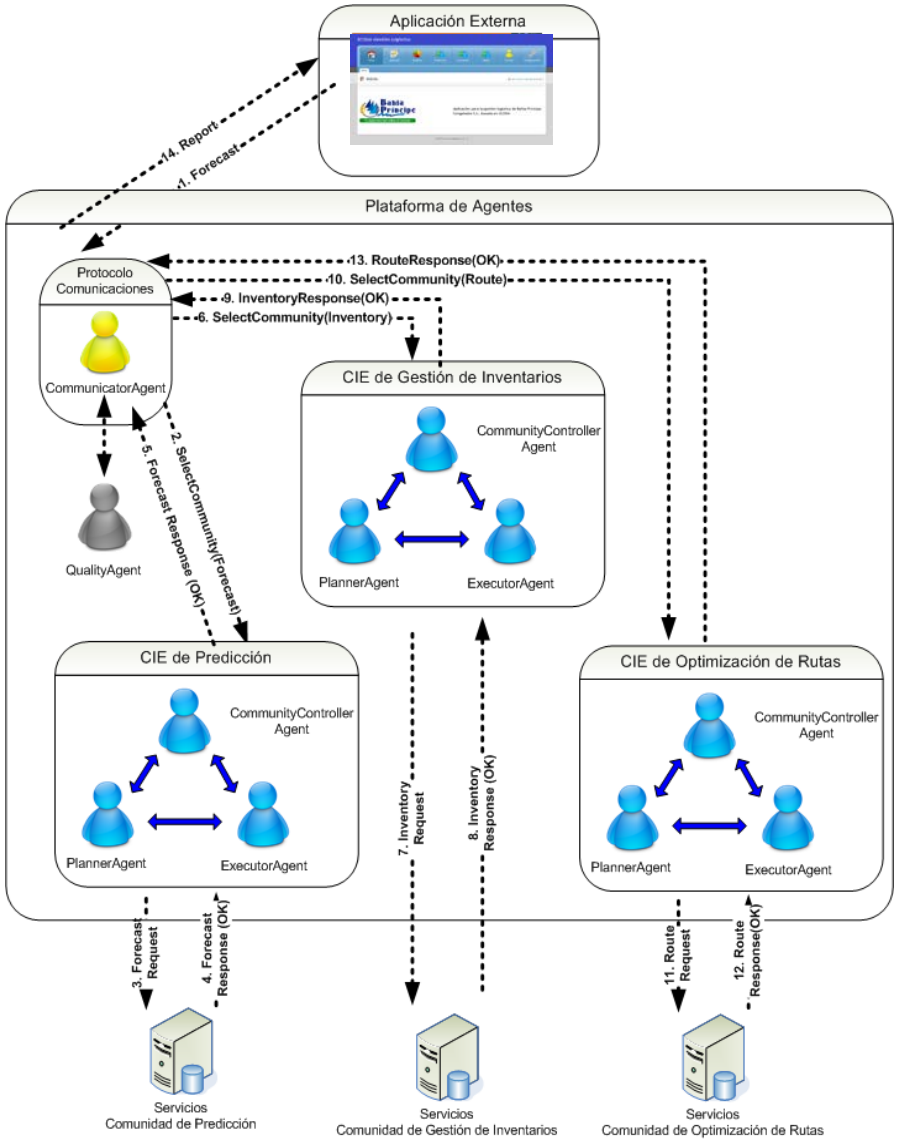


Figura 6.12 Funcionamiento del sistema ante una solicitud de predicción de demanda

3. Una vez recibida la petición de ejecución del servicio de inventarios el *CommunityControllerAgent* de la CIE de inventarios transmite dicha petición al equipo de trabajo de esta CIE, donde el *PlannerAgent* accede a la base de datos de gestión de inventarios y recaba los datos necesarios para la ejecución del servicio. Cuando se obtienen los resultados de ejecución del servicio de inventarios se envían al *CommunicatorAgent*, y a su vez se genera una entrada en base de datos con la orden óptima de pedido, el punto de reorden y el periodo de tiempo (la semana en curso) donde se hará efectivo la aplicación del modelo de inventario. El *CommunicatorAgent* genera un informe con los datos para realizar los pedidos y lo envía por e-mail al responsable de la cámara.

Una vez que se conocen los datos correspondientes a la orden óptima de pedido y al punto de reorden, los cuales han sido almacenados en la base de datos de inventario, y viendo la necesidad de advertir al responsable de la cámara cuando llegue este punto de reorden, se decide implementar otro servicio para la CIE de gestión de inventarios el cual simplemente compruebe el stock de los productos y una vez alcanzado el punto de reorden lo comunique a través de e-mail al responsable de la cámara. La ejecución de este servicio se realiza de forma interna en la CIE de gestión de inventarios, ejecutándose cada hora.

4. Viendo los resultados mostrados en el apartado 6.3.4. sobre la optimización de rutas, por parte de la empresa se propone la posibilidad de realizar esta optimización en todos sus vehículos, para lo cual se dota de acceso a la base de datos general de rutas y vehículos.

Viendo el interés mostrado en este tipo de servicio se decide llevar a cabo, desde la aplicación externa a SCODA, una comprobación semanal de la modificación de la base de datos de rutas con el fin de solicitar a la plataforma la reorganización de las que hayan sido modificadas. Si bien, es cierto que la modificación de las rutas cambia en función de la época del año en que nos encontremos, y de esta forma siempre se puede contar con la ruta óptima para cada uno de los vehículos. Una vez recibida la petición, el *CommunicatorAgent* solicita la ejecución del servicio de optimización seleccionando previamente la CIE responsable del mismo. El *CommunityControllerAgent* ordena la ejecución del servicio y el *PlannerAgent* accede a la base de datos general de rutas para obtener los parámetros necesarios para su ejecución, los cuales son enviados al *ExecutorAgent* que realiza la ejecución del servicio y comunica los resultados la *PlannerAgent*. Este agente modifica la base de datos general de rutas y desde la CIE se envían los resultados al *CommunicatorAgent* que genera un informe y lo envía a la aplicación externa. En la **Figura 6.13.** se muestra el detalle del acceso a una ruta concreta realizada por un vehículo.

5. En los términos establecidos en el capítulo 5 de este trabajo, el *QualityAgent* se encuentra en constante comunicación con el

CommunicatorAgent y con cada uno de los *CommunityControllerAgent* para comprobar su funcionamiento y llevar a cabo acciones correctivas en el caso de que fuesen necesarias.

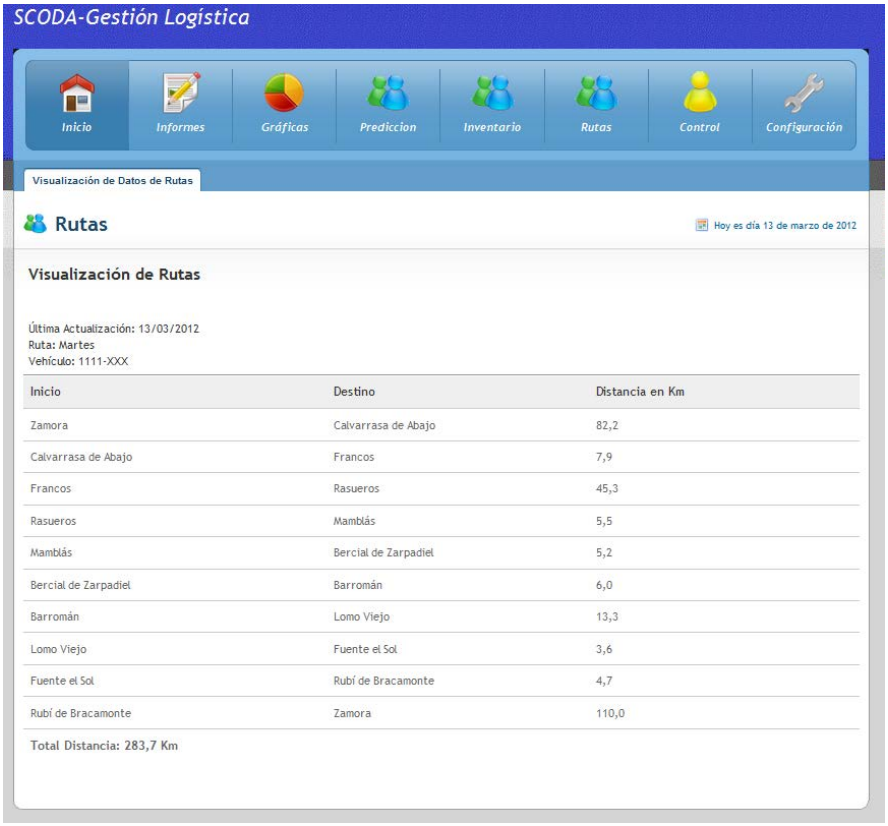


Figura 6.13 Detalle de una ruta concreta optimizada por la CIE de optimización de rutas

6.4. Comparativa con otros Sistemas

En este punto se presenta una comparativa de SCODA con un sistema multiagente desarrollado para el caso. Para ello se ha decidido implementar un sistema multiagente (MAS) que resuelva el problema dado. Esta comparativa se realizará en función de una serie de características que se refieren al funcionamiento en sí de la arquitectura independientemente de la plataforma de implementación. El MAS implementado constará de un *CoordinatorAgent* que coordina a los demás agentes, y de un agente por cada servicio a implementar, *ForecastAgent, InventoryAgent y RouteAgent.* El MAS se implementa sobre JADEX en la misma máquina donde está implementado el sistema con SCODA. La **Tabla 6.5** muestra las diferencias a nivel estructural entre las arquitecturas seleccionadas para la resolución del problema.

Tabla 6.5 Comparativa de SCODA con un Sistema Multiagente

Características	MAS	SCODA
Tipo Arquitectura	• Basada en agentes • Centralizada	• Basada en agentes • Distribuida
Plataforma	• JADEX	• JADEX
Protocolos de Comunicación	• ACL • HTTP	• ACL • HTTP • TCP/IP
Número Permanente de Agentes	• 4	• 5
Lenguaje de Programación	• JAVA	• JAVA
Robustez	• No existe gestión de incidencias	• Agente dedicado a gestión de incidencias • Estadística de funcionamiento
Reutilización de Recursos	• Funcionalidad integrada en la arquitectura	• Servicios Reutilizables • Arquitectura Reutilizable
Distribución de Recursos	• Cada agente realiza un servicio completo	• Distribución y Replicación de Servicios • Los agentes ejecutan tareas especializadas • Instanciación en Tiempo de ejecución
Carga Computacional del Sistema	• Centralizada	• Distribuida

Analizando la **Tabla 6.5** el MAS implementa una arquitectura basada en agentes centralizada y por el contrario SCODA distribuida. La ventaja que ofrece una arquitectura distribuida es la capacidad de repartir recursos de forma más eficiente. En cuanto a los protocolos de comunicación empleados, ambas arquitecturas utilizan HTTP como comunicación externa y ACL para comunicación interna entre los agentes. Sin embargo, SCODA utiliza sockets TCP/IP para la comunicación con los servicios externos de forma que estos servicios serán ejecutados en máquinas diferentes. Algo a destacar es que el MAS está constituido de 4 agentes permanentes y SCODA de 5. En SCODA existe un agente que es el responsable de llevar a cabo la gestión de incidencias y errores que se producen en tiempo de ejecución. SCODA permite una completa gestión de incidencias y tiene una alta tolerancia a fallos de forma que si uno de los agentes no funcionase correctamente el sistema no caería ya que tiene la capacidad de reinicio en tiempo de ejecución. Esta característica está ausente en el MAS, obligando a reiniciar manualmente los agentes que hayan sufrido alguna interrupción. Además, SCODA proporciona un diario de operaciones a través del que se puede monitorizar el funcionamiento del sistema y gestionar la capacidad de replicación de servicios.

En cuanto a la reutilización y distribución de los recursos, el MAS integra toda la funcionalidad en su arquitectura, de forma que la ejecución se realiza de manera centralizada, consumiendo así recursos de la misma máquina donde está desplegada su arquitectura, y obligando a reprogramar los agentes para cualquier otro desarrollo. Por el contrario, SCODA distribuye los recursos de forma que la arquitectura en si

es estándar en cualquier problema, es decir, siempre se compone de los mismos agentes, y estos agentes siempre tienen la misma estructura. Además la distribución de servicios permite que sean reutilizados en otros desarrollos de forma individual o colectiva.

Finalmente se alude a la carga computacional que existe en la ejecución de ambas arquitecturas. La arquitectura del MAS es centralizada, obligando a consumir los recursos necesarios de la máquina donde se ejecuta el sistema, y por lo tanto, en operaciones que requieren un alto consumo de recursos abarca un porcentaje muy elevado de los mismos, no permitiendo la ejecución de otro tipo de tareas. Por el contrario SCODA distribuye la carga computacional de forma que hace recaer toda esa carga en los servicios externos. La carga existente en la máquina donde se ejecuta SCODA es independiente de que las tareas requeridas necesiten un uso excesivo de recursos, ya que la ejecución de los mismos se realiza de forma distribuida e independiente.

6.4.1. Resultados de la Eficiencia del Modelo Teórico

Para llevar a cabo un estudio de la eficiencia del modelo teórico, y debido a la tipología del caso de estudio que está centrado en la integración y funcionamiento colaborativo del sistema, se hace necesario llevar a cabo una serie de pruebas de rendimiento que demuestre la hipótesis de partida. Para ello se realizarán un conjunto de peticiones de forma simultánea sobre SCODA solicitando alguno de los servicios que provee el caso de estudio, y posteriormente realizar las mismas operaciones sobre el MAS implementado en esta comparativa.

Algo a tener en cuenta es que debido a la distribución de servicios que se lleva a cabo en SCODA, los tiempos de respuesta en ambos casos van a venir marcados por la capacidad de proceso de la máquina donde se aloje el servicio. Es por ello que con el fin de determinar de la forma más veraz posible el rendimiento del modelo teórico, se decide que los servicios a solicitar serán de gestión de inventario en sus tres variantes, *Economic Order Quantity Model*, *Economic Order Quantity Model con Descuentos y Economic Order Quantity Model con Órdenes Pendientes*, ya que su coste computacional es menor que el que puedan tener los servicios de predicción o de optimización de rutas.

Las pruebas a realizar serán las siguientes:

- Peticiones correctas.
- Peticiones con fallos en los parámetros de entrada.
- Peticiones eliminando uno de los agentes responsables.
- QoS a nivel de control y corrección de incidencias en el sistema.

Tabla 6.6 Tiempo de respuesta (ms) ante peticiones simultáneas para un MAS y SCODA

Nº Peticiones	Tiempo de Respuesta MAS	Tiempo de Respuesta SCODA	Tiempo por Petición MAS	Tiempo por Petición SCODA
1	100	100	100	100
5	660	550	132	110
12	2592	1920	216	160
21	5040	3360	240	160
36	ERROR	5760	ERROR	160
60	ERROR	9600	ERROR	160
100	ERROR	17000	ERROR	170

La **Tabla 6.6** muestra los tiempos de respuesta del MAS implementado frente a los de SCODA para peticiones de servicio de inventario en las tres variantes consideradas. Las peticiones son aleatorias, tanto para las variables, como para los datos, con el fin de determinar el rendimiento de ambos sistemas. Como se observa, el tiempo para una petición es de 100 ms en ambos casos, sin embargo a medida que se aumentan las peticiones simultáneas SCODA responde de forma más rápida debido a que la ejecución de las peticiones se realizan a través de hilos diferentes, lo que permite gestionar de forma más rápida la resolución de las peticiones. Algo importante es que a partir de 36 peticiones simultáneas el MAS genera un error en la plataforma JADEX, lo que implica que la acumulación de esas peticiones satura la capacidad del agente. Las **Figura 6.14 y 6.15** muestran de forma gráfica lo expuesto anteriormente.

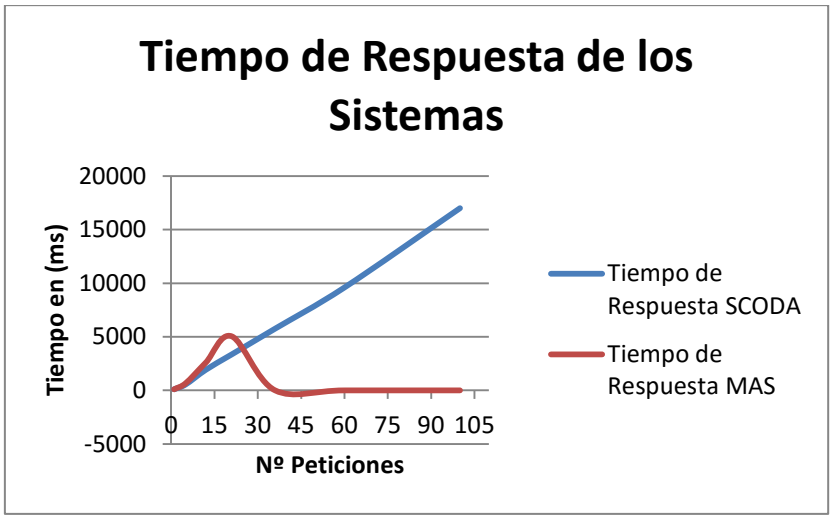


Figura 6.14 Detalle gráfico de respuesta (ms) ante peticiones simultáneas para un MAS y SCODA

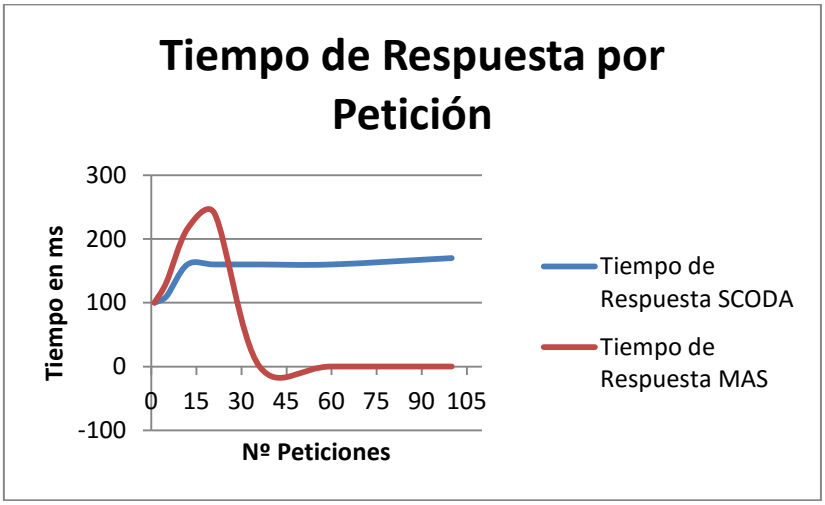


Figura 6.15 Detalle gráfico de respuesta (ms) por petición, ante peticiones simultáneas para un MAS y SCODA

Tal y como se especificaba al principio de este punto, la siguiente prueba que se realiza es la emisión de peticiones en los mismos términos que la prueba anterior, pero introduciendo diversos fallos en los parámetros requeridos, servicios no existentes, etc., de forma aleatoria, para comprobar la reacción de ambos sistemas. La **Tabla 6.7** muestra que los resultados obtenidos siguen una misma línea que en el caso anterior, lo que implica que en ambos la respuesta de la existencia de un error no hace variar prácticamente los tiempos. En las **Figura 6.16 y 6.17,** se muestra lo expuesto de una forma gráfica.

Tabla 6.7 Tiempo de respuesta (ms) ante peticiones simultáneas para un MAS y SCODA introduciendo Errores

Nº Peticiones	Tiempo de Respuesta MAS	Tiempo de Respuesta SCODA	Tiempo por Petición MAS	Tiempo por Petición SCODA
1	105	105	105	105
5	654	545	130,8	109
12	2624,4	1944	218,7	162
21	5260,5	3507	250,5	167
36	ERROR	5868	ERROR	163
60	ERROR	10140	ERROR	169
100	ERROR	17800	ERROR	178

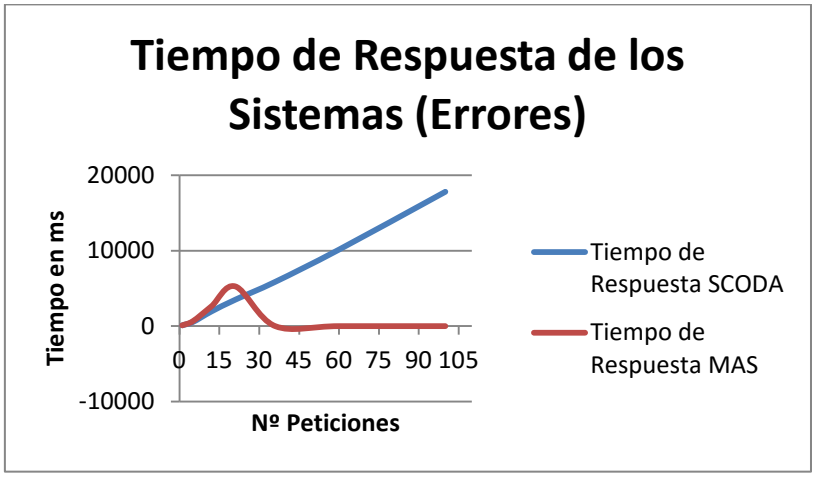


Figura 6.16 Detalle gráfico de respuesta (ms) ante peticiones simultáneas para un MAS y SCODA, introduciendo errores

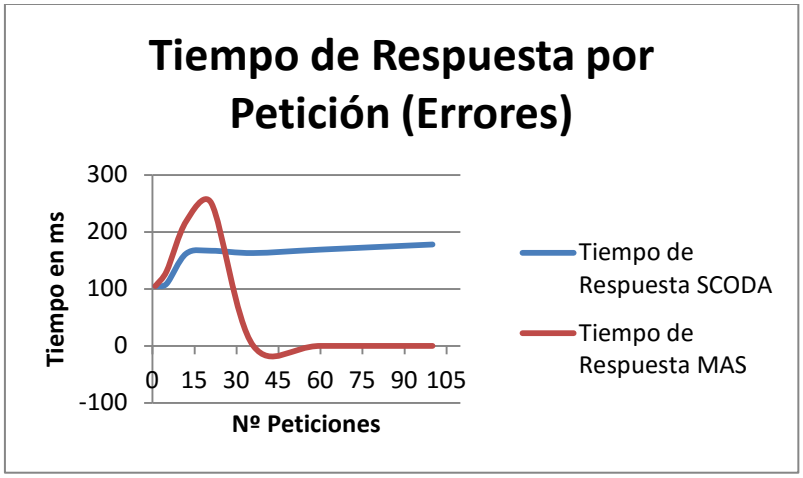


Figura 6.17 Detalle gráfico de respuesta (ms) por petición, ante peticiones simultáneas para un MAS y SCODA

La siguiente prueba que se realiza sobre ambos sistemas consiste en eliminar a uno de los agentes responsables de atender la petición y ver la reacción de los sistemas. En cuanto la MAS, directamente se elimina el agente que implementa las funcionalidades de ejecutar los modelos de inventario, lo cual la petición no se resuelve devolviendo un error, ya que el agente no puede hacerse cargo de resolver la petición.

En SCODA, se elimina al *CommunityController* responsable de la gestión de inventarios. Como se especifica en el capítulo 5, la propia arquitectura tiene capacidad de detectar problemas en los agentes que la componen y reiniciar esos agentes. Por lo tanto, el *CommunityController* es reiniciado y la petición del servicio se ejecuta correctamente.

Finalmente la comparación entre los dos sistemas pasa por la gestión de la QoS a nivel monitorización de incidencias y estadísticas de aciertos y fallos en el sistema. En el segundo tipo de pruebas que se realizaron introduciendo fallos en los parámetros de entrada, los tiempos de respuesta eran similares, sin embargo el MAS no da información del número de peticiones atendidas correctamente, el tipo de fallo que se ha producido, etc., y sin embargo SCODA lleva un completo control de las incidencias ocurridas en el sistema a través del *QualityAgent.*

Las pruebas realizadas y los resultados expuestos anteriormente llevan a validar el modelo teórico presentado en este trabajo. Respecto de su comportamiento ante situaciones donde se requiere una respuesta eficiente, SCODA ha obtenido unos resultados mejores que el sistema MAS implementado.

6.5. Especialización en el Caso de Estudio

En la sección 3.2.1 se presentaban las *Redes Empresariales* como un modelo de colaboración entre diferentes empresas para obtener beneficios mayores que de forma individual. A continuación se calcula el grado de cooperación entre las entidades que componen el sistema implementado bajo SCODA, y a partir del cual se valorará el tipo de cooperación y su grado. Para ello se toma cada una de las Comunidades Inteligentes Especializadas, y los agentes responsables de las comunicaciones y de la gestión de errores que componen el sistema como una unidad dentro de la red empresarial.

$$\begin{cases} N_{EM} = 5, Predicción, Inventario, Rutas, Comunicador \\ \qquad\qquad Calidad \\ N_{SP} = 6, Predicción, Gestión\ de\ Inventario, Gestión\ de\ Rutas \\ Comunicaciones, Gestion\ de\ Errores, Proceso\ Completo \end{cases} \tag{6.1}$$

$$G_C = \frac{N_{SP}}{N_{EM}} = \frac{6}{5} = 1.2 \tag{6.2}$$

Atendiendo a (3.11; 3.12) y teniendo en cuenta el resultado en (6.2) se puede concluir que el Grado de Cooperación Horizontal es medio, y el Grado de Cooperación Vertical es alto. Estos resultados son de esperar ya que los integrantes

del sistema cooperan para lograr alcanzar una meta común, es decir resolver el proceso completo, que de forma individual no lograrían alcanzar.

6.6. Análisis del Caso de Estudio y Conclusiones

En el presente capítulo se ha llevado a cabo la implantación de un sistema basado en la arquitectura objetivo de este trabajo para dar validez a la hipótesis de partida. Para ello se ha seleccionado un caso de estudio en un entorno real empresarial que aglutina diferentes tareas dependientes y que pueden ser automatizadas, de forma que pueda validarse el modelo teórico. El problema propuesto contiene el ciclo completo de desarrollo de negocio de la empresa **Bahía Príncipe Congelados S.L.** la cual amablemente ha colaborado en esta investigación permitiendo la implantación de este sistema multiagente sobre un conjunto de sus activos.

La evaluación del sistema se ha realizado durante cinco meses comprendidos entre el 01/01/2012 y el 31/05/2012, durante los cuales el sistema multiagente desplegado ha funcionado en los términos propuestos en este capítulo, de forma que nos ha permito recabar la información necesaria para realizar un análisis de su funcionamiento y extraer una serie de conclusiones que se presentan en esta sección.

A lo largo de estos cinco meses, el sistema multiagente bajo SCODA ha presentado un rendimiento óptimo, en cuanto a su funcionamiento se refiere, ya que al analizar el fichero de estadísticas de funcionamiento diario que el *QualityAgent* mantiene, tan solo en un 4% de los casos ha existido algún error, que corresponde un fallo en los servicios de las comunidades *(Servicio Inactivo).*

Estos fallos han venido dados por errores inesperados en el servidor donde se alojan los servicios de las CIE, ya que atendiendo al principio de computación distribuida, los servicios se han alojado en un servidor diferente a los de las CIE. Este hecho confirma que las comunicaciones internas entre los agentes, la CIE, el control realizado por parte del *QualityAgent*, la instanciación y destrucción de los equipos de trabajo de cada CIE, y en general el funcionamiento de la arquitectura ha resultado robusta y fiable en este tipo de aplicaciones.

Respecto del rendimiento de las CIE, en cuanto a la ejecución de sus servicios se refiere, los resultados obtenidos han sido satisfactorios. Si bien, es cierto que cada una de las CIE han sido probadas de forma individual y se han expuesto los resultados en las secciones anteriores, a la hora de realizar la implementación del sistema completo de forma que estas CIE colaboren a través de SCODA, se ha comprobado que el funcionamiento, a nivel de servicios, es completamente igual al de su ejecución de forma individual, lo que permite comprobar que se sigue la dirección correcta hacia la filosofía de reutilización de estas CIE.

En cuanto a como se ha llevado a cabo el desarrollo del sistema, se puede afirmar que el coste de implementar el sistema multiagente ha sido cuanto más sencillo debido a que el único esfuerzo de implementación realizado corresponde a los servicios de cada una de las CIE ya que el despliegue de la arquitectura del sistema se ha generado de forma automática a través de la herramienta *Community Agent Generator*, presentada en la sección 5.5. de este trabajo, y desarrollada dentro del marco de la misma para tales fines. Esto confirma que esta herramienta permite el desarrollo de sistemas multiagente basados en SCODA de una forma rápida y eficaz.

Una vez finalizado el periodo de prueba se ha realizado un análisis directamente sobre la repercusión y aceptación que el sistema ha tenido en la empresa. Este análisis no se centra en la propia arquitectura del sistema, sino en los servicios que esta provee ya que es importante para la empresa el analizar los resultados obtenidos a partir de la implantación del sistema.

A lo largo de los dos primeros meses el responsable de la cámara ha realizado un seguimiento de los informes generados por el sistema indicando, que los resultados obtenidos en los informes de predicción respecto de la demanda real difieren en un 2%-5%, siendo el error cometido por el responsable de la cámara un 1,5%-4%. Respecto de las órdenes óptimas de pedido, modifican la metodología seguida, ya que antes de la implantación del sistema se realizaba un solo pedido con una cantidad mucho mayor y sin tener en cuenta un punto de reorden, lo cual en ocasiones era perjudicial ya que existía escasez.

Durante los tres meses siguientes al periodo de prueba no se ha realizado ninguna supervisión de los resultados arrojados por el sistema, no existiendo escasez de ningún producto en stock debido al control de un punto de reorden y que el error cometido en las predicciones es completamente asumible en nuestro caso, permitiendo un correcto flujo de trabajo en la empresa. Este hecho permite un ahorro de tiempo importante ya que la supervisión del inventario sobre los productos expuestos al sistema no es necesaria, siendo el propio sistema el que lo gestiona de una forma satisfactoria.

Por su parte, la optimización de rutas ha sido completamente satisfactoria para la empresa, permitiendo un importante ahorro de combustible debido a que la distancia recorrida en gran parte de las rutas ha sido menor. Además la gestión dinámica de las mismas ha permitido incluir nuevas localidades con la garantía de que el recorrido por las mismas es el óptimo.

Sin embargo, y a pesar de que el sistema funciona de forma satisfactoria, existen muchos procesos mejorables. La predicción de la demanda es algo complejo y que requiere un análisis profundo en función de cada producto. Esta mejora no recae directamente sobre la implementación de SCODA, sino sobre el diseño e implementación de los servicios de las CIE. Además existen otros muchos procesos dentro de la empresa que podrían ser mejorables ampliando el sistema con nuevas CIE y nuevos servicios, tales como la automatización de los pedidos a proveedores,

la automatización del control de los surtidores de gasoil o unión del sistema con el software de gestión de ventas que la empresa posee.

Se ha comparado SCODA con un sistema basado en agentes (MAS) implementado para la ocasión evaluando una serie de características en ambos sistemas para el problema propuesto en este caso de estudio. Se ha demostrado que la implementación bajo SCODA resulta ser más eficiente computacionalmente ya que la distribución de servicios libera a los agentes de carga computacional. Además se ha puesto de manifiesto la capacidad de reutilización de SCODA y su robustez. La tolerancia a fallos que admite SCODA permite resolver fallos en la arquitectura en tiempo de ejecución, y llevar un control en tiempo real del funcionamiento del sistema. Finalmente, se comprueba que los tiempos de respuesta de la arquitectura más bajo que el del MAS, además de proveer una completa gestión de incidencias cuando existen posibles fallos.

A partir de este momento se propone una segunda evaluación, mejorando la arquitectura hardware que alberga el sistema, que comprende desde el 01/09/2012 hasta el 31/09/2013. Esta mejora en cuanto a equipos y una nueva implementación de la arquitectura de red y de su velocidad ha supuesto una mayor capacidad de proceso y una mayor velocidad en las comunicaciones, lo que implica un mejor funcionamiento del sistema a partir de estas mejoras externas.

La posibilidad que ofrece el caso de estudio para comprobar el modelo teórico planteado es idónea ya que se hace referencia a tres procesos claramente diferenciados, que a su vez se complementan para lograr un objetivo global. Es precisamente la filosofía seguida en este trabajo, la especialización a nivel organizacional la que permite que cada parte del sistema alcance una serie de objetivos individuales, y que en conjunción con otras partes del sistema se alcance una meta global.

Concluyendo, se puede afirmar que la implementación de sistemas multiagente sobre la arquitectura SCODA, y en nuestro caso concreto la implementación de un sistema de gestión y mejora de procesos logísticos en un entorno real, es completamente viable, validando así la hipótesis de partida. Teniendo en cuenta el planteamiento inicial sobre la estandarización que ha de poseer la arquitectura, hemos comprobado que problemas de diversa índole han podido ser resueltos a través de SCODA de forma modular, y que a través de una coordinación escalada, se han podido resolver problemas de magnitud mayor.

CAPÍTULO 7.
CONCLUSIONES

7. CONCLUSIONES

La creciente aplicación de sistemas multiagente en la resolución de problemas basados en computación inteligente y distribuida lleva consigo la necesidad de buscar nuevas soluciones que faciliten el desarrollo de los mismos. En esta memoria se ha presentado SCODA, una arquitectura para el desarrollo de sistemas multiagente que facilita y optimiza su implementación, basándose en los principios de estandarización, especialización, facilidad de implementación, reutilización y computación distribuida en cuanto a los servicios ofrecidos por la misma.

Cada uno de los objetivos perseguidos en esta investigación que se han ido completando, han traído consigo nuevas ideas que han permitido dar forma a la arquitectura marco, presentada en esta memoria. En primer lugar, se ha realizado un análisis exhaustivo de la Teoría de la Organización y los tipos de organizaciones. En este análisis se han contrastado las características de cada una de ellas, tanto en humanos como aplicadas a los sistemas multiagente. Como tipo de organización surge la definición de Comunidad Inteligente. Basada en el concepto de comunidad, constituye el núcleo de la arquitectura presentada en esta memoria y posee un carácter más espontáneo y consta de un número de agentes menor que las organizaciones analizadas. Este hecho implica que es el tipo de organización ideal para nuestra arquitectura.

La especialización del puesto de trabajo de un individuo dentro de una organización es una característica importante. Cuando se enfoca un sistema multiagente desde un punto de vista organizacional se ha de tener en cuenta los cometidos de cada uno de los agentes y las relaciones entre agentes y con el entorno. A partir de la especialización que existe en las organizaciones humanas, se ha enfocado hacia los sistemas multiagente definiendo una serie de valores que permitan medir el grado de especialización intra-organización y extra-organización. Estos valores han permitido clasificar las organizaciones de agentes en función de su especialización, y definir las Comunidades Inteligentes Especializadas como unidad fundamental dentro de la arquitectura SCODA.

Una vez analizados los aspectos fundamentales de organizaciones y especialización, y a través de una exhaustiva revisión de la teoría de agentes, se ha propuesto SCODA, una arquitectura que se basa en los principios de estandarización, reutilización, especializada, de fácil implementación y distribuida. Para ello se ha realizado una formalización del modelo empleando herramientas para la Ingeniería del Software Orientada a Agentes (AOSE, Agent Oriented Software Engineering). Una vez realizada esta formalización se ha llevado a cabo su implementación sobre un problema real para comprobar su funcionamiento y su utilidad en los términos propuestos. A partir de esta implementación se ha podido evaluar SCODA, tanto a nivel arquitectónico como a nivel del problema a resolver, y se ha podido comparar

con otros modelos permitiendo abrir nuevas líneas de investigación y mejora de la arquitectura propuesta.

A partir de la definición formal e implementación de la arquitectura propuesta en esta memoria se ha podido probar la hipótesis de partida. La arquitectura definida es modular, especializada y posee un comportamiento organizacional. Es estándar, probándose este hecho ya que la estructura de las Comunidades Inteligentes Especializadas es la misma independientemente del problema a resolver y del desarrollo donde se requieran. La reutilización de los módulos que componen SCODA es completa permitiendo la utilización de una Comunidad Inteligente Especializada en diferentes desarrollos basados en SCODA. En función del número de tareas que realiza cada uno de los agentes de SCODA y atendiendo a la propuesta realizada en el capítulo 3 de esta memoria para la medición del grado de especialización dentro de una organización, se considera a los agentes de SCODA y a la propia arquitectura, especializada. Debido a que la estructura de los agentes es igual ya que donde reside la funcionalidad es en los servicios, la implementación de un sistema basado en SCODA facilita la labor a los desarrolladores. Para comprobar si realmente se facilita el desarrollo en estos términos la implementación del caso de estudio se ha llevado a cabo con la herramienta *"Community Agent Generator"*, propuesta en la sección 5.5 de esta memoria, la cual ha permitido implementar los componentes necesarios del sistema de forma modular y escalada. Finalmente, SCODA está propuesta en términos de ser una arquitectura distribuida, de forma que la funcionalidad del propio sistema resida de forma escalada en las Comunidades Inteligentes Especializadas, y a su vez en servicios asociadas a ellas. De esta forma la carga computacional puede ser distribuida liberando a los propios agentes que componen la arquitectura.

7.1. EVALUACIÓN DE SCODA

Para poder evaluar la arquitectura propuesta en este trabajo se ha llevado a cabo su implantación en un entorno real descrito a lo largo del capítulo 6. El problema propuesto en la evaluación contiene el ciclo completo de desarrollo de negocio empresarial, englobando diferentes objetivos, y por lo tanto dándonos la posibilidad de poder probar la versatilidad de SCODA. Además, teniendo en cuenta que el caso de estudio planteado se centra en la integración y funcionamiento del modelo teórico, se ha realizado una comparativa con un sistema multiagente centralizado, comprobando así la validez y mayor rendimiento de SCODA sobre el sistema multiagente planteado.

A lo largo del periodo de pruebas, se ha podido comprobar la robustez del sistema, debido a que los únicos fallos hallados se han producido en los servicios de las Comunidades Inteligentes Especializadas. Estos servicios han sido interrumpidos por fallos del propio sistema operativo, ya que no se han implantado en un servidor dedicado para ellos, y por fallos de corriente, debido a que no se disponía de un SAI (*Sistema de Alimentación Ininterrumpida*) para este servidor, si bien es cierto que

estos fallos se han ido paliando en la segunda fase de pruebas con la mejora de los equipos y de la infraestructura de la red.

Respecto del rendimiento de las CIE, en cuanto a la ejecución de sus servicios se refiere, los resultados obtenidos han sido satisfactorios. Si bien, es cierto que cada una de las CIE han sido probadas de forma individual y se han expuesto los resultados obtenidos, a la hora de realizar la implementación del sistema completo de forma que estas CIE colaboren a través de SCODA, se ha comprobado que el funcionamiento, a nivel de servicios, es completamente igual al de su ejecución de forma individual. Este hecho permite comprobar que se sigue la dirección correcta hacia la filosofía propuesta de reutilización de estas CIE.

Los resultados obtenidos por el sistema en el caso de estudio, han sido completamente satisfactorios para la empresa. Si bien es cierto que el funcionamiento del sistema se ha limitado a un conjunto de recursos reducido, debido a la no intromisión en el funcionamiento diario del negocio, ha permitido una reorganización del flujo de trabajo que ha beneficiado tanto a los trabajadores como a la empresa en general. De esta forma, se deja abierta una colaboración para el desarrollo integral del negocio utilizando un sistema multiagente basado en SCODA.

A pesar de los prometedores resultados que se han obtenido a lo largo de este trabajo, existen determinados aspectos que son mejorables dentro del mismo. La utilización REST como protocolo de comunicación de la arquitectura SCODA con las aplicaciones externas, constituye un punto fuerte ya que se trata de un protocolo estándar que puede ser implementado independientemente del lenguaje de programación. Sin embargo, la comunicación de las CIE con sus respectivos servicios se realiza a través de sockets, lo que implica una complejidad si los servicios se quieren implementar en diferentes lenguajes de programación debido a las diferencias que existen a nivel interno entre los mismos. Otro de los puntos fuertes deriva en la capacidad de SCODA para descargar a los agentes del sistema computacionalmente, ya que son los servicios de las CIE los que implementan las funcionalidades de las mismas, de una forma modular y distribuida, permitiendo así, su reutilización en desarrollos con diferentes objetivos globales.

Respecto a la completa formalización de SCODA, todavía no es un hecho de forma que permita a otros investigadores su utilización de forma sencilla, ya que no se encuentra depurada completamente. Otro punto negativo es que el conjunto de agentes de la arquitectura ha de ser implementado sobre las herramientas y lenguajes de programación que soporte la plataforma sobre la que se realice su despliegue, si bien es cierto, que la elección de JADEX como plataforma de despliegue para SCODA, se ha hecho porque se considera un motor de razonamiento con la capacidad de ejecutarse de forma independiente. La investigación realizada en este trabajo ha proporcionado otra filosofía en el desarrollo de sistemas inteligentes, y con ello sistemas multiagente, de forma que estos sean considerados como pequeños grupos (Comunidades Inteligentes Especializadas) que, de forma modular, escalada y coordinada, tengan la capacidad de resolver problemas complejos, y puedan reutilizarse en diferentes desarrollos.

7.2. COMPARATIVA DE SCODA

Para evaluar SCODA frente a otros tipos de sistemas multiagente se ha desarrollado un sistema que cumpla los mismos cometidos que el sistema desplegado bajo SCODA para el caso de estudio. La comparativa puede verse en la sección 6.4 de esta memoria, y viene dada en función de una serie de características referidas al funcionamiento en sí de la arquitectura planteada.

El sistema multiagente implementa una arquitectura basada en agentes, centralizada y por el contrario SCODA distribuida. La ventaja que ofrece una arquitectura distribuida sobre una arquitectura centralizada es la capacidad de repartir recursos de forma más eficiente, y por tanto no cargar computacionalmente a los propios agentes que componen el sistema. Otra de las fortalezas de SCODA frente a la arquitectura con la que se compara es la gestión de incidencias y errores producidos en el sistema en tiempo de ejecución. SCODA permite una completa gestión de incidencias y tiene una alta tolerancia a fallos de forma que si uno de los agentes no funcionase correctamente el sistema tiene la capacidad de reinicio en tiempo de ejecución. Esta característica está ausente en muchos sistemas multiagentes, obligando a reiniciar manualmente los agentes que hayan sufrido algún percance. Otra ventaja que se da en SCODA es que proporciona un diario de operaciones a través del que se puede monitorizar el funcionamiento del sistema y gestionar la capacidad de replicación de servicios.

La reutilización de las funcionalidades del sistema multiagente son muy limitadas ya que se encuentran integradas en su arquitectura, de forma que la ejecución se realiza de manera centralizada, consumiendo así recursos de la misma máquina donde está desplegada su arquitectura, y obligando a reprogramar los agentes para cualquier otro desarrollo. Por el contrario, SCODA distribuye los recursos de forma que la arquitectura en si es estándar en cualquier problema, es decir, siempre se compone de los mismos agentes, y estos agentes siempre tienen la misma estructura. Además la distribución de servicios permite que sean reutilizados en otros desarrollos de forma individual o colectiva.

Finalmente se alude a la carga computacional que existe en la ejecución de ambas arquitecturas. La arquitectura del sistema multiagente es centralizada, obligando a consumir los recursos necesarios de la máquina donde se ejecuta el sistema, y por lo tanto, en operaciones que requieren un alto consumo de recursos abarca un porcentaje muy elevado de los mismos, no permitiendo la ejecución de otro tipo de tareas. Por el contrario SCODA distribuye la carga computacional de forma que hace recaer toda esa carga en los servicios externos. La carga existente en la máquina donde se ejecuta SCODA es independiente de que las tareas requeridas necesiten un uso excesivo de recursos, ya que la ejecución de los mismos se realiza de forma distribuida e independiente.

7.3. CONTRIBUCIONES DE LA INVESTIGACIÓN

A continuación se describen las principales contribuciones realizadas a partir de esta investigación:

Definición de Comunidad Inteligente: Se ha realizado un análisis de la Teoría de la Organización enfocado a recabar las similitudes entre las organizaciones humanas y las organizaciones de agentes. La finalidad de este análisis es la adaptación de una forma de organización que entrase dentro de los parámetros de nuestra investigación. Finalmente se definen las Comunidades Inteligentes como forma de organización de agentes, las cuales se adaptan perfectamente a nuestra investigación.

Estudio del concepto de especialización como proceso de mejora productiva: Se ha llevado a cabo un análisis de los diversos tipos de especialización existentes en las organizaciones empresariales y sistemas artificiales. Este análisis permitirá adaptar esta especialización a los sistemas multiagente de forma que se produzca una optimización en su diseño y posterior implementación. Para ello se han definido una serie de medidas que reflejan el grado de especialización dentro de las organizaciones, que también son aplicables a los sistemas multiagentes.

Análisis y diseño de sistemas multiagente: Para la definición formal basada en la Ingeniería del Software, se ha llevado a cabo una revisión de las diferentes herramientas análisis y diseño de sistemas multiagente. Finalmente se ha optado por combinar Gaia y AML de forma esta combinación permita modelar, de forma eficiente, tanto los agentes que constituyen un sistema como las interacciones existentes entre ellos.

Propuesta de una arquitectura para el desarrollo de sistemas multiagente: Se ha propuesto una arquitectura, basada en Comunidades Inteligentes, que optimiza el desarrollo de sistemas multiagente independientemente para el fin que se creen. Esta arquitectura se basa en la especialización y el crecimiento modular escalado, implementando sus funcionalidades de forma distribuida a través de servicios. Sobre la arquitectura propuesta se ha desplegado un sistema multiagente, en un entorno empresarial real, obteniendo una evaluación positiva en el funcionamiento de cada uno de sus módulos de forma independiente, como de forma conjunta.

Creación de una herramienta para el desarrollo y despliegue de arquitecturas SCODA: Se ha llevado a cabo el diseño e implementación de la herramienta *Community Agent Generator*. Esta herramienta permite llevar a cabo desarrollos basados en SCODA de forma rápida y eficiente. Además, permite llevar a cabo la integración de diferentes Comunidades Inteligentes Especializadas en desarrollos ya realizados, incrementando así, la reutilización de las mismas.

Aplicación de la arquitectura a un entorno empresarial: Se ha aplicado la arquitectura propuesta en este trabajo a un entorno empresarial real para llevar a cabo su evaluación. Esta aplicación ha consistido en el desarrollo de un sistema multiagente basado en SCODA a partir del cual se han definido soluciones aplicadas a problemas logísticos tales como la gestión de inventarios y la optimización de rutas. De esta forma se ha podido comprobar la viabilidad de la propuesta obteniendo unos resultados satisfactorios, en cuanto a su despliegue y posterior funcionamiento, mejorando el flujo de trabajo de la empresa.

Intercambio de conocimiento: A lo largo del proceso de desarrollo de este trabajo se ha incidido en obtener un intercambio de conocimiento con otros investigadores en áreas afines a esta investigación, con el fin de mejorar de forma permanente los conocimientos adquiridos.

Además, los conceptos fundamentales de la arquitectura propuesta han sido, y siguen siendo, evaluados en varios proyectos de investigación e innovación educativa, y desarrollos empresariales con el fin de seguir mejorando su definición y ampliar su campo de aplicación. Los proyectos citados son los siguientes:

- Desarrollo de un Sistema Inteligente para el Tratamiento Estadístico de Datos. Llevado a cabo conjuntamente por la empresa O_2 y por el Plan Municipal sobre Drogodependencias, del Ayuntamiento de Zamora. 2010.

- Aplicación de Redes Neuronales Artificiales y Visión Artificial para avalar la calidad de la Marca de Garantía del Chorizo Zamorano. 18.VAZQ 463A.C.06. Otorgado por Caja Rural. 2010.

- Desarrollo de una Plataforma Software as Services. 022/10/ZA/0008. Otorgado por la Junta de Castilla y León a la empresa MkZ Soluciones de Ingeniería, S.L. 2011.

- Gestión de Tutorías Automatizadas Vía Web. ID11/027. Otorgado por la Universidad de Salamanca en la convocatoria de ayudas para la innovación docente. 2012.

- Gestión Integral de la Producción. Proyecto desarrollado desde la empresa Servicios de Innovación Zamoranos C.B. para uno de sus clientes. 2013.

- Sistema Inteligente para la Gestión del Flujo de Trabajo del Consorcio Provincial de Bomberos de Zamora. Proyecto desarrollado desde la empresa Servicios de Innovación Zamoranos C.B. para uno de sus clientes. 2014.

- Sistema Inteligente para la gestión de historias y cálculo de parámetros de una clínica especializada en nutrición. Proyecto desarrollado desde la empresa Servicios de Innovación Zamoranos C.B. para uno de sus clientes. 2015.

7.4. Líneas de Trabajo Futuras

A continuación se presentan algunas de las posibles líneas de trabajo futuras a partir de esta investigación:

Desarrollo de un framework para implementar sistemas basados en SCODA: Se propone la creación de un framework que permita la creación de sistemas basados en la arquitectura SCODA, con el fin de implementar solamente las características necesarias para el funcionamiento correcto y evitando así muchas de las posibilidades que ofrecen las plataformas existentes no necesarias en la ejecución de SCODA.

Servicios de las Comunidades Inteligentes Especializadas: Se propone la implementación de los servicios de las CIE a partir de tecnologías que no dependan de la definición interna de las plataformas y lenguajes de programación, así pues, estos servicios seguirán una implementación SOA.

Medición de la Especialización en sistemas ya desplegados: Se propone el desarrollo de una metodología que permita medir el grado de especialización de sistemas ya desplegados, en los términos propuestos en este trabajo.

Metodología de análisis y diseño: A la vista del la arquitectura presentada en este trabajo y sus características, parece importante el desarrollo de un metodología para el análisis y diseño de este tipo de sistemas, como método ágil de desarrollo de sistemas multiagente, basada en las actuales metodologías de Desarrollo Ágil.

Comunidades Inteligentes Especializadas: Atendiendo a la definición de Comunidad Inteligente Especializada propuesta en este trabajo, se pretende la creación de un repositorio de CIE, con diferentes finalidades, las cuales puedan ser utilizas en diversos desarrollos.

Aplicación a nuevos problemas: Con la finalidad de realizar una comprobación de la arquitectura que permita su evaluación, de forma exhaustiva, en diversas líneas de desarrollo, se propone su despliegue sobre sistemas multiagente aplicados a entornos prácticos.

Pruebas y validación: Se hace necesario llevar a cabo pruebas continuas, minuciosas y exhaustivas que permitan evaluar la arquitectura propuesta sobre factores de calidad en cuanto a tiempos de desarrollo de sistemas basados en SCODA, facilidad de implementación, tiempos internos de comunicación y

ejecución de servicios, etc.. A través de estas pruebas se permitirá una mejora continua de la arquitectura, y con ello, sistemas más robustos y depurados.

CAPÍTULO 8. RESEARCH OVERVIEW

8. RESEARCH OVERVIEW

8.1. INTRODUCTION

The development of more and more complex Systems involves the necessity of making components that can be reused in other developments, being compatible. This philosophy is the one, the technology aimed at objects follows, (*Lieberherr, 1996; Elrad et al., 2001; Rashid et al., 2003; Kiczales et al., 1997*) where the objects are encapsulated in an independent way and can be reused in different developments with very different purposes accelerating the time of development.

This approach related to the Multi-Agent Systems (*Multi-Agent Systems, MAS*) implies that the development of an application, based on Multi-Agent systems, has to be able to be reused in other type of developments, no matter the global aim. In order to achieve this, and bearing in mind the definition of Multi-Agent systems as a powerful computer model, based on software entities that cooperate together through an autonomous, cooperative and distributed perspective (Amigoni y Fugini 2007), we have to think that the group of agents that makes a Multi-Agent system must have standard characteristics and size so that it can be reused.

The concept of organization has been extensively studied in areas such as economy, sociology and psychology. Besides, several authors have applied the concept of organization to the development of Multi-Agent systems. (*Zambonelli et al., 2000; Ferber et al., 2003; Giorgini et al., 2004; Furtado 2005; McCallum 2005; Hubner et al. 2006*).

The application of the concept of organization to a Multi-Agent system, increases its efficiency because the aims (whether they are individual or collective), that must be accomplished, are controlled. Some rules of behavior between agents are established and, in some way, there is a division of tasks or specialization, that is to say, the system operation behaves as if It was established by a protocol (*Zambonelli et al., 2003; Pavón et al. 2003; Pavón et al. 2005; Dignum y Dignum, 2006; Hubner et al. 2006*).

If we also want a Multi-Agent system to be able to be used in different developments no matter the global aim, it would be necessary to determine a new specialized modular architecture based on an organization with a tiny size that can establish the rules of coordination among the agents. This type of architecture would be highly specialized regarding the organizational level, because the architecture of a Multi-Agent system determines the composition of the system and the mechanisms that the Agents use to interact with their environment (*Corchado, 2005*).

The aim is to design and implement a standard Multi-Agent architecture that can be used in different developments in a very efficient way. This architecture is based on a business organization philosophy that improves the integration and the development of the Multi-Agents, promoting the distribution of the agents that define the system and the services they provide.
In order to do that, the concept of Agents Community as an organization unit as well as the concept of specialization applied to the Agents Community. In order to get an optimization in the Management and the execution of the system that is to be developed.

This article is structured as it follows. The 2nd section introduces the state of the art of Multi-Agent systems, of the business organization and of the specialization. The 3rd section introduces the concept of community and the application of the business specialization to the Multi-Agent systems, specially to the community of Agents. The 4th section deals with SCODA (*Distributed and Specialized Agent Communities*) as the framework architecture presented in this article. The 5th section shows the application of SCODA to a specific case that will be analyzed and finally in the 6th section we will show the conclusions.

8.2. STATE OF THE ART

The Multi-Agent systems correspond with distributed powerful computer models, based on software entities that cooperate among them (*Amigoni y Fugini 2007*)**.** The MAS are usually characterized according their internal behavior and external interactions among agents (*Wooldridge, 2002*). The main features to characterize the internal behaviour of the Agents are: their kind of reasoning and working regarding the way they behave. For example, they can be reactive, based on models, based on goals; their level of adaptability; their perception and characterization of the environment where they are, including their computer infrastructure and their relations to other agents; and the level of autonomy in the actions they carry out (*Posland, 2007*).

According to their own features, the agents can be classified into *reactive*, when they carry out easy tasks responding to external events without being able of carrying out reasoning through mechanisms of representation of knowledge; and *cognitive* or *deliberative agents* that have mechanisms of reasoning and explicit representation of knowledge, therefore the cognitive agents can perceive stimuli of their environment, they reason a planning and they work according to that planning.

In the deliberative architecture, BDI *(***Belief, Desire, Intention***)*, the agents that implement it, have some mental states associated, such as **Beliefs**, **Desires** and **Intentions** (*Bratman, 1988; Rao y Georgeff, 1995*). In (*Rao y Georgeff, 1995*).

In addition, in order to reach the aims, it is necessary to make plans (*Corchado, et al., 2008; Oijen y Dignum, 2011*) that correspond with the sequences of actions

needed to fulfill that goal. These plans, linked to the fulfillment of that goal, make up the intentions of the agent (*Corchado, 2005; Oijen y Dignum, 2011*).

The deliberative agents BDI are shaped through an abstract structure, based on the logic of situations (possible worlds), called *temporal tree*, with multiple futures and only one past (*Rao y Georgeff, 1991*). Each node of this tree corresponds with a situation, and the branches correspond with the options of the agent in a specific time. For each situation, several new situations are defined, this situations can be reached by the agent according to the beliefs (possible situations), the desires (situations that are hoped to reach) and the intentions (situations that are tried to reach) (*Rao, et al., 1995)*. For this model, it is necessary to have some relations between the beliefs, the desires and the intentions of the agent (*Corchado, 2005; Koster et al., 2012; Oijen et al., 2011*):

- **Compatibility between beliefs and aims:** if the agent adopts the desire of reaching an aim, it must believe that the aim is true.
- **Compatibility between aims and intentions**: before adopting an intention as such, an agent must adopt it as a desire.
- **The intentions lead to actions:** if one of the intentions is a simple action, the agent carries it out.
- **Relation between beliefs and aims:** the agent knows its aims and desires.
- **There are not infinite delays:** when an agent adopts an intention, it keeps on having it until a specific moment in future.

Based on the way the agent's intentions lead or influence its decisions about future intentions, we can identify different types of agents (*Corchado, 2005*):

- **Blind:** the agent keeps its intentions until it is sure it has achieved them, therefore he will reject those beliefs and desires that go against them.
- **Firm**: the agent keeps its intention as long as it believes it can achieve them.
- **Impartial:** the agent keeps its intentions as long as they correspond with its desires.

The deliberative BDI has been used in a very successful way in different developments (Corchado, *et al.*, 2008, 2010; Bajo, *et al.*, 2009, 2010; Tapia, *et al.*, 2006; 2008; 2009; Rodriguez *et al.*, 2011; Zato *et al.*, 2012) . This type of agents provide solutions in dynamic environments and this makes them suitable when developing systems that need some stability based on the global aim of the system. This is why the BDI agents are really important for the development of the SCODA architecture.

A Multi-Agent system is made up a group of agents that work as a kind of organization. The organization is used to describe that group of agents coordinated through a number of behaviour patterns as well as the establishment of some specific roles to achieve the aims of the system.

Next, we will mention different authors considered experts at this field, and their different perspectives regarding the term “organization”.

For L. Gasser (*Gasser, 2001*), the concept of organization consist on a very structured system that has activity, knowledge, culture and history patterns as well as different and complementary abilities of any particular agent. The organizations exist at a completely independent level of the individual agents that make them, these can be replaceable and they can also occupy a region of the time-space, symbolic-space etc…
This is why an organization of agents provides a working setting where there is an interactive activity of the coordinated agents by means of a number of roles, expectations and rules. In (Zambonelli *et al.,* 2003) the organizations of the agents are understood as a group of roles related one another that interact among them in a systematic and institutionalized way.

J. Feber believes that the organization of the agents provides a good way to divide the system in groups that constitute the unit of the agents interaction. He also bases this organization on two aspects: structural and dynamic (Ferber *et al.,* 2003). The structural aspect represents the static part of the organization, that is, everything that stays when the different components join or leave the organization. He also defines the organizing units and its relations as well as the rolls and rules needed to carry out the aims of the organization. The dynamic aspect takes into account the models of interaction that have been defined for each role, describing the ways to enter and leave the organization as well as the duties and obligations required for every role and its assignation to the agents.

On the other hand R. Snyder and D. MacKenzie (*Snyder and MacKenzie, 2004*) highlight that the ability that the agents need to have to organize themselves in abstract groups called *communities*, is a powerful tool to structure the organizations of the agents in Multi-Agent systems with a large number of agents. These communities interact among them and with their environment in a kind of organizations called society, which is focused on the achievement of global aims.

An approach of the organizational of systems takes place in (*Esteva et al., 2001;2004*), through the assimilation of the functioning of human institutions. This approach supports the mediation of electronic institutions that regularize the Agents’ behaviour, therefore the interactions among them must be in accordance with some rules defined by the institution.

In (*Pavón et al., 2003; Pavón et al., 2005*) the organizations describes the environment in which the agents, resources, tasks and aims coexist, and this organization is defined by the social structure, functionality and relations. As for the structure, the organization is considered as a group of entities whose association is carried out through the relations of legacy and aggregation and following this pattern that some social relations as well as work flows are determined. The organization structures the agents in groups that can be formed by agents, roles, resources and

applications being their assignment an organizing purpose that makes its coordination easier.

Another expert worth mentioning is V. Dignum, who says that the organization has global aims and the agents have individual aims that exist independently of the agents (*Dignum and Dignum, 2006*). The roles of the agents, represent positions in the organization that have the responsibility of concluding some of the final aims of the organization thanks to a number of established interaction rules. In this case, the agents can or can't take several system roles.

In (*Hubner et al., 2006*) the organization is presented as a group of behaviour restrictions imposed by a group of agents to the agents in order to control their autonomy and achieve the global aims of the organization in an easy way.

In (*Garijo et al., 2001; Gómez-Sanz et al., 2006*), the organization is considered as something in which the types of agents and resources are identified. However, those agents and resources can be destroyed as well as created during the execution time, depending on the necessities of the system. Bearing in mind the approaches mentioned above, we can get some characteristics that all the organizations of agents have in common:

- An organization of agents is made up of agents, roles and the coordination and interaction rules among those roles.
- The organization has a common and global aim that does not depend on the individual aims of the agents.
- There is a division of tasks through the assignment of roles to the agents, so that they can be specialized always focusing on the consecution of the organization global aims.
- The organization divides the system into groups thanks to the departmentalization. These groups will be the unit of interaction among the agents.
- It also defines a number of limits regarding the relevance of the agents to the organization, their rules of interaction, its functionality and the services that if provides.
- The members joining or leaving the organization determines its capacity of dynamism, being able to change their roles depending on the aims of the organization.

In relation to the organization theory, It is worth mentioning H. Nicklisch, an expert that develops a theory based on the concept of *community* (*Nicklisch, 1928*). The concept of community is different from the concept of organization because the former is more natural and spontaneous. The concept of *community* is understood as something alive, dynamic, and organic that arises due to habits and customs settled in the tradition. Nicklisch thinks that the community is supported by the common interests of a work group, and it is necessary to establish a contractual and congestion system to carry out its regulation (Nicklisch, 1932)

According to Nicklisch, an organization with the shape of a community must have the following characteristics: Unity and articulation: a community must have the ability to decide, to draw its members together, and to communicate and coordinate in a fluent way; Division of working tasks and groups: this means a specialization of tasks that leads to a better production; Economic laws: it demands the maximum efficiency by means of a established aim; Group formation to establish a working methodology and the relations among its members; Group maintenance: the necessity of cohesion and a sense of belonging to the group so that the community can not break up.

Nicklisch thinks that the social value of the organizations is the most important value of all, making its members the most important things of the organization, assuming that these members are able to consider themselves members of the organization, this is why they will put their strengths together to achieve the aim of their organization. (*Nicklisch, 1920; Larsen 2000*).

In works by (*Glinton et al., 2010; Parachuri et al., 2010*) the term *community* is used to call a group of heterogeneous agents. (*Snyder and MacKenzie, 2004*) gives a definition of the Cougaar Agents Community.

Cougaar is an architecture based on the Java programming language to build applications based on systems of agents that are part of a big group of agents. (*BBN Technologies, 2004*). The global group of agents of this architecture that interact to achieve a global aim is called *society*. A Cougaar Community provides a way of organization of societies in different groups of agents or communities, that is, a *community* can be formed by agents that are members of the society and by other communities which are defined in a static way in a designing time. (*Snyder and MacKenzie, 2004*). This process is transparent for the agents because the communities infrastructure puts the corresponding agents in those communities at the beginning. Another possibility is the creation of dynamic communities, in which case it is necessary for a responsible enough agent to do it in an explicit way. In Cougaar Communities, the agents that are part of them can have several roles which represent the functionality of the agents in the community. The control of the agents in the Cougaar Communities is made by an agent appointed to it. However, if the community is created during the execution time, the agent in charge of the creation of the mentioned community is responsible for the agents that form the community as well as its correct functioning. (Snyder and MacKenzie, 2004)

Finally, some characteristics of the Cougaar communities can be summarized as it follows (*Snyder and MacKenzie, 2004*):

- The interaction of the members is multidirectional
- An agent of a community can have several roles.
- The community permits the creation, elimination of dynamic communities as well as the joining, the leaving and the modification of

the agents of the community (modification understood as the different roles that an agent can have).
- The communities are able to work in spread environments
- The communities have coordination mechanisms among agents
- The communities are scalable to solve difficult problems.
- Every community has a hierarchy.
- In these communities, the nesting of other communities can take place.

Bearing the definition of Cougaar communities as well as their characteristics in mind, we will present the definition of ***Intelligent Communities*** (IC). These communities adopt much defined characteristics in other type of organizations of agents besides the Cougaar ones, and they take into account key concepts arising from different organization theories mentioned in this chapter.

8.3. INTELLIGENT COMMUNITIES

The intelligent communities consist of a small group of cooperative agents that work together to achieve a common aim. According to Nicklishc's theory, an agent which is a member of a community must consider itself as such, that is, it has to place the good of the community before its own. This is the reason why there must be an explicit representation of the community aims, beliefs and plans. In the F**igure 8.1** we can see the graphic representation of an agent community.

The Intelligent communities are made up of a small number of agents so that the communication among the agents can be multidirectional instead of representing an additional computer effort. Its base line is the organizational structures of the teams, that is, a group of agents can deal with more difficult problems than those which a single agent can deal with, allowing redundancy aspects, better adaptation to the environment and making the most of the resources. The decision making is centralized regarding the community global decisions, and it will be autonomous for each agent in the tasks they carry out individually, in the community's benefit. This is why the tasks are formalized providing the agents with a tool for the individual decisions making.

The role taken by the agents in an Intelligent Community must be static, that is, the agents of a community take a specific role without having the possibility of changing it, this makes them specialized. The global aim of the community must be divided into sub-aims. That will be assigned to every agent of the community. In case a specific functionality is needed, a new agent will be allowed to join the community as well as to take the role needed to have that functionality. The specialization also exist at community level, therefore the community aims have to be included in the same context (being the concept of *context* understood as the group of aims focused to achieve the same global aim).

The dynamicity of the Intelligent Community, regarding the joining or leaving of its members as well as its internal association, allows the internal grouping of its members in small work groups so as to coordinate similar tasks. It also tries to settle the agents as well as teams without allowing other agents to enter the community. It is important to point out that the scalability is permitted in case of serious global problems, working as an open system (*González-Palacios and Luck, 2007*). In addition, we must bear in mind that when a member of the community does not work properly or fails to fulfill the rules of the community can be expelled and replaced by a similar member.

In an Intelligent Community there will be two levels of hierarchy regarding the control activities of the community. One of the agents will control the others based on a series of community rules that will guarantee the proper functioning of the community. This is because (although Nicklisch considers every member of the community to be good), in a community of agents we have to take into account the individual goals as well as the deviations of the community global aims.

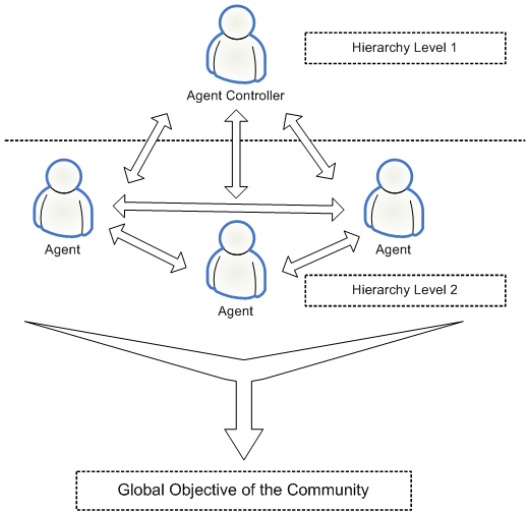


Figure 8.1 Representation of an Intelligent Community (IC)

Finally, Intelligent Communities must possess the ability to function in a distributed way, and provided services should not be embedded in each member, but must be executed in a distributed manner in order to release computationally the Community structure and the member agents.

8.3.1 Specialization

Specialization is a feature that gives a particularity to an individual belonging to a group. This specialization is observed within organizations as a mechanism to achieve greater efficiency in the objectives pursued by it. Specialization has been discussed in depth by (Theraulaz et al. 1991a; Theraulaz et al. 1991b; Murcia et al.,

1997 Li, et al., 2002 Li, et al. 2004; Lei, et al., 2007, Okamoto, et al., 2008) among others. This specialization aims to provide greater efficiency for the multi-agent system, such that there is an improvement in the achievement of its overall objectives. H. Mintzberg (Mintzberg, 1989) defines the specialization of tasks as the number of different tasks that develops in a job and how often are repeated, and the power of decision that is exercised on his design. Specialization of tasks, in turn, is divided into horizontal task specialization, and vertical task specialization. This classification is taken from (Mintzberg, 1989).

The horizontal task specialization has a close relationship with the division of labor and denotes the number of tasks to perform in a particular site job and its main aim to increase productivity. This specialization regulates, the amplitude of the job and can be high, when the job, must develop a small number of repetitive tasks, and low, if the job has to perform many tasks repetitive. Moreover, the Vertical specialization of tasks refers to the differentiation between the execution of the tasks, and planning and control over them. In this case, can also be high when a job executes only tasks that have been planned by someone else, and low when tasks are scheduled to run by someone else, but on its own initiative.

The relationship between these two types of specialization is very close, since, a job in which there is a high horizontal specialization also requires a high vertical specialization because the person performing a particular task may lose the global view of objectives, and therefore it is necessary that another person, who has this vision, must plan, organize and control such work (Mintzberg, 1989).

Thus, according to the above, we can measure specialization of tasks as follows:

$$\begin{cases} N_T \neq 0 \;\; number\ of\ task\ performed\ by\ a\ person \\ P = 1\ particular\ job \end{cases} \tag{8.1}$$

$$E_T = \frac{P}{N_T} \tag{8.2}$$

Where E_T is the degree of specialization of task in a given job. Defining E_{TH} as the degree of horizontal specialization task, then:

$$\begin{cases} Si\ E_T > 0.2, \rightarrow \;\; E_{TH}\ is\ very\ high \\ Si\ 0.2 > E_T \geq 0.1, \rightarrow \;\; E_{TH}\ is\ high \\ Si\ 0.1 > E_T \geq 0.05, \rightarrow \;\; E_{TH}\ is\ low \\ Si\ E_T < 0.05, \rightarrow \;\; E_{TH}\ is\ very\ low \end{cases} \tag{8.3}$$

You may encounter a case in which several tasks are assigned certain number of people, which would imply that P > 1. In this case $E_T >> 0.2$, implying that E_{TH} still be very high.

N_{TP} is the number of tasks performed by one person and, that are planned and are controlled by another person, and E_{TV} the degree of vertical specialization in a particular job. We have that:

$$\begin{cases} N_{TP} \neq 0; \\ if\ N_{TP} = 0, we\ consider\square\ no\ specialization \\ N_{TP} \leq N_T \end{cases} \quad (8.4)$$

$$E_{TV} = \frac{P}{N_{TP}} \quad (8.5)$$

We can obtain the following results:

$$\begin{cases} If\ E_{TV} \cong E_T \ \rightarrow\ E_{TV}\ is\ very\ high \\ If\ E_{TV} > E_T, \rightarrow\ E_{TV}\ is\ high \\ If\ E_{TV} \gg E_T \ \rightarrow\ E_{TV}\ is\ low \end{cases} \quad (8.6)$$

Through these equations we can measure the degree of specialization of tasks in a given job. Furthermore, it has taken into consideration the peculiarity that several jobs perform the same task, which would imply a very high specialization of tasks as productivity grow. In this case it must be noted that not all tasks in a job allow several people to do them parallel. These thresholds are given from the observation of different jobs, depending on the tasks to be performed on them. To do such tasks, are specified and quantified in a way that we can measure somehow the specialization. The definition of these thresholds is based on the observation of different jobs, from workers on a production line, to specialists. Moreover is also taken into account the works of (Rushing, 1967, Samuel and Mannheim, 1970; Reimann, 1974) where the results obtained are taken and assimilated to our particular problem.

The organizational specialization approach can be seen as a unit, through enterprise networks. In literature, the concept of enterprise network can be found in works by authors such as (Sonquist and Koenig, 1975; Galaskiewicz, 1979, Burt, 1980, Walker et al., 1997; Lazer, 2003; Borgatti and Foster, 2003; Tracey and Clark, 2003; Johansson and Quigley, 2004; Pretoria 2004; Pöyhönen and Smedlund 2004; Ibarra et al., 2005; Eraydin and Köroglu Armatli-2005; Cabus and Vanhaverbeke, 2006) mentioned under (Becerra, 2008), for which a set of groups, institutions or organizations interact with each other to get favorable results both individual units or to the network as a whole.

Thus, an enterprise network can be regarded as an association of companies working together so that there is a specialized complementation each other, with the aim of resolving situations, that individually could not resolve, in the most efficient way possible.

According to (Lopez, 2003) the objectives of an enterprise network are:

• Increase the competitiveness and profitability of the network.
• Induce the specialization of firms in some of the different stages of the production process.
• Consolidate the market presence of the companies that make up the network.
• Facilitating access to services companies, that they are individually inaccessible.

C. Lopez, considered essential the number of members of a corporate network so that there is efficiency from operations, i.e. must consist of a limited number of companies, no more than 15 or 20, and well defined (Lopez, 2003) . In enterprise networks, there are two types of modality in terms of cooperation concerns: horizontal networks and vertical networks business enterprise (Becerra, 2008) as shown in **Figure 8.2.**

Horizontal networks are alliances between groups of companies offering the same service or product. The cooperation between these companies is given in a series of activities; however they compete in the same market. An example of this type of network would in two companies which manufactured tires. Clearly, the association of businesses from raw material suppliers will benefit both in terms of their size and therefore its price.

Moreover, vertical networks are alliances between different companies partnering to achieve competitive advantage that would be impossible individually. So we will have an example in several companies engaged in manufacturing computer components. The vertical connection of these companies can carry out marketing complete and assembled computers, which opens a new market that benefits all.

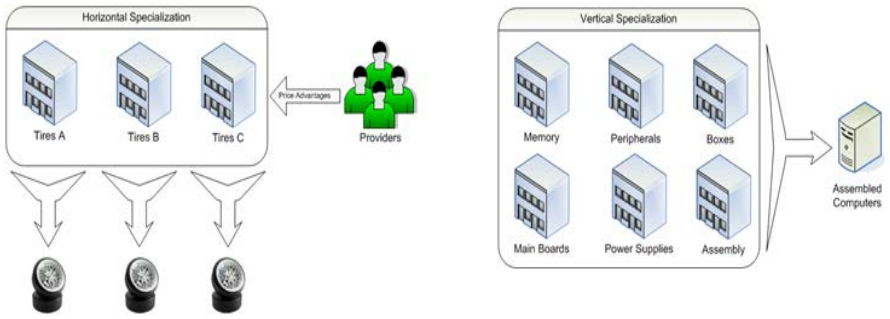


Figure 8.2 Vertical and Horizontal Specialization in Enterprise Networks

As in the case of specialization of tasks, we can also measure the degree of specialization within an organization and within enterprise network, the degree of vertical and horizontal cooperation reached by their companies. N_{TE} he number of task forces to work in an organization. We understand the task group, the different tasks such as administration, marketing, etc., plus those on different production tasks. And N_E, the number of employees. Then:

$$E_E = \frac{N_{TE}}{N_E} \tag{8.7}$$

Where E_E is the degree of enterprise specialization. E_E can take the following values:

$$\begin{cases} Si\ E_E > 1, \rightarrow\ \ E_E\ is\ very\ low \\ Si\ 1 \geq E_E > 0.75, \rightarrow\ \ E_E\ is\ low \\ Si\ E_E \leq 0.75, \rightarrow\ \ E_E\ is\ high \\ Si\ E_E \cong 0, \rightarrow\ \ E_E\ is\ very\ high \end{cases} \tag{8.8}$$

Now, having defined the degree enterprise specialization, we somehow wish to quantify the degree of cooperation, both horizontally or vertically on a group of companies that form an enterprise network. We first define the following indicators. G_{CV} is the degree of vertical cooperation and the degree of G_{CH} horizontal cooperation within the enterprise network, depending on the terms defined above. Considering the above in (Lopez, 2003), enterprise networks must not exceed a certain number of members up to a threshold of 15 or 20.

$$\begin{cases} N_{EM} > 2, \text{is the number of companies within the enterprise network} \\ N_{SP} > 0, \text{is the number of products or services produced by the organization} \end{cases} \tag{8.9}$$

$$G_C = \frac{N_{SP}}{N_{EM}} \tag{8.10}$$

Where $G_C > 0$ is the degree of cooperation, through which we can obtain the degree of vertical and horizontal cooperation as follows:

$$\begin{cases} If\ G_C < 1, \rightarrow\ \ G_{CH}\ is\ high \\ If\ G_C = 1, \rightarrow\ \ G_{CH}\ is\ medium \\ If\ G_C > 1, \rightarrow\ \ G_{CH}\ is\ low \end{cases} \tag{8.11}$$

$$\begin{cases} If\ G_C > 1, \rightarrow\ \ G_{CV}\ is\ high \\ If\ G_C = 1, \rightarrow\ \ G_{CV}\ is\ medium \\ If\ G_C < 1, \rightarrow\ \ G_{CV}\ is\ low \end{cases} \tag{8.12}$$

As seen in the equations described in (8.11) and (8.12), the degree of horizontal and vertical cooperation are reversed. Within an enterprise network, as more services are offered by each of its members to the corporate network, the greater the vertical specialization. Conversely, if there is only one product or service that are developed by a group of companies, increases the degree of horizontal specialization. As in the specialization of tasks defined before, the threshold values come from the

observation of different companies and taking into account the conclusions of the works of (Rushing, W. 1967, Samuel, Y. y Mannheim, B. 1970; Reimann, B. 1974).

8.4. SCODA

Individuals or organizations are generally accustomed to generating solutions to solve problems that can occur at any given moment. Many mechanisms to solve these problems have already been identified (Jensen, 1992). In the field of computational systems, one such mechanism is Object Oriented technology (Lieberherr, 1996; Elrad *et al.,* 2001; Rashid *et al.,* 2003; Kiczales *et al.,* 1997), a model in which objects are encapsulated independently and can be reused in different developments to achieve various objectives. An approach focused on the use of multiagent systems is precisely one of the objectives of this study, which seeks an Intelligent Community based architecture that can be executed independently and engage in atomic collaboration with other Intelligent Communities; that is, as closed organizational units whose objectives are complementary.

SCODA (Distributed and Specialized Agent Communities) (Román *et al.* 2011) is a new architecture focused on the development of Multiagent systems and based on five basic principles: standardization, specialization, ease of implementation, reusability, and distributed computing. SCODA integrates one or more highly specialized Intelligent Communities, which will be referred to as **Specialized Intelligent Communities** (SIC) (Roman *et al.* 2011). Each SIC can function as an independent and specialized multiagent system that offers distributed services, and whose aim is to achieve a global objective. This structure allows different SIC to collaborate toward achieving an objective that they would otherwise, by working independently, be unable to accomplish. This philosophy is based on "*Business Networks*" (Sonquist y Koenig, 1975; Galaskiewicz, 1979; Burt, 1980; Walker *et al.,* 1997; Lazer, 2003; Borgatti y Foster, 2003; Tracey y Clark, 2003; Johansson y Quigley, 2004; Viedma, 2004; Pöyhönen y Smedlund, 2004; Ibarra *et al.,* 2005; Eraydin y Armatli-Köroglu, 2005; Cabus y Vanhaverbeke, 2006) through which a set of groups, institutions or organizations interact to obtain favorable results for each individual unit as well as the network they constitute. It is precisely because of this focus that SCODA is based on the concept of the Intelligent Community as an organizational unit with the characteristics needed to fully develop along these lines. SCODA is an architecture that can be implemented in any multiagent system platform, and supports BDI agents (Bratman, 1988; Rao and Georgeff, 1995). In order to fully develop and execute this architecture, the Jadex platform (Pokahr *et al.,* 2003; 2007) was selected, as it is a reasoning engine and can be executed independently.

The SCODA architecture is based on five principles whose combined use will result in greater efficiency of the multiagent systems it develops. **Table 8.1** identifies and describes the five principles:

Table 8.1 Principles on which SCODA is Based

Príncipe	Descripción
Standardization	The different Specialized Intelligent Communities have a common structure that is independent from the objective pursued in the multiagent system that will be implemented.
Specialization	This principle is based on human specialization and "Business Networks" (Sonquist y Koenig, 1975; Galaskiewicz, 1979; Burt, 1980; Walker *et al.*, 1997; Lazer, 2003; Borgatti y Foster, 2003; Tracey y Clark, 2003; Johansson y Quigley, 2004; Viedma, 2004; Pöyhönen y Smedlund, 2004; Ibarra *et al.*, 2005; Eraydin y Armatli-Köroglu, 2005; Cabus y Vanhaverbeke, 2006) which enables cooperation among the Specialized Intelligent Communities.
Ease of Implementation	Within the framework of the SCODA architecture, the multiagent systems should be easy to implement. This can be achieved since the structure of the Specialized Intelligent Communities is standard and the services are what determine what will be programmed.
Reusability	As each Specialized Intelligent Community within SCODA is adopted as an independent multiagent system, the reusability of these Specialized Intelligent Communities should be feasible in any SCODA-based development.
Distributed Computing	The required services are not provided directly by the agents in the Specialized Intelligent Communities; instead a distributed execution reduces the computational load associated with the agents and prevents variations in the structure of the architecture.

The modular structure of SCODA allows its agents to administer and coordinate the architecture and its functionalities. As shown in **Figure 8.3**, SCODA is defined in six basic modules: External Applications, Communication Protocol, Control, Agent Platform, Specialized Intelligent Communities, and Community Services.

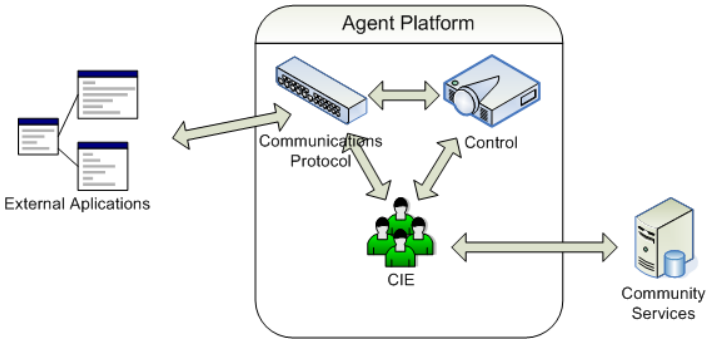


Figure 8.3 Structural Diagram of SCODA

The **External Applications** consists of the programs and users that use SCODA and request the services it offers. The **Communication Protocol** is in charge of processing the requests made by the External Applications and soliciting a

response from the Specialized Intelligent Communities. The **Control Module** oversees the coordination and the functionalities provided by the architecture, and adheres to the tolerance policy for operational errors. The **Specialized Intelligent Communities** are the core of SCODA, and what ensure the deliberate and optimal effectiveness of the requests and responses. The **Community Services** are executed in a distributed manner; this is where the processing capacity of each **Specialized Intelligent Community** is located. Finally, the Agent Platform represents the environment in which SCODA is executed. It is composed of the agents that form the architecture and the Specialized Intelligent Communities. These six blocks compose the structure of SCODA and are defined below.

The agents that compose SCODA follow the deliberative BDI model (Bratman, 1988; Rao and Georgeff, 1995; Koster *et al.*, 2012; Oijen *et al.*, 2011), and the services offered by the Specialized Intelligent Communities are managed by these same agents. One of the agents, the *PlannerAgent*, is responsible for determining which services are most optimal and which parameters are required so that the solution can be adjusted to the needs of the user or the external application that has made the request. Another characteristic applied to all SCODA agents is their ability to use reasoning mechanisms and learning techniques, because they are deliberative BDI agents, to manage and coordinate the functionalities according to the specific context in which they are executed.

External Applications: Represent all of the programs that can use the functionalities provided by the multiagent system implemented over SCODA. These applications are dynamic and react differently according to specific situations, such as managing a personal agenda. They can be executed locally or remotely, even from mobile devices with limited processing capabilities, since the tasks that require a high computational load are performed in a distributed manner by the SCODA agents through the community services.

Communication Protocol: SCODA implements a communication protocol based on REST (REpresentational State Transfer) (Fielding, 2000). REST permits external applications to communicate directly with the services offered by the Specialized Intelligent Communities. The protocol is completely independent from the programming languages used, and is based on requests made over HTTP (Hyper Text Transfer Protocol) (RFC2616, 1999). An HTTPRequest is sent through the external applications to specify the required service, and follows the format shown in **Table 8.2.**

Table 8.2 Format for HTTP Request in SCODA

Used Methods	Description
GET/POST	*Request = Simple-Request	Full-Request* *Simple-Request = "GET"|"POST" SP Request-URI CRLF* *Full-Request = Request-Line*

	**(General-Header | Request-Header | Entity-Header)* *CRLF* *[Entity-Body]*

This request identifies all of the parameters needed to complete the required tasks. All external requests follow the same communication guidelines and protocol, while internal communication within the agent platform follows the guidelines specific to the platform on which SCODA is implemented. In the case of Jadex (Pokahr*et al.,* 2003; 2007), FIPA Agent Communication Language (ACL) (FIPA, 2005) is used. Finally, if an agent invokes a service from its community, the request creates a new thread associated with a socket and an open communication is maintained until either the task has been completed and the result sent to the corresponding Specialized Intelligent Community, or an error has been produced in the execution of the service, in which case the SIC is also informed. The format for responding to the external application that requested the services is *HTTPResponse*, as shown below:

Table 8.3 Format of HTTP Response in SCODA

Description
Response = Simple-Response | Full-Response *Simple-Response = [Entity-Body]* *Full-Response = Status-Line* **(General-Header | Response-Header | Entity-Header)* *CRLF* *[Entity-Body]*

Within the body of the *HTTPResponse*, and in line with REST specifications, the response can be represented in either HTML, XML or JSON formats (Muehlen*et al.,* 2005).

The use of sockets when invoking a service makes it possible to process multiple requests simultaneously. When processing services that require a high computational load, the Specialized Intelligent Community is responsible for balancing the load by invoking another service that can handle the request. **Figure 8.4** shows an example of a service request in SCODA.

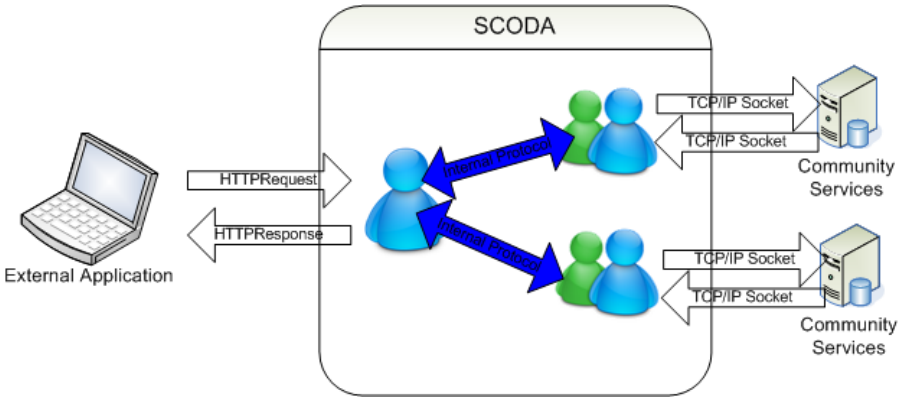


Figure 8.4 Example of Service Request in SCODA

Code 8.1 shows the communication phases between an external application and SCODA. The external application requests a service from SCODA using an *HTTPRequest*. SCODA receives the request and selects the Specialized Intelligent Community that can process the request. This community looks for the most convenient service for the request and selects the optimal parameters to execute the service. If there is no service that can adequately respond to the request, an error is generated; this also occurs if the execution of the services produces an error. The Specialized Intelligent Community notifies the response, and SCODA uses *HTTPResponse* to send the execution response to the external application. Internal communication in SCODA is carried out through ACL messages.

External communication phases in SCODA
1. Extern Applications → HTTP(Request-Service (service))
2. SCODA → HTTP-Request(Reception- Service (service))
3. SCODA → ACL (Select-Community(service))
4. While (Exist-Services)
5. COMMUNITY → Seek- service (service)
6. If (Service -Avalilable)
7. SCODA → TCP/IP (Community (Call- Service (service, parameters)))
8. SCODA → TCP/IP (Community (Send-Response))
9. End If
10. End While
11. If (Service –Run-OK)
12. COMMUNITY → ACL (Send- Response (service))
13. SCODA → HTTP-Response (Send - Response (service))
14. Else IF
15. COMMUNITY → ACL (Send - Response (error))
16. SCODA → HTTP-Response (Enviar-Respuesta(error))
17. Fin Si

Code 8.1. External communication phases with SCODA

Control Module: SCODA implements a control mechanism on its agents, services and any communication that takes place. With this module, which is implemented

by various agents, the architecture is ready to deal with application failures and solve them independently through a deliberate decision making process. If one of the active agents malfunctions or simply stops functioning during the execution process, the control module will be able to reset the agent to correct its operation. Additionally, if one of the services associated with any of the Specialized Intelligent Communities also malfunctions, the control module will be able to find a replica of the service that can respond to the given request. The module controls the communication that takes place among external applications, SCODA agents, and the services associated with the Specialized Intelligent Communities. Consequently, if any type of unusual communication is detected, the module will attempt to correct it by either restarting the communication or, if necessary, resetting all of the entities involved in the given communication process.

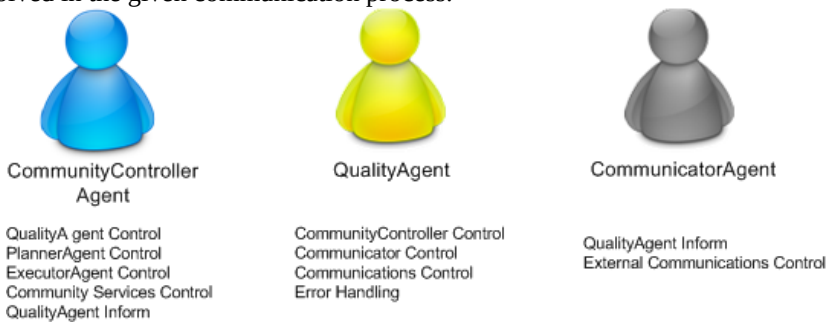


Figure 8.5 SCODA Agents in the Control Module

Figure 8.5 illustrates the role of each of the agents in the control module. The *CommunityControllerAgent* controls the *QualityAgent* as well as the team composed of the *PlannerAgent* and the *ExecutorAgent*. It also ensures that the services within its community function properly. All of the information flow controlled by this agent is transmitted to the *QualityAgent*, which in turn controls each *CommunityControllerAgent* in SCODA as well as the *CommunicatorAgent*. Additionally, it controls internal communication on the platform and manages the errors that may occur. Finally, the *CommunicatorAgent* controls external communication and informs the *QualityAgent* of the movements it oversees. All of the internal communication on the platform takes place through ACL messages, in line with FIPA (FIPA, 2005) specifications. **Figure 8.6** shows an example of the various agents in the control module and the actions they perform. An external application requests a service from SCODA. The *CommunicatorAgent* receives the request, selects the community best able to provide a response, and informs the *QualityAgent*. The *CommunicatorAgent* waits for the *QualityAgent* to confirm receipt of the information (*ACK message*). Once the confirmation has been received, it sends the request to the *CommunityControllerAgent*, which informs the *QualityAgent* of the creation of a work team. However, there is a problem in creating the work team, prompting the *QualityAgent* to reset the agents that form part of the team.

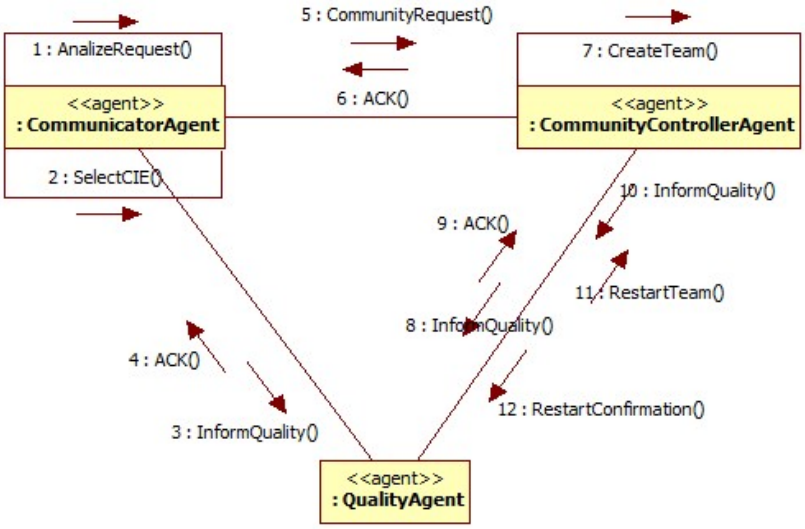


Figure 8.6 Diagram Showing how the Control Module functions

Specialized Intelligent Communities: Specialized Intelligent Communities are the essence of SCODA as they contain the ability to distribute and select work intelligently. They are composed of a controller agent (*CommunityController*) and a team that includes a planner agent and an executor agent (*PlannerAgent* and *ExecutorAgent*), which are instantiated in execution time when they are needed and released upon completing their task. The internal architecture of the community combined with its guiding philosophy with regard to instantiating and releasing requested agents make SCODA a distributed architecture with regard to its services, and efficient with regard to the management of internal resources for its execution platform. These Specialized Intelligent Communities function as autonomous systems. As a result, a SCODA-based implementation can be composed of one or various Specialized Intelligent Communities. It can use the services provided by these communities separately, or coordinate the use of various Specialized Intelligent Communities and the services they offer, allowing for a collaborative effort in solving greater problems.

Example of individual Specialized Intelligent Communities
9. *External Software → Request Service(translation)*
10. *Communicator→Recieve Petition(translation)*
11. *Communicator→ Select Community(translation)*
12. *TranslateCommunity→ Execute Service(translation);Send Response(translation)*
13. *External Software → Request Service(dictionary)*
14. *Communicator→ Recieve Petition(dictionary)*
15. *Communicator→ Select Community(dictionary)*
16. *DictionaryCommunity→ Execute Service (dictionary); Send Response (dictionary)*

Code 8.2. Example of executing individual Specialized Intelligent Communities

Figure 8.7 shows the Specialized Intelligent Communities working individually without any type of collaboration. There are N independent Specialized Intelligent Communities, each one associated with M services. The requests that enter SCODA are handled by the Specialized Intelligent Community best able to provide a response. As an example, **Code 8.2** easily illustrates the process that is taking place. There are two Specialized Intelligent Communities, one that provides translation services and another that provides dictionary services. External applications request unrelated translation and dictionary services from SCODA, and the corresponding Specialized Intelligent Communities respond independently.

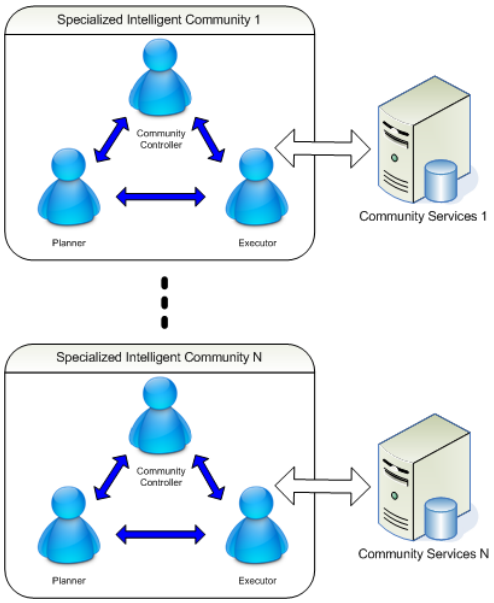


Figure 8.7 Specialized Intelligent Communities working independently

Figure 8.8 demonstrates how scaled collaboration between the Specialized Intelligent Communities provides results that neither community could have achieved individually. The same Specialized Intelligent Communities from the previous case are used, one that executes translation services and the other that provides dictionary services. In this case, an external application requests both a dictionary and translation service, implying that there should be a collaborative effort between both Specialized Intelligent Communities in order to process the request made by the external application.

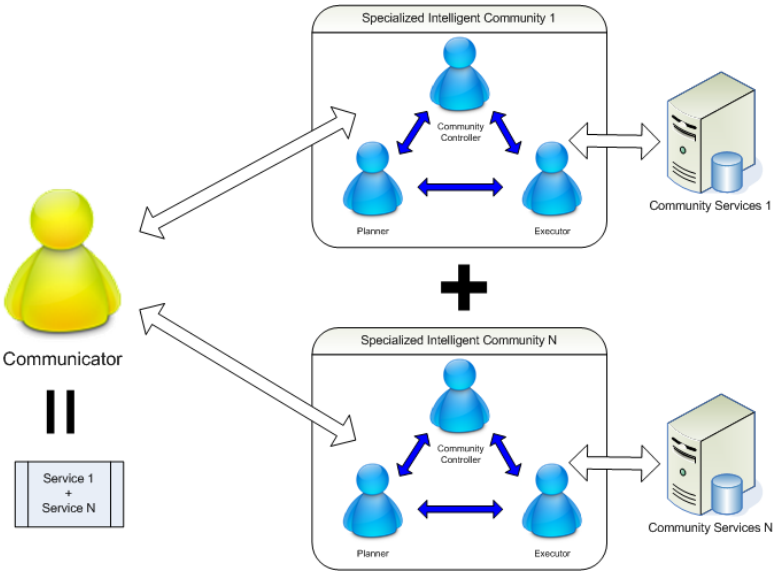


Figure 8.8 Scaled use of Specialized Intelligent Communities

Code 8.3 shows the detailed collaboration between the Specialized Intelligent Communities that offer translation and dictionary services. This is a scaled collaboration planned by the CommunicatorAgent, which can select the Specialized Intelligent Communities needed to achieve the global objective using a type of DirectoryFacilitator that stores the Specialized Intelligent Communities and their respective information for future use.

Example of collaborating Specialized Intelligent Communities
9. *External Application → Request Service(dictionary+translation)*
10. *Communicator →Response(dictionary+translation)*
11. *Communicator → Select Community(dictionary)*
12. *Community Dictionary → Run Service (dictionary);Send Response(dictionary)*
13. *Communicator →Response(dictionary)*
14. *Communicator → Select Community (translation, DictionaryResponse)*
15. *Community Translation → Run Service (translation, DictionaryResponse); Send Respuestas(translation, Respuesta dictionary)*
16. *Communicator → Enviar Respuesta (dictionary+translation)*

Code 8.3. Example of Specialized Intelligent Communities working in collaboration

Point 3 establishes the threshold values required so that an Intelligent Community can be considered Specialized. It is first necessary to estimate the indicators required to calculate the degree of business specialization at the Intelligent Community level, after which the degree of specialization for the tasks is calculated.

Table 8.4 Degree of business specialization for a Specialized Intelligent Community in SCODA

	Indicators	Value
Business Specialization	N_{TE}	1
	N_E	3
	E_E	0.33

Table 8.4 shows the indicator values for a Specialized Intelligent Community in SCODA. According to the composition and the number of general tasks performed by a community, the number of agents N_E is 3, which complies with the restriction of having a value equal to or less than 4. The only general task that the Specialized Intelligent Community in SCODA performs is to execute a service, which gives N_{TE} a value of 1. These values elicit a value of 0.33 in E_E, which falls within the established threshold needed for an Intelligent Community to be specialized at the organizational level.

Table 8.5 Degree of specialization for tasks in a Specialized Intelligent Community in SCODA

	Indicators	Value
CommunityController	N_T	7
	P	1
	E_T	0.14
Planner	N_T	3
	P	1
	E_T	0.33
Executor	N_T	2
	P	1
	E_T	0.5

Table 8.5 shows the values for the specialized tasks that exist in a Specialized Intelligent Community. In order to carry out this task, each of the agents is typified as a job that performs a number of repetitive tasks. Using the threshold indicators defined in section 3.3, we can observe that the *PlannerAgent* and the *ExecutorAgent* reach very high levels of specializations. This is to be expected since the agents limit themselves to executing a distributed service, while the *CommunityControllerAgent* reaches a high level of specialization by performing control and information tasks. However, all of the agents lie within the threshold of specialization. This implies that the degree of specialization for Specialized Intelligent Communities in SCODA is quite high, and that, consequently, this specialization results in a simplified development of these agents because of the reduced number of different tasks that must be carried out by each agent.

Community Services: These services represent the functionalities offered by the architecture. The community services are the core of the processing offered by the system. These systems can be accessed in a ubiquitous and distributed manner, thereby removing the computational load from the SCODA agents and creating a lighter architecture. The Community Services are in permanent active mode in order to receive requests from their respective Specialized Intelligent Community. They are designed for remote access through sockets, as a result of their easy

implementation and the positive results obtained in studies such as (Sunwook *et al.*, 2009; Balaji *et al.*, 2006; Douglas y Pai 2006). The services are organized by category according to their specialization; that is, the services related to a particular functionality are grouped together for their access by the specialized community that offers those services. This makes it possible to look for efficiency based on specialization within business networks (Sonquist y Koenig, 1975; Galaskiewicz, 1979; Burt, 1980; Walker *et al.*, 1997; Lazer, 2003; Borgatti y Foster, 2003; Tracey y Clark, 2003; Johansson y Quigley, 2004; Viedma, 2004; Pöyhönen y Smedlund, 2004; Ibarra *et al.*, 2005; Eraydin y Armatli-Köroglu, 2005; Cabus y Vanhaverbeke, 2006). Community Services, as well as other distributed services such as *Web Services* and *Architecture Oriented Services* (Papazoglou et al., 2007) must have publication and discovery services. They should also have an updated directory of services which can be accessed as needed (Leymann *et al.*, 2002; Papazoglou *et al.*, 2007), which is why SCODA has a flexible service directory from which the services can be invoked, modified, added and eliminated dynamically through the Specialized Intelligent Communities that provide the services. However, the insertion, modification or elimination of the Community of Services must be performed manually for security reasons (López *et al.*, 2006).

Table 8.6 Structure of the service directory for the Specialized Intelligent Communities in SCODA

Community	Invocation	Predicate	Host	Port	Extra	Description
[Forecast]	[demanda]	[type=1; urldatos=data1…]	[community1]	[8998]	[empty]	[predicción de la demanda de un producto]
[EOQ]	[eoq]	[type=3; urldatos=data1…]	[community3]	[8787]	[empty]	[gestión de inventarios]

Table 8.6 shows the structure of the service directory for the Specialized Intelligent Communities. This directory includes data about the name of the community, how the required service must be invoked, the predicate with the required parameters, the host where the service is located and its access port, and the description of the service.

Agent Platform: This platform is the core of SCODA. It integrates the agents that compose the Specialized Intelligent Communities, as well as the special control and communication management agents, each of which has special characteristics and specific behavior. Each agent has a series of functionalities that are determined according to the role they adopt. The architecture comprises BDI agents which it provides with the ability to act as system administrators, thus enabling the agents to support communication control and manage the behavior of other agents, including themselves.

One advantage provided by the development of the SCODA architecture is that external applications that use the services provided by SCODA can be programmed in any programming language, which increases the ability of the architecture to be

used with a multitude of devices. As shown in **Figure 8.8**, the services offered by the Specialized Intelligent Communities can be accessed by desktop devices, laptops, mobile phones, PDAs, etc. The different agents that compose the SCODA architecture, and their functionalities, are described below.

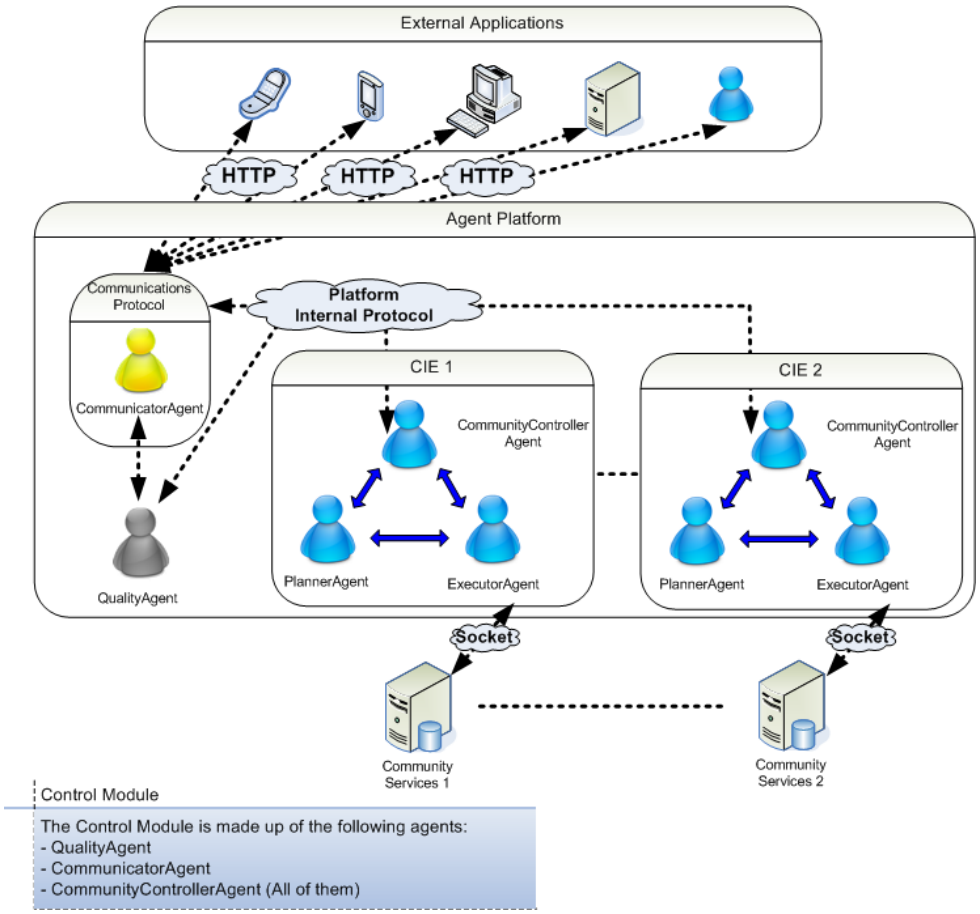


Figure 8.8 SCODA Architecture

Table 8.7 Description of the CommunicatorAgent

CommunicatorAgent	
Description	This agent is responsible for external communication with the platform. It receives requests from users, applications or external agents and transmits them to the Specialized Intelligent Communities for processing. It maintains a directory of the Specialized Intelligent Communities and a list of services provided by each. It is also responsible for providing responses as part of the external communication. This agent is continuously listening through a *Server socket*. If a request is received through HTTP, this agent looks for the appropriate community and sends the request through the platform's internal communication protocol. The thread waits to obtain a response that it can communicate to the entity that has made the request. At the same time, a response is sent to the *QualityAgent*, which performs the necessary operations to maintain a XXX and, if needed, solve any problems that might occur, such as a system crash in any of the agents in the community, including the *CommunicatorAgent* itself. This agent belongs to the Control Module and communicates with the *QualityAgent* to provide information regarding the communication that has taken place, whether sent or received, reviewing all messages received. It can also restart the *QualityAgent* if it detects any malfunction or inactivity.
Role	Communicator

Table 8.8 Description of the QualityAgent

QualityAgent	
Description	Responsible for controlling the operation and function of the other agents. Is in constant communication with the *CommunicatorAgent* and with each *CommunityController*. Is able to initiate the other agents in SCODA, acting on them in case of malfunction. Also calculates operation statistics for the operations that are carried out, which implies that all operational movements are redirected to this agent in execution time.
Role	Quality

Table 8.9 Description of the CommunityControllerAgents

CommunityControllerAgent	
Description	This agent controls the Specialized Intelligent Communities. Once it has been selected by the CommunicatorAgent to solve an eternal request, this agent creates a *Work Team* composed of a *PlannerAgent* and an *ExecutorAgent*, which are responsible for executing the response. This agent can create and disband the work team in execution time, which allows it to create as many instances of work teams as there are requests, and eliminate them once its goal has been reached or if it detects a malfunction in any of the agents from the *Work Team*. This agent also belongs to the Control Module and communicates with the *QualityAgent* to provide information regarding the services provided and the resolution of a request. It maintains a history of data related to the system's operation, and is also able to initialize the *QualityAgent* if it detects any malfunction or inactivity.
Role	Controller

Table 8.10 Description of the PlannerAgent

PlannerAgent	
Description	As one of the component of the Work Team, it is directly responsible for selecting

	and executing the appropriate service in response to an external request. This agent is created in execution time by the CommunityController when a Community Service needs to be invoked, and receives the request from the CommunityController in its community. It can select the specific service and its parameters among those provided by the community. This optimizes the execution of the service, which is then communicated to the ExecutorAgent, who is responsible for invoking it. The PlannerAgent has reasoning capabilities, which enables it to detect the state of the services at any given time, select a service not in use, and thus provide greater flexibility to the Specialized Intelligent Community.
Role	Planner

Table 8.11 Description of the ExecutorAgent

ExecutorAgent	
Description	This agent is in charge of invoking the appropriate service to resolve an external request. The invocation is done through sockets, which allows the services of each Specialized Intelligent Community to be distributed remotely, and also maintains the computational load within the machine where the services are located. The *ExecutorAgent* waits for a response and once received, communicates the response to inform that its work is complete.
Role	Executor

SCODA uses a series of reasoning mechanisms in its agents, thus providing the architecture with the ability to optimize the use resources and services. The control ability in SCODA is implemented in various agents and provides the architecture with a fault tolerance with regard to the agents' activities. This allows the *QualityAgent* to restart the *CommunicatorAgent* and any of the *CommunityControllers* within the architecture if it detects a malfunction or if the agent is simply not functioning at all. Likewise, the *CommunicatorAgent* as well as each of the *CommunityControllers* can instantiate a new *QualityAgent* if it is inactive. The agents are controlled periodically through messages that confirm their correct functioning, as shown in **Figure 8.9.**

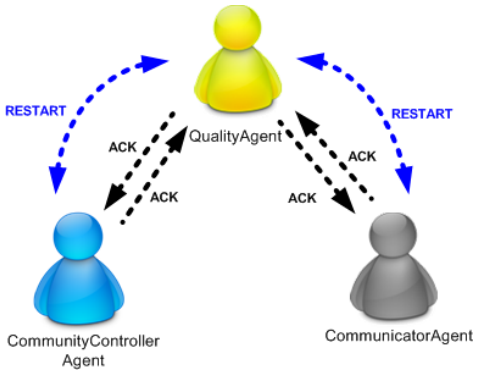


Figure 8.9 Control of Agents in SCODA

ACK (acknowledge) message are informative and require confirmation of receipt by the receiver. With this type of message, it is possible to determine are functioning normally within the system. The RESTART messages inform the receiver that it will be restarted. That is, the instance of the agent must first be killed before a new instance can be created.

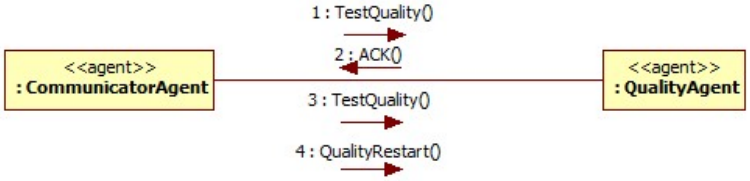


Figure 8.10 Example of how Agents are Controlled in SCODA

Figure 8.10 provides an example of how the CommunicatorAgent controls the *QualityAgent*. Messages 1 and 2 show the correct functioning of the *QualityAgent*, while messages 3 and 4 show how the *QualityAgent* fails to respond to the ACK send by the *CommunicatorAgent*, which then restarts the *QualityAgent* to avoid a potential problem. The *QualityAgent* can provide statistical control for the requests to the Specialized Intelligent Communities and each of the services they provide. These data are constantly updated and include the following:

Table 8.12 Operation Management in SCODA

Description of Operating Data in SCODA	
Maximum Time of Service Execution	These data indicate the maximum execution time for the service.
Total Requests	Correspond to the number of times that the corresponding service has been invoked.
Total Fails	Total number of fails that have been produced by invoking the corresponding service.
Accuracy Rate	Total number of correct responses after invoking the corresponding service.
No Parameters	Corresponds to the number of errors produced by the incorrect invocation of a service as a result of incorrect parameters.
FailTeam	Takes into account the errors that are produced as a result of an incident during the creation or execution of each Work Team that is instantiated to execute a service.
Inactive Service	Corresponds to the number of errors produced as a result of invoking an inactive service.
No ProperService	Indicates the number of errors produced from requesting a service from the platform where the service either does not exist or is not supported.
Problem in ServiceExecution	Refers to the errors that have resulted from the incidents during execution of service, whether from internal problems of the service itself, or an

	important delay in its execution.

This information allows *QualityAgent* to perform supervisory functions over the Specialized Intelligent Communities and service in SCODA, and to implement a high fault tolerance in the system. If an error occurs during the creation of the "Work Team", this agent orders the corresponding *CommunityControllerAgent* to execute the request again and to instantiate a new "Work Team". The services are also controlled by *QualityAgent*, so that if a specific service is inactive, this agent can restart it.

These data are very useful when determining the number of service replicas required for the system to perform optimally. Using the maximum execution time for the service and the number of requests for a particular service, the system can dynamically determine the optimal number of services within a specific amount of time by using the following formula (*8.13*):

$$N_R = \frac{N_S \times T_{MAX}}{P \times Th} \tag{8.13}$$

Where:
N_R, is the number of replicas required.
N_S, el the number of requests made in a period of time **P.**
T_{MAX}, is the maximum execution time for the service.
P, a specific period of time.
Th, the number of threads open simultaneously on a service.

Depending on the number of replicas that exist for a particular service, the *PlannerAgent* is able to select the service that supports the lowest computational load, thus optimizing system performance. The agent selects the service by referring to a directory, maintained by each Specialized Intelligent Community, that lists the replicas and the direction of each service.

8.5. CASE OF STUDY

The companies engaged in the distribution of frozen products, particularly those that sell these products to individuals, must take into account several factors to maximize profits such as the routes to follow, a sales forecast that will assure that there is no lack of product, and in turn allows you to make orders that offer some kind of discount and streamlining the intermediate supply procedures of every means of transport of the company, among other things.

To evaluate SCODA we count on the co-operation of Bahia Principe Congelados S.L. , which markets its products Monday through Friday, the workday begins at 8:30 for their workers. Each vendor is responsible for the loading of his truck and has to keep track of sales of each product reference so that they can replenish products that are running out or is about to do so. This is one of the issues

that will be addressed because there is a significant loss of time in checking how much must be replaced.

Another of the problems, referring to the product, is the stock level in the principal freezer, which has to supply each of the trucks in addition to the orders already placed with the office. To do this it must be borne in mind that all stock levels should be controlled from the central office but also a way to predict as accurately as possible the quantities of products that will leave the principal stock room or freezer in order to supply it. Finally, another important aspect to be addressed is the analysis of the routes that the trucks have to follow to sell their products in the homes of their clients. These trucks do a different route each day of the week, doing several hundreds of kilometers a day, which implies significant fuel consumption and minimizing this is more than necessary to help maximize the benefit of the company.

Depending on the methodology of this company and the large amount of products they sell it is essential to accurately estimate the quantities of the products that will be needed, and thus the necessary stock forecast to maintain their stock level of their main freezer. Knowing product demand implies predicting the amount that will be required to meet all the orders that could be made by their customers. There are also a number of factors that has an important influence on demand as shown in **Table 8.13**.

Table 8.13 Products that influence product demand

Factor	Description
Calendar	Depending on the time of year, month, and day of the week, the demand for this product varies significantly. What should be taken into account are bank holidays and holiday periods in when there could be an increase in domestic tourism and therefore hotel occupancy in the area. We should also look at the tendency of individuals to spend these holidays in various places and specifically looking for this product. Thus, timing is an important factor to consider in the estimates made on the demand for the product.
Geographic Zone	In the geographic area that this company serves there is great variety of products on offer. Each of the areas has a number of customers who usually consume the same products, which means that there must be a product forecast based on the day of the week and the area to visit as seven days will go by until they return to visit this area. This factor also affects the kilometers that each vendor has to drive, which implies that a minimization of these visits and some reorganizing will optimize fuel consumption, time, and as a result the performance of the company.
Temporal Space	This factor refers to the time the employee spends driving from one point of sale to another without commercializing any product. Minimizing this time is clearly related to the distances that has to be traveled by the worker to move between points of sale and implies that a reducing it would not only increase sales but have a positive impact on worker`s performance.

Moreover, with respect to orders made by entities such as hotels and retirement homes is directly influenced by the period of time required for the goods to get from the suppliers to the central warehouse or freezer. The company has placed at our disposal a truck with all the necessary data from the products it sells and the routes it

follows, data relating to the entities placing orders directly to the company, the data provided by their suppliers and a cold store in which to store the aforementioned products, in order not to interfere directly in the bulk of the organization and marketing of their products.

Given this data, the implementation of the multi-agent system will be as follows:

1. Carry out the implementation of the Specialized Intelligent Community for prediction, which has the ability to predict the demand for products for both the trucks doing their routes and orders made directly to the company.
2. Another Specialized Intelligent Community will perform stock control in order to optimize the order processing and stock control, based on stock management models.
3. Finally the implementation of a Specialized Intelligent Community to optimize the routes the trucks has to drive.

Through the SCODA architecture the SICs mentioned above will have the ability to communicate and function as a single system to optimize business processes

8.5.1 System Development

Specialized Intelligent Community of Forecast

The SIC of forecast has the ability to make predictions of demand for products using SVM (Support Vector Machines) implemented as an external service to SIC, thus following the principles that the SCODA architecture is based on. The SIC Inventories is able to perform the calculations necessary to manage product stock holdings and an adaptation of the theory of inventories in all its variants in which the demand is Determinable. Finally, SIC responsible for optimizing routes, generate a schedule that minimizes the associated cost of the trucks and maximize the benefits of the company.

The SIC of prediction has the task of performing the daily demand forecasting of products. For this we have made use of the library *LibSVM* (LibSVM, 2012) through which, it is possible to implement a regression model based on SVM, able to make such predictions. To make the training of the SVM are provided data relating to the demand for a series of products marketed by the company, both the retail and wholesale. Furthermore, these products are from different backgrounds, so it can be evaluated the temporal influence on the demand for them. The data provided by the company correspond for a period of time between *01/01/2009* and *30/09/2011*, and its characterization has been made of (Martin-Merino, 2006).

The choice of parameters for training the SVM has been performed via cross-validation (Martín-Merino, 2006; Velasquez et al., 2010) which has generated different models, because each evaluated product has a serial of temporary demand

different. While it is true that there are products with similar time series, which makes, they had been grouped to use the same model.

Once obtained the models for daily forecasts for each product, the prediction system is implemented as a set of services prediction SIC which are deployed on a different server, following the philosophy of SCODA, that will allow a distributed model that will download computationally the server where are running the SIC.
To evaluate these prediction models, it was decided to use three error measures which are complementary and allow us to have a vision of the predictions made and the comparative with reality.

1. **Mean Absolute Percentage Error (MAPE),** is the mean absolute between the prediction and the observed values expressed as a percentage of the observed values and is defined as:

$$MAPE = 100 \frac{\sum_{i=1}^{n} \left| \frac{L(i) - \hat{L}(i)}{L(i)} \right|}{n} \tag{8.14}$$

Where *L(i)* is de average demand of products, $\hat{L}(i)$ is the estimated demand y *n* the size of the data sample.

2. **Root Mean Square Error (RMSE):** Large errors are important because if these errors are by default, is induced a isolated loss selling of merchandise very important, for this reason is included this error since the quadratic function gives more weight to large errors. This error is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \left(\frac{L(i) - \hat{L}(i)}{L(i)} \right)^2}{n}} \tag{8.15}$$

3. **Maximum Error (ME):** This measure complements the previous and evaluates the maximum deviation between the predicted sales and the actual sales. Through it we can assess which type of product is which biggest mistake throws. Is mathematically defined as:

$$ME = \max_i \left| L(i) - \hat{L}(i) \right| \; i = 1, ...,n \tag{8.16}$$

Specialized Intelligent Community for Inventory Management

This community will allow control the existing stock in the principal camera, so that when it detects that a product has not enough stock, generates a purchase order to be sent by mail to the responsible of the camera. This order is illustrative, because is the responsible who confirm or modify the order. This SIC has ability to access a database, which stores the stock of products, and the discounts make providers depending on the amount of product and the time taken the product to arrive. So, this SIC alone is not able to estimate the future demand data, and until it amends the inventory database cannot act.

In our particular case, the SIC responsible for managing inventories, will use three models for orders: *EconomicOrderQuantityModel*, *EconomicOrderQuantityModel with discounts*, *EconomicOrderQuantityModel with Discount and Pending Orders*, which will be implemented as services SIC of inventory, and may be accessed by this community.

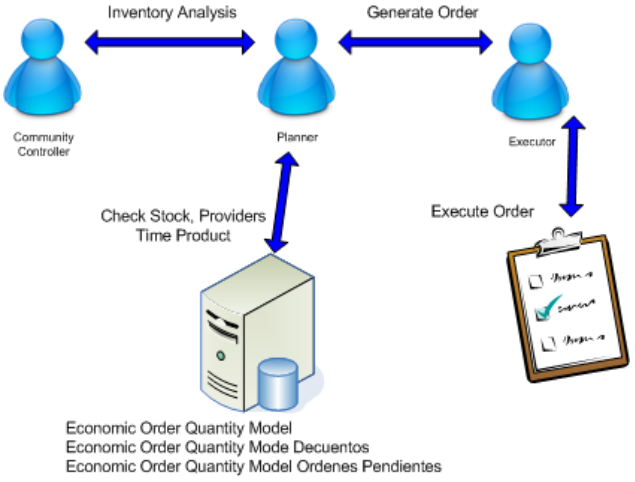


Figure 8.11 Representation of the Specialized Intelligent Community of Inventories

Figure 8.11 shows a diagram of the SIC responsible of inventories. The *PlannerAgent* receives the order of its *CommunityControllerAgent*, to run an inventory analysis. Then it access the database where it checks the stock of each product, then, it access the data of the providers where is the information about discounts and the time it takes to serve these orders of products, and which these data, is determined the point reorder of each of them dynamically, depending on the demand for them, to finally select the type of inventory model more in line and generate a purchase order. After obtaining the necessary parameters for the type of model, the *ExecutorAgent* through SIC services makes effective the request. The following is an example of two products with different demand features, over which the SIC of inventory will work. This will have to consider the weekly total product

demand (D), shipping costs (C_O), costs of order, which is calculated through the time spent and the hourly wage of the employee responsible (C_H), and the time it takes to arrive the products to the company(L_D).

Product A: D (445 unities); C_O (0 €); C_H ($\frac{6(€) \times 5(days)}{445(unities)}$); L_D (3 days)

Product B: D (169 unities); C_O (0 €); C_H ($\frac{3(€) \times 5(days)}{169(unities)}$); L_D (2 days)

The importance of determining the optimal order a product order is given among others by the quality of them as to the expiration date, and also for the money disbursed by the company, because if the orders are staggered and are of smaller magnitude (when there are no any quantity discounts), the company has more capacity of recuperation because exist daily sales, which gives a significant margin to face the payments.

Specialized Intelligent Community for the Optimization of Routes

This SIC is included in the case of study to show the efficiency of the architecture, its versatility and its ability to increase in size and functionality. This way it can check if changing the route along localities to visit, can get a saving in distance and thus, get fuel savings and working hours.

First are analyzed a number of classics problems in the literature such as the Chinese Postman Problem (*Chinese Postman Problem, CPP*) proposed by the Chinese mathematician, Kwan in 1962 and solved by Edmonds and Johnson in 1973, which broadly consist in pass by all edges of a graph with minimum cost (Kwan, 1962, Edmonds and Johnson, 1973; Minieka, 1979; DeArmon, 1981; Assad et al. 1987; Eiselt et al., 1995a; Pearn and Chou, 1999; Ghiani and Improta, 2000). The Rural Postman Problem (*Rural Postman Problem, RPP*), which is a generalization of the CPP so that only a subset of edges necessarily have to be traveled (Eiselt et al., 1995b; Hertz et al., 1999; Benavent, et al., 2005; Benavent et al., 2007), and the Travelling Salesman Problem (*Travelling Salesman Problem, TSP*), which consist to minimize the cost to go over a graph, starting and ending it at the same vertex, passing only once for each vertex (Miller et al., 1960, Perez-Delgado, 2001; Toulouse and Wolfer, 2009, Petersen et al., 2010; Lusby et al., 2010) .

After analyzing the possibilities to achieve the proposed aims, is designed a simple algorithm that loops through all the towns of each route with the lowest cost. They are the necessary data required by the SIC responsible for optimizing the distances between each of the locations, which have been obtained from ***Google Maps***. The proposed algorithm is shown below:

begin
load matrix *(distances)*
total.distance=0
route=null
keep.route*(beginning point, end point)*
for each *(unvisited location)*
search *(closer location)*
keep.route*(localidad visitada)*
total.distance=total.distance+actual.distance
if (all locations are visited)
total.distance=total.distance+distance.initial.vertex
end each
return (route+total.distance)
end

Algorithm 8.1. Algorithm for Route Optimization (Own Elaboration)

Once implemented the algorithm as a service of the SIC of route optimization, is deployed and run tested on existing routes. It must be noted that this SIC has the ability to access the database that stores the actual locations and routes to visit, so you can rearrange the routes whenever there is a change in them. As shown in **Figure 8.12**, is the *PlannerAgent*, once received the order of his *CommunityControllerAgent*, which access the database that stores the different routes and distances between locations. The *PlannerAgent* sends to the *ExecutorAgent* the parameters needed to run the route optimization service, and in this way will become effective the request. Then the *ExecutorAgent* responds to *PlannerAgent* with the result. In case one of the routes has been changed, or one of the paths is performed so that exists less distance, is the *PlannerAgent* which modifies the database and updates the new route.

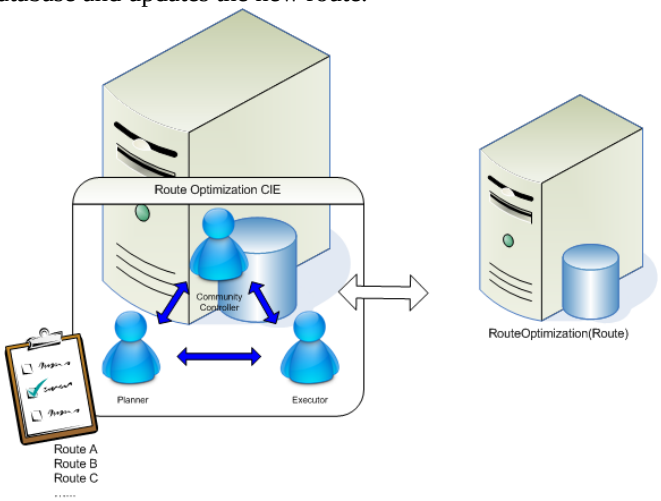


Figure 8.12 Representation of the Specialized Intelligent Community of Route Optimization

Having seen the role of each of the Specialized Intelligent Communities, and detailed their functions, in this section we proceed to explain the operation of the entire system. It will explain the relationship and coordination of each SIC, and the steps that follow to reach its achievements, under SCODA architecture.

As discussed in previous sections, the data necessary for the proper functioning of our multi-agent system have been introduced into a database created exclusively for this case of study, so that the data managed by the company are not affected.

Figure 8.13 shows the SCODA architecture deployed for case of study. Communication process is described below, and depicted in **Figure 8.14**.

1. An external application has the ability to access the system. From this application, can be displayed the generated control files by the *QualityAgent*, which show the system operation in real time. Furthermore, it allows visualizing the operations of each of the SIC, and checking the information handled by them. It is also allowed to request the services offered by the SIC in an independent way, because it can be a need to perform some operation on them outside their operating cycle. **Figure 8.15** shows a detail of access to one of the records of the database. In this case is shown the prediction of a given product demand during a week.
2. Through the external application, every Monday at 6.00 running a request for predicting demand, which will exist throughout the week. It should be noted that in view of the results obtained in predicting demand for different products, it is decided that the system operation will be performed on products with Type C, because the prediction errors obtained for these products are acceptable to test the system. This request is collected by the *CommunicatorAgent*, which selects the corresponding SIC, and it requests a prediction service execution to the *CommunityControllerAgent* corresponding. After execution of the request by the SIC of prediction generates an entry in the database with the results, and these results are returned to the *CommunicatorAgent* that automatically sends a request to the SIC of inventories management, to gets the roadmap ordering for this week.

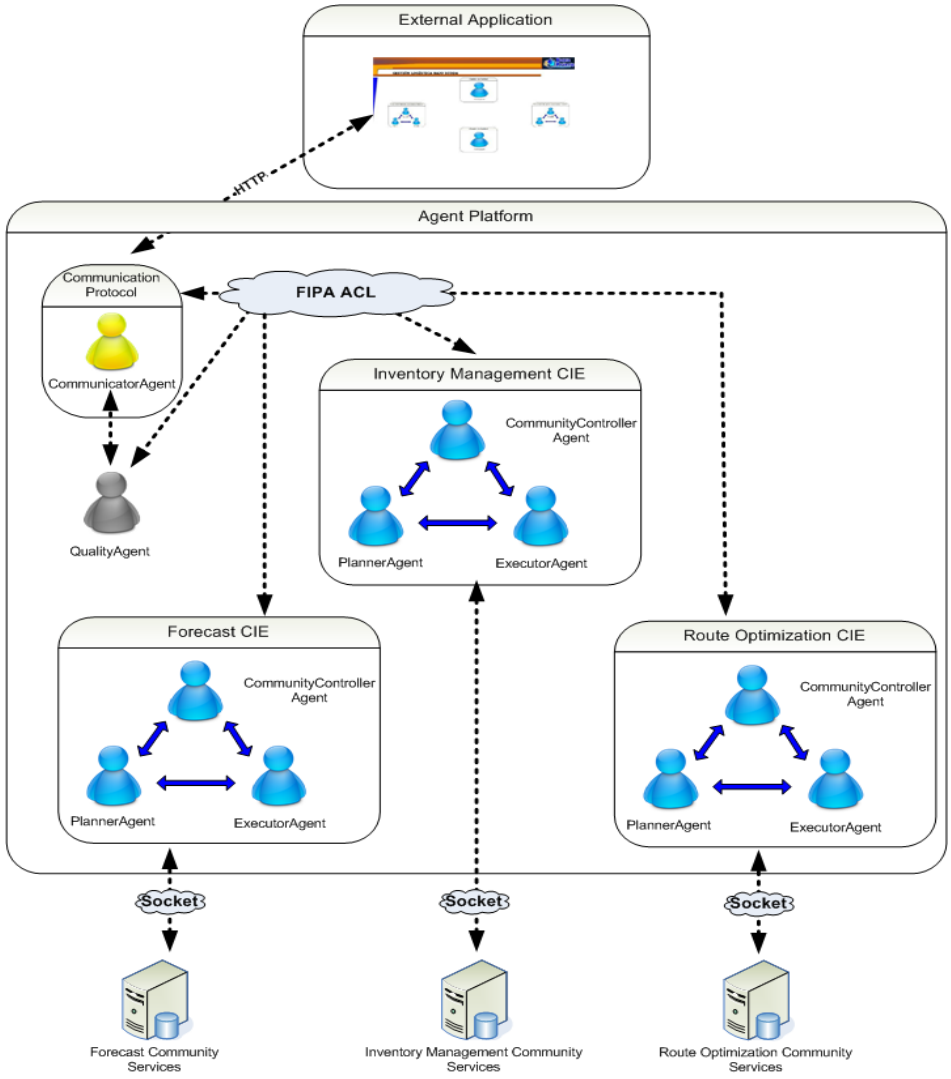


Figure 8.13 Representation of SCODA Architecture for Study Case

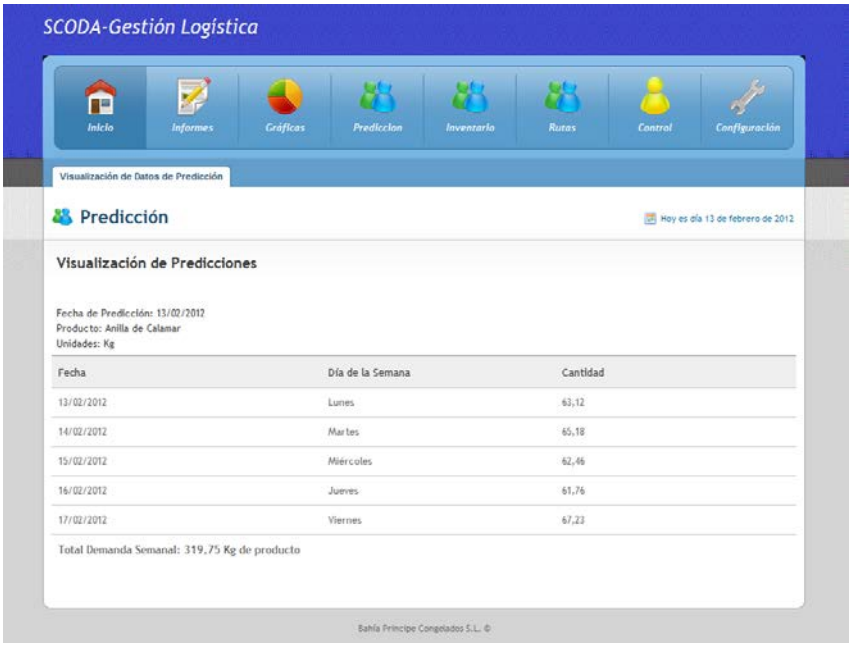


Figure 8.14 Visualization of the Demand Forecast on a Product

3. Once receipt of the request for service execution of inventory, the *CommunityControllerAgent* of the SIC of inventories transmits the request to the work team this SIC, where the *PlannerAgent* gains access to the database inventory management and collects the data necessary for the execution the service. When the results of the execution of the service of inventory are obtained, are sent to the *CommunicatorAgent*, and in turn is generated an entry in the database with the optimal quantity order, the period of time and the reorganizing point (the current week), where will become effective the implementation of the inventory model. The *CommunicatorAgent* generates a report with the data to make orders and send it by e-mail, to the responsible of the camera.

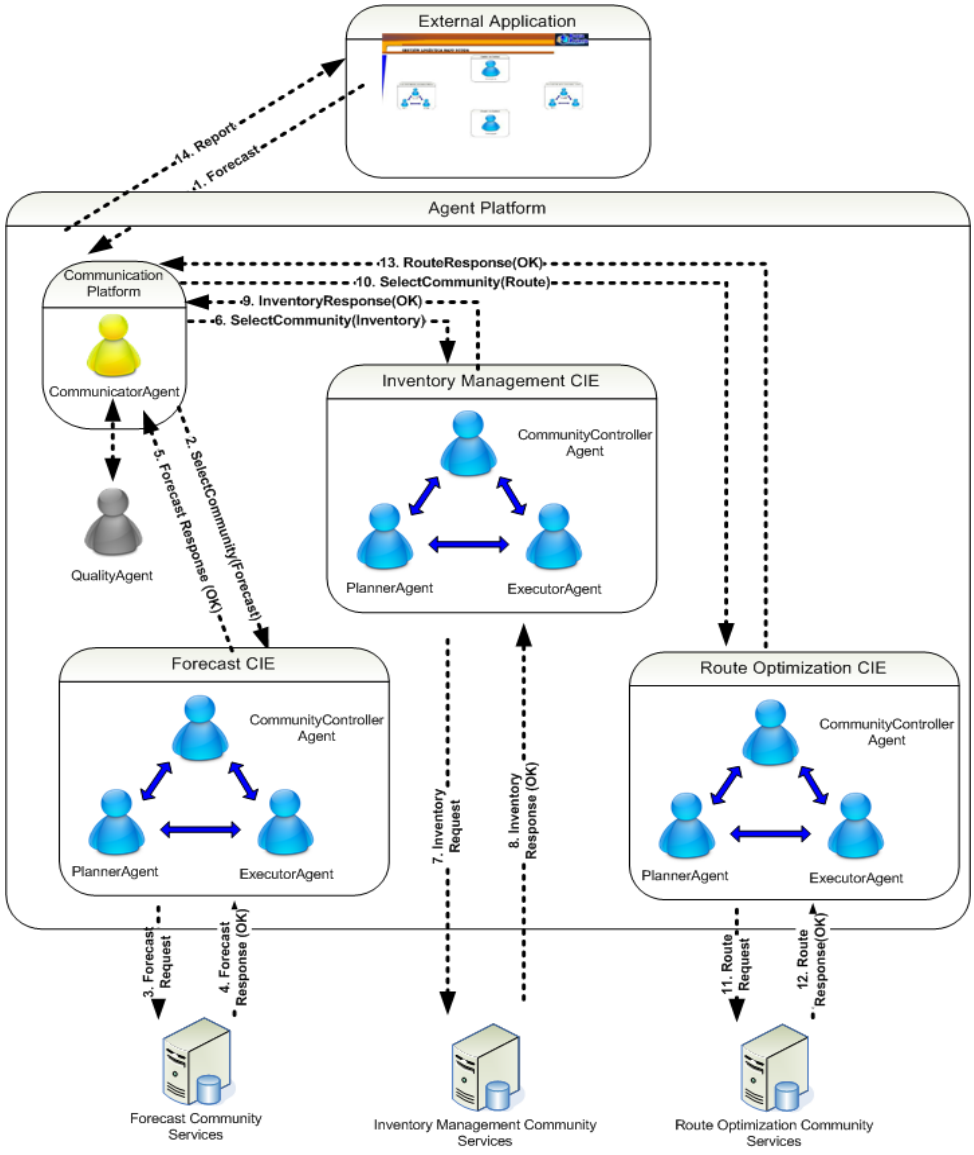


Figure 8.15 Visualization of the Demand Prediction on a Product

Once you know the order data for optimal order, and the reorder point, that is, the number of products that must be ordered and the number of products that have to be in the camera so that the order becomes effective, which have been stored in the database of inventory, and seeing the need to warn the person responsible for this camera when arrives the reorder point, it is decided to implement another service for the SIC of inventory management, which simply check the stock of products and once reached the reorder

point, it must communicate via e-mail with the responsible of the camera. The execution of this service is done every hour internally in the SIC of inventory management.

4. Looking at the results shown in route optimization, the company proposes the possibility of performing this optimization in all of its vehicles, so it allows the access to the general database of routes and vehicles. Viewing their interest in this type of service is decided to carry out, from external application, a weekly check of the modification of the routes database in order to request to the platform, the reorganization of the routes that had been modified. While it is true, there are changes in the rerouting depending on the time of year that we are, and in this way we can always have the optimal route for each vehicle. Once receiving the request, the *CommunicatorAgent* makes a request for optimization service, after selecting the SIC responsible.
   The *CommunityControllerAgent* requests the execution the service and the *PlannerAgent* accede to the general database of routes to obtain the parameters necessary for its implementation, which are sent to the *ExecutorAgent* for execute service, and then communicates the *PlannerAgent* results. This agent modifies the general database of routes, and from the SIC, is sent the results to the *CommunicatorAgent* that generates a report and sends it to the external application.
5. As provided in *Chapter 5* of this work, the *QualityAgent* is in constant communication with the *CommunicatorAgent* and with each *CommunityControllerAgent* to check its operation and perform corrective actions in the event that they were needed.

8.6. Analysis and Conclusions

In order to evaluate the system raises a number of issues that affect both the architecture and the result of the execution of the services associated with each community and their integration from SCODA. First of all, we evaluate the communities at their level of integration and operation with regard to their specialization. The following evaluation is established for the services associated with each community in order to measure their efficiency.

The system was evaluated over a period of five months from 01/01/2012 to 31/05/12. During this time the multiagent system that was deployed for this purpose worked according to the terms proposed in this section. This allowed us to collect the information required to properly analyze the system's functioning and to compile the list of conclusions that are presented in this section.

During the five month period, the multiagent system under SCODA demonstrated optimal performance with regard to its integration: an analysis of the daily statistics file kept by the *QualityAgent* showed that only 4% of the cases produced an error, which corresponded to an error in the services of each community (Inactive Service). These errors were unexpected in the server housing

the SIC services due to the fact that, in line with the principle of distributed computing, the services and the SICs were kept in two different servers. This confirms that the internal communication among agents, the most appropriate SIC section as per the requests sent to the system, the control carried out by the *QualityAgent*, the instantiation and subsequent destruction of the work teams for each SIC, and the overall functioning of the architecture have all been shown to be robust and reliable in this type of application.

In a second period of time, the system was evaluated from 01/09/2012 to 31/09/13 we improved the hardware architecture and communications. This improvement allows the system runs in an optimized way.

The following charts demonstrate the results obtained by the system in the proposed case study, in terms of the execution of the services associated with the SIC. First of all, the results obtained by the prediction SICs were analyzed. The obtained results, implemented for the effects, varied significantly according to the product. As previously mentioned, this was to be expected since there are already products in existence whose time series does not follow any particular pattern and whose prediction is quite costly. The obtained results are shown in **Table 8.14** and correspond to the tests carried out during the month of September, 2011.

Table 8.14 Errors of Forecast on Daily Demand of Products

1. The typology refers to products with a similar set of time series demands.

Product	Tipology[1]	MAPE	ME	RMSE
Product 1	A	17,15%	33,46	14,60
Product 2	C	4,23%	12,51	4,26
Product 3	C	3,16%	11,39	3,78
Product 4	A	17,12%	31,91	14,86
Product 5	B	81,51%	15,74	6,81
Product 6	A	16,24%	35,91	13,99
Product 7	B	29,79%	21,32	10,65
Product 8	B	37,23%	12,11	5,94
Product 9	B	32,39%	21,53	12,52
Product 10	C	3,94%	12,52	4,30
Product 11	A	16,67%	34,03	14,39

With this in mind, products that had similar characteristics in their demand series were grouped together in various typologies. The following groups were created:

A: *Products with a daily demand of up to 100 units, with no regular recurrence in their series.*

B: *Products with a daily demand of up to 30 units, with no regular recurrence in their series.*

C: *Products with a daily demand of up to 80 units, with seasonal recurrence in their series.*

Given the results shown in **Table 8.14**, we can see that the estimate for the series of products with a seasonal recurrence, i.e., those consumed in mass quantities during specific times of the year, has a much lower MAPE error than the others.

Furthermore, for the series with no seasonal variation, the percentage of the MAPE error increases significantly when product demand fluctuates among lower numbers. The following three charts correspond to one of the products according to type. We can see the different demand curves as well as the prediction and errors. These graphs allow us to easily observe the maximum errors of prediction. In **Figure 8.16** we can see that the errors are more pronounced due to the fact that the quantities used in this series are less than those in the other two cases.

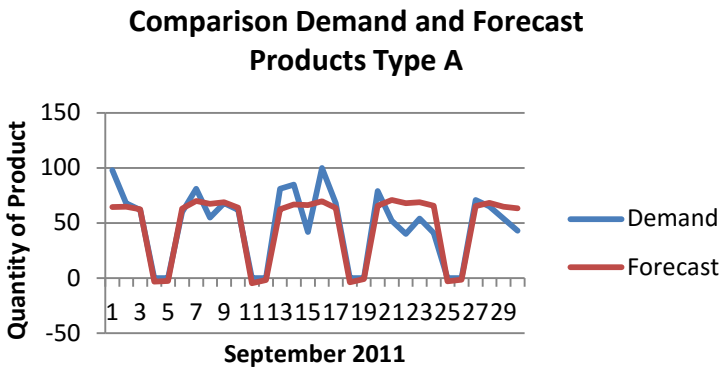


Figure 8.16 Comparison Demand and Forecast for Type A Products

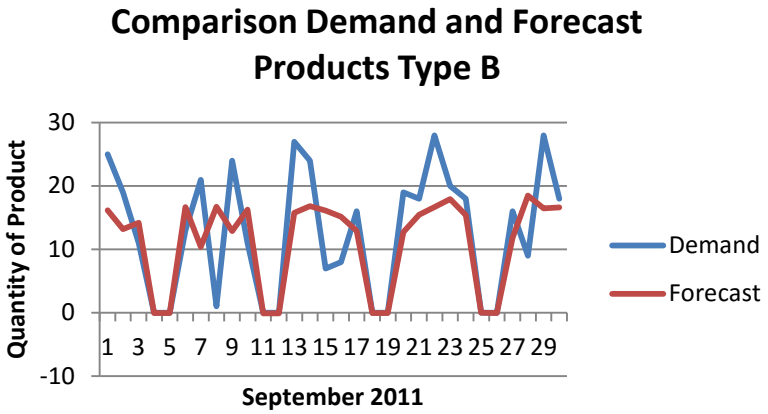


Figure 8.17 Comparison Demand and Forecast for Type B Products

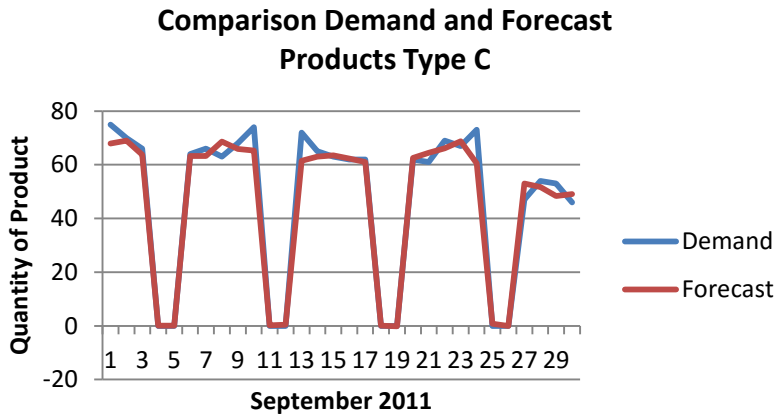


Figure 8.18 Comparison Demand and Forecast for Type C Products

The prediction models used in the study were not able to precisely detect the peaks in the graph. This is because non-working days, in which there is no product demand, are detected perfectly by the model. The detection of these peaks is an interesting task for future study, since we intend to develop an SIC repository in the future, which will require more testing with other parameters or even prediction techniques. Once the results obtained by the prediction models have been explained, it should be noted that the SIC itself, through the *PlannerAgent*, is responsible for selecting the prediction models according to the type of product. This accelerates the selection of the service needed for the prediction, since it is not possible to indicate the model, allowing the *ExecutorAgent*, using the data provided by the *PlannerAgent*, to carry out the request made by the corresponding service.

With regard to the SIC that manages the inventory, and in light of the results shown in **Table 8.15** and the characteristics of each of the products, we can see how the reorder point for *Product A* (referring to the minimal amount of product that must be available so that there is no shortage) is 190.46, while its optimal order point is 115.25. In other words, when the amount of *Product A* available is at 191 units, the system will generate an order for 116 units. The same occurs with *Product B* allowing us to ensure there will be no shortages.

Table 8.15 Results obtained on application of the EOQ Model for A and B Products

Product	Optimal Order Quantity	Reorder Point
Product A	115,25	190,46
Product B	100,16	127,27

Finally, **Table 8.16** shows the results obtained after applying the optimized routes as recommended by the corresponding SIC. The optimization is applied to the routes for the truck that the company made available to us for our study, which includes five different routes every week throughout the provinces of Salamanca, Valladolid and Zamora.

Table 8.16 Results with Optimization for five Different Routes

Route	Current Kms	Optimized Kms	Kms Saved
Route A	390,4	383,9	6,5
Route B	329,8	283,7	46,1
Route C	147,9	113,8	34,1
Route D	89,8	89,8	0
Route E	356,1	279,2	76,9

The results shown in **Table 8.16** indicate a significant reduction in total distance travelled. This implies that a route optimization is needed for the different company vehicles. Just with the vehicle analyzed in the case study, the differences in three of its routes, with regard to total distance travelled between the current and the optimized route, were significant. Considering that these routes are travelled four or five times per month, the total number of kilometers is very important as it can lead to significant savings in fuel and time for the company's entire fleet of 10-15 vehicles.

With regard to execution of services, the performance of the SIC produced very satisfactory results. Although each SIC was tested individually, with the results shown in previous sections, when the complete system is implemented and the SICs are working together with SCODA, we have demonstrated that the performance at the service level is exactly the same as when each is executed individually. This allows us to confirm that we are headed in the right direction with regard to our philosophy of reusing the SICs.

The specialization of the SICs allows them to function independently and resolve the simple problems their servers are equipped for. Furthermore, the added benefit of joining the specializations of each SIC, thus emulating a business network, is the ability to solve more complex problems, allowing each SIC to provide its unique ability, while maintaining SCODA as the common framework that unifies and orchestrates the process. This provides the implemented SICs with the ability to work cooperatively and reach the system objective of the case study. It should be noted that the implementation of these SICs can be used in various developments, regardless of their objective.

The repercussions and acceptance of the system within the company were specifically analyzed. During the first two months, the person in charge of the camera tracked some of the reports generated by the system, noting that the results obtained in the prediction report as compared to the actual demand differed between 2%-5%, of which between 1.5% and 4% were attributed to the error made by the

same individual. With regard to the optimal order requests, the methodology was modified due to the fact that before implementing the system, there had been only one order for a substantially larger quantity which, by not taking the reorder point into account, could result in a shortage of material. By controlling the reorder point, in addition to the fact that our system could allow for errors in the prediction, there was no need to monitor the results generated by the system during the last three months of the testing period as there was no shortage of stock, resulting in the correct amount of workflow in the company. This resulted in important time savings since there was no need to supervise the product inventory, the system itself managed the inventory quite satisfactorily.

The route optimization was completely satisfactory for the company, since the shorter distances travelled allowed for important savings in fuel. Additionally, by dynamically managing the routes, it was possible to include new locations while also ensuring that the route followed was always the most optimal.

However, while the system did function satisfactorily, there are many areas for improvement. Predicting demand is a complex process that requires further analysis for each individual product. This improvement is not specifically limited to the implementation of SCODA, but to the design and implementation of the SIC services. Furthermore, there are many other processes within the company that can be improved by expanding the system with new SICs and new services such as automating orders and providers, automating the control of fuel providers, or joining the system with the company's existing sales management software.

At a conceptual level, the proposal used to measure the specialization of the organizations, both at an internal and external level, has made it possible to establish the parameters of and quantify the specialization of each SIC. Using this level of specialization, it is possible to quantify the performance of an agent in terms of its functionality and its characteristics for interacting with its surroundings. This measure of organizational specialization can be applied to any type of multiagent system, making it possible to compare different systems and indicate the internal level of cooperation of the agents.

BIBLIOGRAFÍA

9. BIBLIOGRAFÍA

Ah-Hwee, T., Yew-Soon, O., Akejariyawong, T. (2012). A hybrid agent architecture integrating desire, intention and reinforcement learning. *Expert Systems with Applications.* 38 (7), 8477-8487.

Amigoni, F. y Fugini, M. (2007). An Agent-Based Environment for Web Service Enabled E-Science. *BioinfoGRID Symposium 2007*, LITA CNR-ITB, Milan, Italy.

Argente Villaplana, E. (2008). GORMAS: *Guías para el desarrollo de sistemas multiagente basados en organizaciones*. Tesis Doctoral. Universidad Politécnica de Valencia.

Argente, E., Botti, V., y Julian, V. (2011). Gormas: an organizational-oriented methodological guideline for open mas. *In Proceedings of the 10t international conference on Agent-oriented software engineering*, 32-47. Springer-Verlag.

Arkin, R. (1998). *Behavior based robotics*. Cambridge, USA: MIT Press.

Arkin, R., y Balch, T. (1999). Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14 (6), 926-939.

Arruñada, B. (1990). *Economía de la Empresa: Un enfoque contractual*. Ariel, Barcelona, 1990,

Assad, A., Pearn, W.L. y Golden, B.L. (1987). The capacitated Chinese postman problem: Lower bounds and solvable cases. *Am. J. Math. Mgt. Sci.* 7(1-2), 63–88.

Atiya, A.F., El-Shoura, S.M., Shaheen, S.I. y El-Sherif, M.S. (1999). A comparison between neural network forecasting techniques-case study: river flow forecasting, Neural Networks. *IEEE Transactions on Neural Networks*, 10(2), 402-409.

Aumada L. (2001). *Teoría y cambio en las organizaciones: un acercamiento desde los modelos de aprendizaje organizacional.* Ediciones Universitarias de Valparaíso. Santiago de Chile.

Axhausen, K.W. y Smith, R.L. (1984). Evaluation of Heuristic Transit Network Optimization Algorithms. *Transportation Research Record,* 976, 7-20.

Baaj, M. H. y Mahmassani, H. S. (1991). An AI-Based Approach for Transit Route System Planning and Design. *Journal of Advanced Transportation,* 25(2), 187-210.

Bajo J., Corchado, J.M., Botti, V. y Ossowski, S. (2009). Practical Applications of Agents and MAS: Methods, Techniques and Tools for Open MAS. *Journal of Physical Agents. Special Sessions on Practical Applications of Agents and MultiagentK*, 3(2), 1-2.

Bajo J., Fraile J.A., Corchado J.M. y Pérez-Lancho B. (2010). The THOMAS Architecture in Home Care Scenarios: A case study. *Expert Systems with Applications*, 37(5), 3986-3999.

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

Balch, T. (2002a). Measuring robot group diversity. In T. Balch y E. Parker (Eds.), *Robot teams: From diversity to polymorphism*, 93-135. Natick, USA: A K Peters.

Balch, T. (2002b). Taxonomies of multi-robot task and reward. In T. Balch y E. Parker (Eds.), *Robot teams: From diversity to polymorphism*, 23-35. Natick, USA: A K Peters.

Barnard, C. (1938). *The Functions of the Executive*. Harvard University Press, Cambridge, MA, (fifteenth printing, 1962).

Barney, J. (1991). Firm resources and sustained competitive advantage. *Journal of management*, 0, 39-61.

BBN Technologies. (2004). *Cougaar Architecture Document V11.0*, 8 March 2004.

Becerra, F. (2008). Las redes empresariales y la dinámica de la empresa: aproximación teórica. *INNOVAR. Revista de Ciencias Administrativas y Sociales (en línea)*, 18. Disponible en: http://www.redalyc.org/pdf/818/81803203.pdf.

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., y Rimassa, G. (2001). Developing multi agent systems with a fipa-compliant agent framework. *Software - Practice And Experience*, 31(2), 103-128.

Belousov, A., Verzakov, S.A. y Von Frese, J. (2002). A flexible classification approach with optimal generalisation performance: support vector machines. *Chemometrics and Intelligent Laboratory Systems.* 64, 15-25.

Benavent, E., Corberán, A., Piñana, E., Plana, I. y Sanchís, J.M. (2005). New heuristic algorithms for the windy rural postman problem. *Computer Operations Research*, 32(12), 3111–3128.

Benavent, E., Carrotta, A., Corberán, A., Sanchís, J.M. y Vigo, D. (2007). Lower bounds and heuristics for the windy rural postman problem. *Eur. Journal in Operations Research*, 176(2), 855–869.

Biswas, P., Mishra y K., Mahanti, N.C. (2005). Computational Efficency of Optimized Shortest Path Algorithms. *International Journal of Computer Science & Applications,* II(II), 22-37.

Birkin, M. y Wu, B. (2012). A Review of Microsimulation and Hybrid Agent-Based Approaches. *Agent-Based Models of Geographical Systems.51-68.*

Blumenthal, J., y Parker, G. (2004). Competing sample sizes for the co-evolution of heterogeneous agents. In *Proceedings of the 2004 ieee/rsj international conference on intelligent robots and* systems, 1438-1443. Sendai, Japan: IEEE Press.

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

Bonabeau, E., Sobkowski, A., Theraulaz, G., y Deneubourg, J. (1997). Adaptive task allocation inspired by a model of division of labour in social insects. In D. Lundh, B. Olsson, y A. Narayanan (Eds.), *Bio-computing and emergent computation*, 36-45. Singapore: World Scientific.

Bonabeau, E. y Theraulaz, G. (1999). Role and variability of response thresholds in the regulation of division of labour in insect societies. In J. Deneubourg & J. Pasteels (Eds.), *Information processing in social insects*, 141-163. Basel, Switzerland: Springer Verlag.

Bongard, J. (2000). The legion system: A novel approach to evolving heterogeneity for collective problem solving. *In Proceedings of eurogp-2000*, 16-28. Edinburgh, UK: Springer-Verlag.

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

Borisov, E.F., Zhamin, V.A. y Makarova, M. F. (1975). *Diccionario de Economía Política.* Barcelona, España.

Bradshaw, J. (1997). *Software Agents.* MIT Press, Cambridge, MA.

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

Brenner, W., Wittig, H. y Zarnekow, R. (1998). Intelligent software agents: Foundations and applications. *Secaucus, NJ, USA: Springer-Verlag*. New York, Inc.

Brooks, C., y Durfee, E. (2000). Congregating and market formation. *In Proc. 1st Int. Joint Conference on Autonomous Agents and Multiagent Systems,* 96-103. ACM Press.

Brooks, C., y Durfee, E. (2003). Congregation formation in multiagent systems. *Autonomous AGents and Mulit-AGent Systems* 7, 145-170.

Brustolini, J.C. (1991). *Autonomous agents: characterization and requirements.* Technical Report CMU-CS-91-204, Carnegie-Mellon University.

Bryant, B., y Miikkulainen, R. (2003). Neuro-evolution for adaptive teams. *In Proceedings of the 2003 congress on evolutionary computation*, 2194-2201). Canberra, Australia: IEEE Press.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2 (2), 121-167.

Burns, T. y Stalker, G.M. (1961) *The Management of Innovation*. Chicago. Quadrangle Books.

Burns, D. (2007). *Systemic Action Research: A strategy for whole system change*. Bristol: Policy Press.

Burt, R. (1980). Cooptive corporate actor networks: A reconsideration of interlocking directorates involving American manufacturing. *Administrative Science Quarterly*, *25*(4), 557-582.

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

Caire, G., Coulier, W., Garijo, F., Gomez, J., Pavon, J., Leal, F., Chainho, P., Kearney, P., Stark, J., Evans, R. y Massonet, P. (2002). Agent-oriented analysis using MESSAGE/UML. *Lecture Notes in Computer Science 2222*, 119-125.

Calderone, N. y Page, R. (1988). Genotypic variability in age polyethism and task specialization in the honey bee. *Apis mellifera. Behav. Ecol. Sociobiol*, *22*(1), 17-25.

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

Campos, C., Theraulaz, G., Bonabeau, E. y Deneubourg, J. (2001). Dynamic scheduling and division of labor in social insects. *Adaptive Behavior*, 8(2), 83-94.

Carlési, N., Michel, F., Jouvencel, B., Ferber, J. (2011). Generic architecture for multi-AUV cooperation based on a multi-agent reactive organizational approach. Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, 5041-5047.

Castro, D., Braga, R., Reis, L.P. y Oliveira, E. (2012). Designing a meta-model for a generic robotic agent system using Gaia methodology. *Information Sciences.* 195, 190-210.

Cervenka, R. (2005). *Modeling Multi-Agent Systems*. PhD Thesis. Department of Computer Science. Faculty of Mathematics, Physics and Informatics. Comenius University in Bratislava.

Cervenka, R., Trencansky, I. (2004). *Agent Modeling Language: Language Specification Version 0.9*. Technical report, Whitestein Technologies.

Cervenka, R., Trencansky, I., Calisti, M. y Greenwood, D. (2005). AML: Agent Modeling Language Toward Industry-Grade Agent-Based Modeling. *In Proc. AOSE 2004*, LNCS 3382, 31-46.

Cervenka, R., y Trencansky, I. (2007). AML. The Agent Modeling Language. *Whitestein Series in Software Agent Technologies and Autonomic Computing*. Birkauser Verlag.

Chakrabartty, S. (2003). *The giniSVM toolkit*. Technical Report No. 48, *CLSP, JHU.*

Chang, C.C. y Lin, C.J. (2011). LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*. 2, 1-27.

Cheng, C. E. (1991). An economic order quantity model with demand-dependent unit production cost and imperfect production processes. *IIE Transactions.* 23, 23–28.

Collins, J. y Ndumu, D. (1999). *The Zeus Agent Building Toolkit*. ZEUS Technical Manual.

Collobert, R. y Bengio, S. (2001). SVMTorch: Support Vector Machines for Large-Scale Regression Problems. *Journal of Machine Learning Research.*

Corchado, J.M. y Lees, B. (2001). A hybrid case-based model for forecasting. *Applied Artificial Intelligence: An International Journal.* 15, 105-127.

Corchado, J.M. y Aiken, J. (2002). Hybrid Artificial Intelligence Methods in Oceanographic Forecasting Models. *IEEE SMC Transactions Part C.* 32, 307-313.

Corchado, J.M. (2005). *Agentes Software y Sistemas Multiagente.* Pérez de la Cruz J.L. (Ed). Pearson Education.

Corchado, J. M., Bajo, J., De Paz, Y. y Tapia, D. I. (2008). Intelligent environment for monitoring Alzheimer patients, agent technology for health care. *Decision Support Systems*, 44 (2), 382-396.

Corchado J.M., Tapia D.I. y Bajo J. (2010). Multi-Agent System to Monitor Oceanic Environments. *Integrated Computer-Aided Engineering (ICAE),* 17(2), 131-144.

DeArmon, J. (1981). *A comparison of heuristics for the capacitated Chinese postman problem.* Master's thesis, University of Maryland, College Park MD.

Decker, K., Lesser, V., Prasad, N. y Wagner, T. (1995). MACRON: An Architecture for Multi-Agent Cooperative Information Gathering. *In Proc. CIKM Workshop on Intelligent Information Agents.*

Díaz, A., Glover, F., Ghaziri, H.M., González, J.L., Laguna, M., Moscato, P. y Tseng, F.T. (1996). *Optimización Heurística y Redes Neuronales en Dirección de Operaciones e Ingeniería*. Editorial Paraninfo S.A., Madrid (España).

Dignum, V., y Dignum, F. (2006). *A landscape of agent systems for the real world*. Technical Report 44-CS-2006-061. Institute of Information and Computing Sciences, Utrecht University.

Dignum, V., y Dignum, F. (2012). *A logic of agent organizations*. Logic Journal of IGPL. Institute, 20 (1), 283-316.

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

Edmonds, J. y Johnson, E. L. (1973). Matching, Euler tours and the Chinese postman. *Math Programming*, 5, 88-124.

Eiselt, H.A., Gendreau, M. y Laporte, G. (1995a). Arc routing problems, part I: The Chinese Postman Problem. *Operations Research.* 43, 231–242.

Eiselt, H.A., Gendreau, M. y Laporte, G. (1995b). Arc Routing Problems, Part 2: The Rural Postman Problem. *Operations Research*, 43, 399-414.

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

Eraydin, A. y 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.

Esteva, M., Rodriguez, J., Sierra, C., Garcia, P. y Arcos, J. (2001). On the formal Specification of Electronic Institutions. *Agent Mediated Electronic Commerce*, 1991, 126-147.

Fanyol, H. (1949). *General and Industrial of Management.* London, Pittman

Ferber, J., Gutkenecht, O., y Michel, F. (2003). From agents to organizations : an organizational view of multi-agent systems. *In Proc. AAMAS03 - Agent-Oriented Software Engineering Workshop (AOSE).*

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

FIPARationale. (1996). Technical report, *Foundation for Intelligent Physical Agents.*

Fischer, K. (1999). Agent-based design of holonic manufacturing systems. *Journal of Robotics and Autonomous Systems*. 27(1-2), 3-13.

Franklin, S. y Graesser, A. (1996). Is it an agent, or just a program?: A taxonomy for autonomous agents. In J. P. Müller, M. Wooldridge, y N. R. Jennings (Ed.), *Proceedings of the 3rd International Workshop on Agent Theories, Architectures and Languages.* LNCS 1193, 21-35. Springer-Verlag.

Funes, P., Orme, B., y Bonabeau, E. (2003). Evolving emergent group behaviors for simple humans agents. *In Proceedings of the seven European conference on artificial life*, 76–89. Berlin, Germany: Springer-Verlag.

Furtado, V., Melo, A., Dignum, V., Dignum, F. y Sonenberg, L. (2005). Exploring congruence between organizational structure and task performance: a simulation approach. In: Boissier, O., Dignum, V., Matson, E., Sichman, J. (eds.), *Proc. of the 1st OOOP Workshop*.

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

Garcia-Fornes, A., Hbner, J. F., Omicini, A., Rodriguez-Aguilar, J. A. y Botti, V. (2011). Infrastructures and tools for multiagent systems for the new generation of distributed systems. *Engineering Application of Articial Intelligence*, 24, 1095-1097.

García-Montoro, C. (2007). *Análisis y clasificación de lenguajes de programación orientados a agentes.* Technical Report.

Garcia-Ojeda, J.C., DeLoach, S.A., Robby, Oyenan, W.H., y Valenzuela, J. (2007). O-MaSE: A Customizable Approach to Developing Multiagent Development Processes. *Agent-Oriented Software Engineering VIII: The 8th International Workshop on Agent Oriented Software Engineering (AOSE 2007), LNCS.* 4951, 1-15.

Garijo, F., Gomez-Sanz, J., Pavon, J. y Massonet, P. (2001). Multi-Agent System Organization. An Engineering Perspective. *Proceedings MAAMAW*, 1-15.

Gasser, L. (2001). Perspectives on organizations in multiagente systems. In Luck, M., Marik, V., Stepankova, O., y Trappl, R., eds., *Multi-agent Systems and Applications. 9th ECCAI Advanced Course, EASSS*, Lecture Notes in Computer Science, 1-16. Springer.

Gautrais, J., Theraulaz, G., Deneubourg, J., y Anderson, C. (2002). Emergent polyethism as a consequence of increased colony size in insect societies. *Journal of Theoretical Biology*, 215(1), 363-373.

Ghiani, G. y Improta, G. (2000). An algorithm for the hierarchical Chinese postman problem. *Operations Research*. Letters 26 (1), 27-32.

Giddens, A. (2000). Las organizaciones modernas. *En Sociología*, 443-446. España: Alianza Editorial.

Giorgini, P., Müller, J., Odell, J. (2004). AOSE IV, *LNCS* 2935, Springer-Verlag, Berlin.

Giret, A. (2005). ANEMONA: *Una metodología multi-agente para sistemas holónicos de fabricación*. Tesis Doctoral. Universidad Politécnica de Valencia.

Giunchiglia, F., Mylopoulos, J. y Perini, A. (2002). The Tropos software development methodology: Processes, models and diagrams. *In Proc. Workshop on Agent Oriented Software Engineering (AOSE-2002)*, 63-74.

Glinton, R., Paruchuri, P., Scerri, P., y Sycara, K. (2010). Self-Organized Criticality of Belief Propagation in Large Heterogeneous Teams. In M. J. Hirsch, P. M. Pardalos y R. Murphey (Eds.), *Dynamics of Information Systems: Theory and Applications*, Springer, Berlin, Germany.

Golden, B., Magnanti, T. y Nguyen, H. (1977). Implementing vehicle routing algorithms. *Networks*, 7, 113–148.

Gómez, J. J. (2002). *Modelado de Sistemas Multi-Agente*. Tesis Doctoral. Universidad Complutense de Madrid.

Gómez-Sanz, J. y Pavón, J. (2005). Implementing Multi-agent Systems Organizations with INGENIAS. *In Programming Multi-Agent Systems: Third*

International Workshop, ProMAS. Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, Amal ElFallah Seghrouchni (Eds.), LNCS 3862, Springer-Verlag, 236 - 251.

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

Guide J. y Srivastava R. (1997). Repairable inventory theory: models and applications. *European Journal of Operational Research*, 102, 1-20.

Guzmán, I. (1983). *Reflexiones en torno al Orden Social*. México: Jus.

Hadley, G. y Whitin, T. (1963). *Analysis of Inventory Systems.* Prentice Hall, Englewood Cliffs, N.J.

Hannan, M. T. y Freeman, J. (1989). *Organizational Ecology*. Cambridge, MA: Harvard University Press.

Haynes, T. y Sen, S. (1996). Evolving behavioral strategies in predators and prey. In G.Weiss y S. Sen (Eds.), *Adaptation and learning in multi-agent systems: Lecture notes in computer science*, 113-126. Berlin, Germany: Springer-Verlag.

Hermoso, R., Billahardt, H., y Ossowski, S. (2010). Role Evolution in Open Multi-Agent Systems as an Information Source for Trust. *In Proc. AAMAS*, 1, 217-224.

Hernández, J. y Salazar, S. (2006). Implementación de una Máquina de Vectores Soporte Empleando FPGA. *Scientia Et Técnica*, 31(XII), 47-52.

Hertz, A., Laporte, G. y Nanchen, P. (1999). Improvement Procedures for the Undirected Rural Postman Problem. *INFORMS Journal on Computing*, 11(1), 53-62.

Hodge, B. J., Anthony, W. y Gales, L. (2003). *Teoría de la Organización: un enfoque estratégico*. Pearson Educación.

Horling, B., Mailler, R., Shen, J., Vincent, R. y Lesser, V. (2003). Using Autonomy, Organizational Design and Negotiation in a Distributed Sensor Network. *In Distributed Sensor Networks: A multiagent perspective*, 139-183. Kluwer Academic Publishers.

Horling, B., y Lesser, V. (2004). A survey of multiagent organizational paradigms. *The Knowledge Engineering Review*, 19, 281-316.

Hsu, C.W. y Lin, C.J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks.*

Hsu, C.W., Chang, C.C. y Lin, C.J. (2010). *A practical guide to support vector classification*. Department of Computer Science National Taiwan University.

Hubner, J., Sichman, J. y Boissier, O. (2006). S-moise+: A middleware for developing organised multi-agent systems. *In Proc. Int. Workshop on Organizations in Multi-Agent Systems, from Organizations to Organization Oriented Programming in MAS*, 3913 Lecture Notes in Computer Science, 64-78, Springer.

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

Iglesias, A., Garijo, M., Gonzalez, J. y Velasco, J. (1996). A methodological proposal for multiagent systems development extending CommonKADS. *In Proc. 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, 1.

d'Inverno, M., Luck, M., Noriega, P., Rodriguez-Aguilar, J.A. y Sierra, C. (2012). Communicating Open Systems. *Artificial Intelligence*, 186, 38-94.

Jennings, N. (1995). Controlling cooperative problem solving in industrial multi-agent systems. *Artificial Intelligence*, 75(2), 195–240.

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

Johansson, B. y Quigley, J. (2004). Agglomeration and networks in spatial economies. *Regional Science*, 83, 165-176.

Juan, T., Pierce, A. y Sterling, L. (2002). Roadmap: Extending the Gaia methodology for complex open systems. *In Proc. 1st Int. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, 3-10.

Kast, F.E. y Rosenzbeig, J.A. (1976). *Administración en las Organizaciones*. México, McGraw-Hill

Kauffman, S. (1995). *At Home in the Universe. The Search for Laws of Self-Organization and Complexity*. NewYork/Oxford: Oxford University Press.

Kawamura, T., Hasegawa, T., Ohsuga, A. y Honiden, S. (2000a). Bee-gent: Bonding and Encapsulation Enhancement Agent Framework for Development of Distributed Systems. *Systems and Computers in Japan*, John Wiley & Sons, Inc., 31(13), 42-56.

Kawamura, T., Kase, N., Araki, D. y Ohsuga, A. (2000b). Development of a Distributed Cooperative Scheduling System based on Negotiations between

Scheduling Agents. *Systems and Computers in Japan,* John Wiley & Sons, Inc., 31(1), 92-101.

Kefalas, P. y Stamatopoulou, I. (2010). Towards modelling of reactive, goal-oriented and hybrid intelligent agents using P Systems. *Lecture Notes in Computer Science*. 6501, 265-272.

Khan, M., Jaber, M.Y., Guiffrida, A.L., y Zolfaghari, S. (2011). A review of the extensions of a modified EOQ model for imperfect quality items. *International Journal of Production Economics.* 132, 1-12.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J. y Irwin, J. (1997). Aspect-Oriented Programming. *In Proceedings of the European Conference on Object-Oriented Programming (ECOOP).* 1241, 220-242.

Koster, A., Schorlemmer, M. y Sabater, J. (2012). Opening the black box of trust: reasoning about trust models in a BDI agent. *Journal of Logic and Computation*.

Kwan, M.-K. (1962). Graphic programming using odd or even points. *Chinese Mathematics*, 1, 273–277.

Laporte, G. (1992). The vehicle routing problem: an overview of exact and approximate algorithms. *European Journal of Operational Research*, 59, 345–358.

Larsen, B. (2000). German organization and leadership theory-stable trends and flexible adaptation. *Scandinavian Journal of Management.* 19(2003), 103-133.

Lazer, D. (2003). Information and innovation in a networked world. En R. Breiger, K. Carley y P. Pattison (Eds.), *Dynamic social network modeling and analysis: Workshop summary and papers*, 101-120. Washington: The National Academies Press.

Lei C., Jiawei C., Zhangang H., Zengru D., y Ying F. (2007). Emergence of Specialization from Global Optimizing Evolution in a Multi-agent Systems. In Shi et al. (Eds.). *ICCS 2007, Part IV*, LNCS 4490, 98-105.

Lerman, K., y Shehory, O. (2000). Coalition formation for large-scale electronic markets. *In Proc. Int. Conference on Multi-Agent Systems.*

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

Li, L., Martinoli, A. y Mostafa, Y. (2002). Emergent specialization in swarm systems. In H. Yin, N. Allinson, R. Freeman, J. Keane, y S. Hubbard (Eds.), LNCS 2412. *Intelligent Data Engineering and Automated Learning Ideal*, 261-266. Berlin, Germany: Springer-Verlag.

Li, L., Martinoli, A. y Yaser, A. (2004). Learning and measuring specialization in collaborative swarm systems. *Adaptive Behavior*, 12(3), 199-212.

Lieberherr, K.J. (1996). *Adaptive Object-Oriented Software: The Demeter Method with Propagation Patterns*. PWS Publishing Company, Boston.

Lind, J. (2001). Iterative Software Engineering for Multiagent Systems: The MASSIVE Method. *Lecture Notes in Computer Science*.

López, C. (2003). *Redes empresariales, experiencias en la Región Andina*. Manual para el Articulador. Lima: Minka.

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

Love, S. (1979). *Inventory Control.* McGraw-Hill, New York.

Lüer, A., Benavente, M., Bustos, J. y Venegas, B. (2009). El problema de rutas de vehículos: Extensiones y métodos de resolución, estado del arte. *Workshop Internacional EIG*.

Luke, S., y Spector, L. (1996). Evolving teamwork and coordination with genetic programming. *In Proceedings of the 1996 international conference on genetic programming*, 150-156. Stanford, USA: MIT Press

Lusby R., Larsen J., Ehrgott M. y Ryan D. (2010). An exact method for the double TSP with multiple stacks. *International Transactions in Operational Research*, 17, 637-652.

Maes, P. (1989). Situated Agents Can Have Goals. *Designing Autonomous Agents: Theory and Practice From Biology to Engineering and Back*, 49-71. Maes, Pattie, (Ed).

Maes, P. (1990). Designing Autonomous Agents. MIT Press.

Maes, P. (1994). Modeling adaptive autonomous agents. *Artifcial Life*, I, 1-2, 1994.

Mangasarian, O.L. y Musicant, D.R. (2001). Lagrangian Support Vector Machines. *Journal of Machine Learning Research.*

March, J.G. y Simon, H.A. (1981). *Teoría de la Organización*. Barcelona, Ariel.

Martín-Merino, M. 2006. *Técnicas Neuronales y Estadísticas para la Predicción de Demanda Eléctrica.* Amarú Ediciones.

Martín-Merino, M. y Román, J.A. (2006a). Electricity Load Forecasting Using Self Organizing Maps. *ICANN, LNCS 4132*, 709-716.

Martín-Merino, M. y Román, J.A. (2006b). A New SOM Algorithm for Electricity Load Forecasting. *ICONIP*, 995-1003.

Martinoli, A., Zhang, Y., Prakash, P., Antonsson, E., y Olney, R. (2002). Towards evolutionary design of intelligent transportation systems. *In Eleventh International Symposium on New Technologies for Advanced Driver Assistance Systems*, 283-290). Siena, Italy.: ATA Press.

Massie, J. (1973). *Bases esenciales de la Administración.* México: Diana.

Maturana, H. y Varela, F.J. (1980). *Autopoiesis and cognition; the organization of the living*. Boston: Ridel.

Maturana, F., Shen, W. y Norrie, D. (1990). Metamorph: An adaptive agent-based architecture for intelligent manufacturing. *Int. Journal of Production Research* 37(10), 2159-2174.

Maudos, J., Pastor, J.M. y Pérez, F. (2002). "Competition and Efficiency in the Spanish Banking Sector: the Importance of Specialisation", Applied Financial Economics, 2002, Vol. 12, 9, 505-516.

McCallum, M., Vasconcelos, W.W. y Norman, T.J. (2005). Verification and Analysis of Organisational Change. In Boissier, O. et al. (eds.), *Proc. 1st OOOP Workshop*.

Melinkoff, R.V. (1969). *La Estructura de la Organización.* Universidad de Venezuela.

Miller C., Tucker A. y Zemlin R. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7, 326-329.

Minieka, E. (1979). The chinese postman problem for mixed networks. *Management Science*, 25, 643-648.

Mintzberg, H. (1989). *El diseño de las organizaciones eficientes*. Ed. El Ateneo. Argentina.

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

Murciano, A. y Millan, J. (1997). Learning signaling behaviors and specialization in cooperative agents. *Adaptive Behavior*, 5(1), 5-28.

Murciano, A., Millan, J. y Zamora, J. (1997). Specialization in multi-agent systems through learning. *Biological Cybernetics*, 76(1), 375-382.

Nicklisch, H. (1920). *El camino hacia delante*. P. 122.

Nicklisch,H. (1928). *Cuestiones fundamentales de la Economía de Empresa*. Stuttgart.

Nicklisch,H. (1932). *La economía de empresa*. Stuttgart

Nitschke, G., Schut, M. y Eiben, A. (2007). Emergent specialization in biologically inspired collective behavior systems. *In Intelligent Complex Adaptive Systems*, 100-140. IGI Publishing, New York, USA.

Noirot, C. y Pasteels, J. (1987). Ontogenetic development and the evolution of the worker caste in termites. *Experientia*, 43(1), 851-860.

Nolfi, S., y Floreano, D. (2000). *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines.* Cambridge, USA: MIT Press.

Nolfi, S., y Parisi, D. (1997). Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior*, 1(5), 75-98.

Nolfi, S., Deneubourg, J., Floreano, D., Gambardella, L., Mondada, F. y Dorigo, M. (2003). Swarm-bots: Swarm of mobile robots able to self-assemble and self-organize. *Ecrim News*, 53(1), 25-26.

Oijen, J. y Dignum, F. (2011). Towards a Design Approach for Integrating BDI Agents in Virtual Environments. *Intelligents Virtual Agents, LNCS*, 6895, 462-463.

Oijen, J., Doesburg, W. y Dignum, F. (2011). Goal-Based Communication Using BDI Agents as Virtual Humans in Training: An Ontology Driven Dialogue System. *Agents for Games and Simulations II, LNAI*, 6525, 38-52.

Okamoto, S., Scerri, P. y Sycara, K. (2008). The Impact of Vertical Specialization on Hierarchical Multi-Agent Systems. *In Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence.*

OMG. (1998). *OMASIF-RTF Results*. Technical report, Object Management Group.
Padgham, L. y Winikoff, M. (2002). Prometheus: A methodology for developing intelligent agents. *In Proc. Workshop on Agent Oriented Software Engineering (AOSE-2002)*, 135–145.

Padgham, L. y Winikoff, M. (2004). Developing *Intelligent Agent Systems: A Practical Guide*. JohnWiley and Sons.

Papazoglou, M. y Georgakapoulos, G. (2003). Introduction to the Special Issue about Service-Oriented Computing. *CACM*, 46(10), 24-29.

Papazoglou, M. y van den Heuvel, W. (2006). Service-Oriented Design and Development Methodology. *In Proc. Int. J. of Web Engineering and Technology (IJWET).*

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

Parachuri, P., Glinton, R., Sycara, K., y Scerri, P. (2010) Effect of humans on belief propagation in large heterogeneous teams. In M. J. Hirsch, P. M. Pardalos & R. Murphey (Eds.), *Dynamics of Information Systems: Theory and Applications.* Springer, Berlin, Germany.

Parlarm, M. y Berkin, D. (1991). Future Supply Un certainty in EOQ Models. *Naval Res. Logist.*, 38, 107-121.

Pavón, J. y Gómez-Sanz, J. (2003). Agent Oriented Software Engineering with INGENIAS. *In Proc. 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2003)*, V. Marik, J. Müller, M. Pechoucek (Eds.), Multi-Agent Systems and Applications II, LNAI 2691, Springer- Verlag, 394-403.

Pavón, J., Gómez-Sanz, J.J. y Fuentes, R. (2005). The INGENIAS Methodology and Tools. *In Agent Oriented Methodologies.* B. Henderson-Sellers and P.Giorgini (Eds.) IDEA Group Publishing, 236-276.

Pearn, W.L. y Chou, J.B. (1999). Improved solutions for the Chinese Postman Problem on mixed networks. *Computers & operations research*, Elsevier Science, 819-827.

Peiro, J. (1991). *Psicología de la Organización*. Universidad Nacional de Educación a Distancia. Ed. Toran, S.A.

Pérez-Delgado, M.L. (2001). *Los mapas de rasgos autoorganizativos y el problema del viajante de comercio*. Tesis Doctoral, Universidad de Salamanca.

Petersen H.L., Archetti C. y Speranza M.G. (2010). Exact solutions to the double travelling salesman problem with multiple stacks. *Networks*, available online at DOI: 10.1002/net.20375.

Pfeffer, J. (1989). *Organizaciones y Teoría de la Organización.* Buenos Aires. El Ateneo.

Pires, A. M. y Machado, V.C. (2005). Management by Processes in the Design of Organizations. *Información Tecnológica*, 17 (1), 35-44.

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

Posland, S. (2007). Specifying protocols for multi-agent systems interaction. *ACM Trans. Auton. Adapt. Systemst*., 2(4), 15.

Potter, M., Meeden, L. y Schultz, A. (2001). Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *Proceedings of the international joint conference on artificial intelligence*, 1337-1343. Seattle, USA: AAAI Press.

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

Quinn, M., Smith, L., Mayley, G. y Husbands, P. (2003). Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. *Philosophical Transactions of the Royal Society of London, Series A: Mathematical, Physical and Engineering Sciences*, 361(1), 2321-2344.

Rao A. S. y Georgeff M. P. (1991). Modeling Rational Agents within a BDI-Architecture. *Second International Conference on Principles of Knowledge Representation and Reasoning (KR91).* San Mateo, C.A.

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

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

Reimann, B. (1974). Dimensions of Structure in Effective Organizations: Some Empirical Evidence. *The Academy of Management Journal*, 17(4) 693-708.

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

Robbins, S. (2004). *Comportamiento Organizacional*. Pearson Educación.

Rodríguez, J., Torres, M. y González, E. (2007). *Avances en Sistemas e Informática*. 4, (2).

Rodriguez, S. (2010). Modelo Adaptativo para Organizaciones Virtuales de Agentes. Tesis Doctoral. Universidad de Salamanca.

Rodríguez, S., de Paz, Y., Bajo, J. y Corchado, J.M. (2011). Social-based planning model for multiagent systems. *Expert Systems with Applications, 38, 13005-13023*

Rosen, M., Lublinsky, B., Smith, K. y Balcer, M. (2008). *Applied SOA: Service-Oriented Architecture and Design Strategies.* Wiley, 2008.

Román, J.A., Rodríguez S. y Corchado, J.M. (2013). Distributed and Specialized Agent Communities. *Trends in Practical Applications of Agents and Multiagents Systems*, 221, 33-40.

Román, J.A., Tapia, D.I. y Corchado, J.M. (2011). SCODA para el Desarrollo de Sistemas Multiagente. *Revista Ibérica de Sistemas y Tecnologías de Información.* 8, 25-38.

Roy, T., y Chaudhuri, K. S, (2009). A production-inventory model under stock dependent demand, Weibull distribution deterioration and shortage, *International Transactions in Operational Research*, 16, 325-346.

Rumelt R., Schedel, D. y Teece, D. (1991). Strategies management and economics. *Strategic Management Journal,* 12 (Special Issue), 556-570.

Rushing, W. (1967). The Effects of Industry Size and Division of Labor on Administration. *Administrative Science Quarterly,* 12 (2), 273-295.

Samuel, Y., Mannheim, B. (1970). Multidimensional Approach Toward a Typology of Bureaucracy. *Administrative Science Quarterly,* 15 (2), 216-228.

Sánchez Vidal A (2001). Medida y estructura interna del sentimiento de comunidad: Un estudio empírico. *Revista de Psicología Social, 16*, 157-175.

Sarason, S. (1974). The psychological sense of community. *Proaspects for a community psychology.* Jossey Bass, San Froncisco.

Scholkopf, B. y Smola, A.J. (2002). *Learning with Kernels*. MIT Press. Cambridge, MA.

Schwaller, R. L. (1988). *EOQ under inspection costs. Production and Inventory Management*, 29, 22–24.

Shehory, O. y Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 1(2), 165–200.

Shen, W. y 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.

Shih, M.C., Mahmassani, H.S. y Baaj, M.H. (1998). Planning and Design Model for Transit Route Netwoks with Coordinated Operations. *Transp. Research Record*, 1623, 16-23.

Shoham, Y. (1997). An Overview of Agent-oriented Programming. *In Software Agents*, J. M. Bradshaw ed. Menlo Park, Calif.: AAAI Press.

Silva, D., Braga, R., Reis, L. y Oliveira, E. (2010). A generic model for a robotic agent system using gaia methodology: Two distinct implementations. *Robotics Automation and Mechatronics (RAM)*, IEEE. 280-285

Silver, E., Pyke D. y Peterson R. (1998). *Inventory Management and Production Planning and Scheduling.* John Wiley and Sons. New York.

Snyder, R. y MacKenzie, D. (2004). Cougaar Agent Communities. *In Proceedings of the 1st Open Cougaar Conference*, 143-148, New York City.

Soh, L., Tsatsoulis, C. y Sevay, H. (2003). Distributed Sensor Networks: A multiagent perspective. *Kluwer Academic Publishers*, 109–138.

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

Stanley, K., Bryant, B. y Miikkulainen, R. (2005). Real-time neuro-evolution in the nero video game. *Evolutionary Computation*, 9(6), 653-668.

Stefan, R. (2000). *mySVM Manual.* Computer Science Department, University of Dortmund.

Stone, P. y Veloso, M. (2002). Towards collaborative and adversarial learning: A case study in robotic soccer. *Evolution and learning in multi-agent systems*, 48(1), 83-104.

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

Sycara, K., Paolucci, M., Van Velsen, M. y Giampapa, J. (2003). The RETSINA MAS Infrastructure. *Autonomous Agents and Multi-Agent Systems*, 7 (1/2), 29-48.

Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7, 83–124.

Tapia, D. I., Corchado, J. M. y Bajo, J. (2006). GERMAS: Multi-Agent System for the Care and Control of Patients in Geriatric Residences. *Proceedings of the 5th International Workshop on Practical Applications of Agents and Multiagent Systems (IWPAAMS'06)*, 63-74. Segovia, España: Universidad de Valladolid.

Tapia D.I., De Paz, J.F., Rodríguez, S., Bajo J. y Corchado J.M. (2008). Multi-Agent System for Security Control on Industrial Environments. *International Transactions on System Science and Applications*, 4(3), 222-226.

Tapia D.I. y Corchado J.M. (2009). An Ambient Intelligence Based Multi-Agent System for Alzheimer Health Care. *International Journal of Ambient Computing and Intelligence (IJACI)*, 1(1), 15-26.

Taylor, F. W. (1916). The Principles of Scientific Management. Bulletin of Taylor Society.

Theraulaz, G., Gervet, J. y Semenoff, S. (1991a). Social regulation of foraging activities in polistes dominulus christ: a systemic approach to behavioural organization. *Behaviour*, 116(1), 292-320.

Theraulaz, G., Goss, S., Gervet, J. y Deneubourg, J. (1991b). Task differentiation in polistes wasp colonies: a model for self-organizing groups of robots. *In The first international conference on the simulation of adaptive behavior*, 346-355. Cambridge, USA: MIT Press.

Theraulaz, G., Bonabeau, E., y Deneubourg, J. (1998). Response threshold reinforcement and division of labour in insect societies. *Proceedings of the Royal Society of London B*, 265(1), 327-332.

Toulouse S. y Wolfer, R. (2009). On the complexity of the multiple stack TSP, kSTSP. *Theory and Applications of Models of Computation*, LNCS 5532, Springer-Verlag, Berlin, 360-369.

Tracey, P. y Clark, G. (2003). Alliances, networks and competitive strategy: rethinking cluster of innovation. *Grow and Change*, 34(1), 1-16.

Ulieru, M., Walker, S., y Brennan, R. (2001). Holonic enterprise as a collaborative information ecosystem. *In Proc. Workshop on Holons: Autonomous and Cooperating Agents for Industry.*

Vapnik, V. (1995). *The Nature of Statistical Learning Theory.* Springer. N.Y. 1995.

Vapnik, V., Golowich, S.E. y Smola, A.J. (1996). Support vector method for function approximation, regression estimation, and signal processing. *Advances in Neural Information Processing Systems*, 9, 281-287.

Vapnik, V. (1998). *Statistical Learning Theory*. Wiley. New York, USA.

Velásquez, J.D., Olaya, Y., Franco y C.J. (2010). Predicción de Series Temporales usando Máquinas de Vectores Soporte. *Ingeniare. Revista chilena de ingeniería*. 18, 64-75.

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

Vinowski, S. (1997). CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. *IEEE Communications Magazine*, 14(2).

Voos, H. (2006). Agent-Based Distributed Resource Allocation in Technical Dynamic Systems. *In Proceedings of the IEEE Workshop on Distributed intelligent Systems: Collective intelligence and Its Applications*, 157-162. IEEE Computer Society, Washington, DC.

Wagner, J. y Hollenbeck, J. (2004). *Comportamiento Organizativo*. Thomson.

Waibel, M., Floreano, D., Magnenat, S. y Keller, L. (2006). Division of labor and colony efficiency in social insects: effects of interactions between genetic architecture, colony kin structure and rate of perturbations. *Proceedings of the Royal Society B*, 273(1), 1815-1823.

Walker, G., Kogut, B. Shan, W. (1997). Social capital, structural holes and the formation of an industry network. *Organization Science*, *8*(2), 109-125.

Watson, R., Ficici, S. y Pollack, J. (1999). Embodied evolution: A response to challenges in evolutionary robotics. In J.Wyatt & J. Demiris (Eds.), *Eighth european workshop on learning robots*, 14-22. Lausanne, Switzerland: Springer Verlag.

Watson, R., Ficici, S. y Pollack, J. (2002). Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and autonomous systems*, 39(1), 1–18.

Weber, M. (1947). *The Theory of social and economic organization.* Oxford University Press. N.Y.

Wenseleers, T., Ratnieks, F. y Billen, J. (2003). Caste fate conflict in swarm-founding social hymenoptera: an inclusive fitness analysis. *Evolutionary Biology*, 16(1), 647–658.

Whiteson, S., Kohl, N., Miikkulainen, R., y Stone, P. (2003). Evolving keep-away soccer players through task decomposition. *In proceeding of the genetic and evolutionary computation conference*, 356-368. Chicago, USA: AAAI Press.

Wood, M. F. (2000). *Multiagent Systems Engineering: A Methodology for Analysis and Design of Multiagent Systems*. Tesis Doctoral, Air Force Institute of Technology.

Woodward, J. (1958). *Management and Technology*. Londres. H.M.S.O.

Wooldridge, M. y Jennings, N. R. (1995). Agent Theories, Architectures, and Languages: A Survey. *In Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages,* eds. M. J. Wooldridge and N. R. Jennings, 1-39. Berlin: Springer- Verlag.

Wooldridge, M. y Jennings, N. R. (2000). *Agent-oriented software engineering.* Handbook of Agent Technology.

Wooldridge, M., Jennings, N., y Kinny, D. (2000). The Gaia methodology for agent oriented analysis and design. *Journal of Autonomous Agent and Multi-Agent Systems,* 3, 285–312.

Wooldridge, M. (2002). *An Introduction to Multiagent Systems*. Liverpool. John Wiley & Sons Ltd.

Xu, H. y Shatz, S. (2003). ADK: An agent development kit based on a formal design model for multiagent systems. *Automated Software Engineering,* 10, 337–365.

Yadgar, O., Kraus, S., y Ortiz, C. (2003). Scaling up distributed sensor networks: cooperative large-scale mobile-agent organizations. *In Distributed Sensor Networks: A multiagent perspective*, 139-183. Kluwer Academic Publishers.

Zambonelli, F., Jennings, N. R. y 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.

Zambonelli, F., Jennings, N. y Wooldridge, M. (2003). Developing Multiagent Systems: The Gaia Methodology. *ACM Transactions on Software Engineering and Methodology*, 12, 317-370.

Zato, C., de Luis, A., Bajo, J., de Paz, J.F., Corchado, J.M. (2011). Dynamic model of distribution and organization of activities in multi-agent systems. *Logic Journal of IGPL.*

Zhang, X. y Norrie, D. (1999). Holonic control at the production and controller levels. *In Proc. 2nd Int. Workshop on Intelligent Manufacturing Systems*, 215-224.

Zhang, Y., Martinoli, A., y Antonsson, E. (2003). Evolutionary design of a collective sensory system. *In The 2003 aaai spring symposium on computational synthesis*, 283-290). Stanford, USA: AAAI Press.

Zimmermann O., Schlimm N., Waller G. y Pestel M. (2005). Analysis and Design Techniques for Service-Oriented Development and Integration. *INFORMATIK*, 606-611.

Zipkin, P. (2000). *Foundations of Inventory Management.* McGraw-Hill Higher Education. New York.

ANEXO A: GAIA Y AML PARA LA INGENIERÍA DEL SOFTWARE ORIENTADA A AGENTES

A.1. GAIA

La metodología Gaia (Wooldridge *et al.*, 2000) permite realizar el análisis y el diseño de sistemas multiagente estructurándolos como sistemas organizacionales donde los agentes que los componen juegan una serie de roles y cooperan entre ellos para lograr el objetivo común del sistema. Esta metodología consta de dos fases: análisis y diseño, las cuales se muestran en la **Figura A.1.** Gaia provee el análisis y diseño de sistemas multiagente a un alto nivel, el cual puede ser complementado a través de AML (Cervenka y Trencansky, 2004; 2007; Cervenka, 2005), descrito en la sección A.2 de este apéndice.

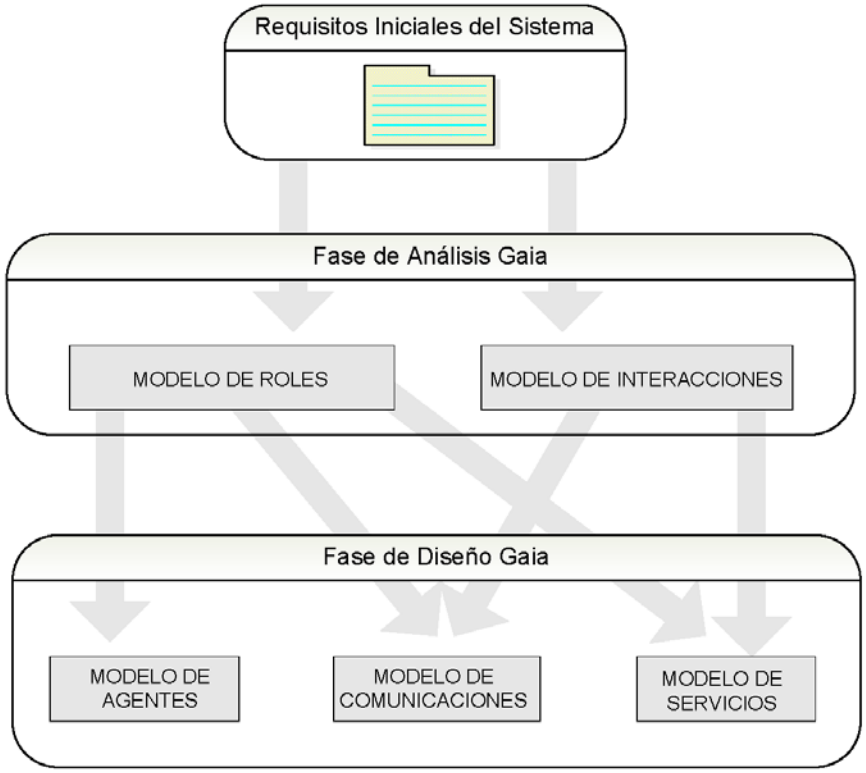


Figura A.1 Modelos en Gaia y sus relaciones

A.1.1. Fase de Análisis GAIA

El objetivo de esta fase consisten en obtener una idea general de la estructura del sistema, sus componentes y las relaciones existentes entre los mismos, así pues, comienza la fase de análisis Gaia en la que se definen los modelos de roles e interacciones (Wooldridge *et al.*, 2000).

1. **Modelo de Roles:** Un rol es definido como una descripción abstracta de las funcionalidades que ha de alcanzar una entidad en el sistema, con lo que, a través del modelo de roles se identifican los roles claves del sistema y por tanto sus funcionalidades (Wooldridge *et al.*, 2000). Para cada uno de los roles han de ser especificadas sus responsabilidades, permisos, actividades y protocolos.

    - **Responsabilidades:** Este atributo se encarga de determinar la funcionalidad del propio rol, da a conocer el ciclo de vida del mismo y las condiciones que el agente ha de satisfacer en su ejecución. Estas responsabilidades pueden ser divididas en dos tipos (Wooldridge *et al.*, 2000):

- **Propiedades *"Liveness":*** Describen las funcionalidades de los agentes y los estados del mismo dadas una serie de condiciones del entorno.

- **Propiedades *"Safety":*** Son las condiciones y restricciones que un agente ha de cumplir durante su ejecución.

- **Permisos:** Determinan que recursos están disponibles para cada uno de los roles del sistema y los límites de dichos recursos.

- **Actividades:** Representan las acciones de cada rol, que los agentes pueden realizar sin necesidad de interactuar con otros agentes.

- **Protocolos:** Definen la forma de interacción entre los agentes. Estas interacciones son definidas con mayor detalle en el modelo de interacción, el cual describe los protocolos entre roles.

2. **Modelo de Interacciones:** En un sistema multiagente en forma de organización han de existe una cantidad importante de dependencias y relaciones entre roles (Wooldridge *et al.*, 2000). A través de este modelo se indican estas dependencias y relaciones, son definidos los protocolos para cada interacción especificando el propósito de la misma con el fin de establecer un orden de intercambio de mensajes. Para definir los protocolos es necesaria la definición de los siguientes atributos representados en la **Figura A.2**:

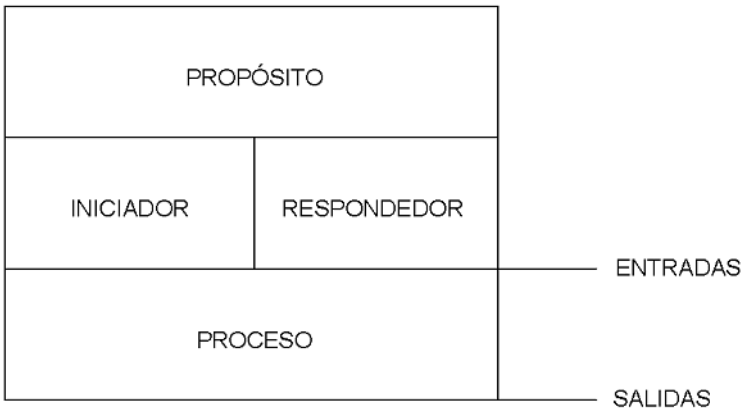


Figura A. 2 Representación de protocolo en el modelo de interacciones Gaia

- Propósito: Breve descripción de la interacción.
- Iniciador: Rol o roles que empiezan la interacción.

- Respondedor: Rol o roles con los que interactúa el iniciador.
- Entradas: Información que utiliza el iniciador en la interacción.
- Salidas: Información que proporciona el respondedor.
- Proceso: Breve descripción de las acciones que son realizadas durante la interacción.

A.1.2. Fase de Diseño GAIA

Una vez finalizada la fase de análisis se da paso a la fase de diseño, a través de la cual, se pretende reducir el nivel de abstracción hasta niveles que permitan la aplicación de otro tipo de metodologías para iniciar la implementación del sistema. En esta fase se modela en detalle la cooperación entre los agentes en la consecución de los objetivos globales de la organización. La fase de diseño consta de tres modelos: agentes, servicios y comunicación, que son descritos a continuación (Wooldridge *et al.,* 2000).

1. **Modelo de agentes:** En este modelo se describen los tipos de agentes que componen el sistema y la cardinalidad de los mismos, entendiendo por ésta el número de instancias de cada uno de ellos en tiempo de ejecución. A cada tipo de agente le es asignado uno o más roles los cuales son estructurados de forma jerárquica. En la **Tabla A.1** se muestran los símbolos utilizados para definir estas cardinalidades.

Tabla A.1 Simbología para la representación de la cardinalidad

Representación	Significado
n	***n*** instancias
m..n	Entre ***m*** y ***n*** instancias
*	***0*** o más instancias
+	***1*** o más instancias

2. **Modelo de servicios:** Un servicio es considerado un bloque de funcionalidad derivado de las actividades y protocolos de los roles del sistema. En este modelo se definen los servicios asociados a cada agente de forma exclusiva.

3. **Modelo de Comunicación o Familiaridad:** En este modelo se definen las comunicaciones entre los diferentes agentes del sistema a partir de los modelos de interacción y de agentes.

La metodología Gaia se centra en la consecución de un objetivo común del sistema. Los modelos que se obtienen a través de esta metodología tienen un alto nivel de abstracción, lo cual dota de independencia a nivel de plataforma y de lenguaje de programación. Este aspecto implica que a la hora de llevar a cabo la implementación del sistema se ha de hacer uso de metodologías y herramientas que permitan una reducción de dicho nivel de abstracción, especificando con mayor detalle la fase de diseño, y así poder realizar una correcta implementación. Una de estas herramientas es el Lenguaje de Modelado de Agentes, AML, (Cervenka y

Trencansky, 2004; 2007; Cervenka, 2005), el cual se describe en el apartado A2 de este apéndice.

A.1.3. Extensión de GAIA: GAIAEXOA

Cabe mencionar la existencia de una extensión de la metodología Gaia, denominada GaiaEXOA (Zambonelli *et al.*, 2003), a través de la cual se permite el diseño de sistemas organizativos abiertos. Esta extensión sigue manteniendo las dos fases de la metodología Gaia, análisis y diseño. En la fase de análisis se especifica el modo de trabajo de la organización de agentes, para lo cual se han de identificar los objetivos organizacionales y el comportamiento esperado de la organización. Esta metodología permite la división de la organización de agentes en unidades suborganizacionales de forma que cada una de ellas persiga un subobjetivo que representa una parte del objetivo global de la organización.

Para alcanzar los objetivos organizacionales, GaiaEXOA amplia tres conceptos:

1. **Modelo del entorno:** A través de este modelo se especifican los recursos que forman parte des sistema, así como sus entidades y su forma de interactuar en el sistema.

2. **Modelo de reglas:** Mediante el cual, se insertan las restricciones en la organización, definiendo las condiciones de participación de los agentes dentro de esta.

3. **Modelo de estructura organizacional:** A través del cual se permite el modelado de la arquitectura global del sistema.

La ampliación de estos modelos implica la modificación de las fases de análisis y diseño. GaiaEXOA modifica la fase de análisis añadiendo una identificación de los objetivos organizacionales, el propio modelo del entorno, un modelo preliminar de roles y un modelo preliminar de interacción con reglas organizacionales. En la fase de diseño se añade el modelo de estructura organizacional a partir del cual se obtiene la arquitectura global del sistema.

A.2. Lenguaje de Modelado de Agentes (AML)

El lenguaje de modelado de agentes *(AML)* (Cervenka y Trencansky, 2004; 2007) es un lenguaje visual y semi-formal (Cervenka *et al.*, 2005) para el análisis, diseño y modelado de sistemas multiagente. AML puede considerarse como una extensión de UML 2.0, mediante el cual, pueden ser modeladas las características de los sistemas multiagente en cuanto a su análisis y diseño, facilitando así, el proceso de ingeniería en el desarrollo de los mismos.

El contexto de aplicación de AML es el desarrollo explícito de sistemas multiagente, aunque también pueden ser modelados procesos de negocio, sistemas organizacionales, sistemas robóticos, etc. De forma general AML puede utilizarse cuando se pretende el modelado de sistemas de los siguientes tipos (Cervenka *et al.*, 2005):

1. Entidades autónomas, concurrentes y/o asíncronas.
2. Entidades con capacidad de interacción con su entorno.
3. Entidades que hacen uso de servicios externos.
4. Estructuras sociales, tales como organizaciones.
5. Entidades con características deliberativas, unitarias u organizadas

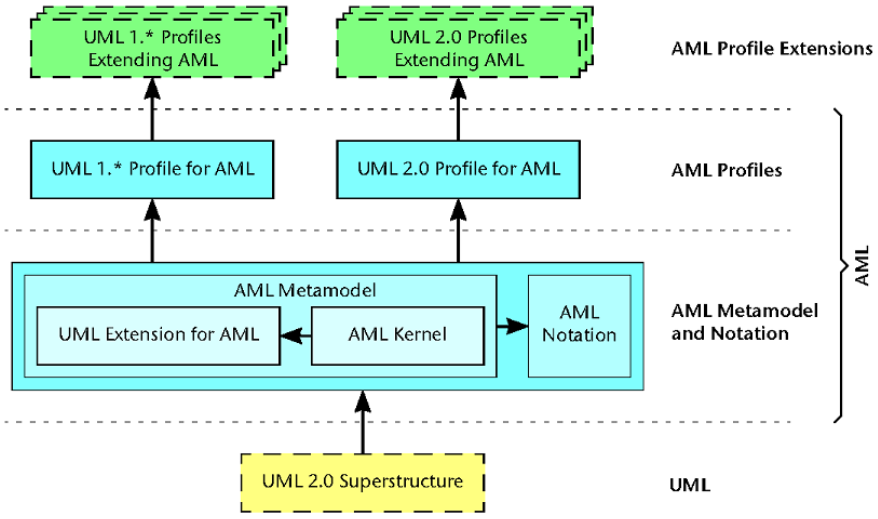


Figura A. 3 Estructura de AML, tomada de (Cervenka y Trencansky, 2004)

En cuanto a las características principales de AML se presentan las siguientes (Cervenka *et al.*, 2005):

1. Está construido a partir de fundamentos técnicos consolidados.
2. Integra buenas prácticas de la Ingeniería del Software Orientada a Agentes (AOSE).
3. Está bien especificado y documentado.
4. Es consistente internamente en cuanto a las perspectivas conceptual, sintáctica y semántica.
5. Es versátil y fácil utilizar.
6. Es independiente de las teorías del desarrollo de software y del entorno de implementación.
7. Está soportado por herramientas CASE.

AML está diseñado para el análisis y diseño de sistemas multiagente de forma que (Cervenka *et al.*, 2005):

1. Soporta el análisis y diseño de procesos mentales de los agentes.
2. Soporta los niveles de abstracción necesarios para modelar la arquitectura y el comportamiento de los sistemas multiagente, tanto a nivel unitario, como a nivel organizacional.

La decisión de adoptar una combinación de la metodología Gaia, descrita en la sección A1, complementada y representada a través del Lenguaje de Modelado de Agentes *(AML)*, hace que características que provee este lenguaje no se utilicen en la definición del modelado de SCODA, haciendo uso solamente de aquellas que tengan la capacidad de representar los modelos de alto nivel Gaia, y de las cuales permitan una representación básica del sistema.

A continuación se describen los diferentes modelos que presenta AML para el modelado de sistemas multiagente (Cervenka y Trencansky, 2004; 2007), utilizados en este trabajo.

A.2.1. Modelado de la Arquitectura del Sistema

Este modelo permite a los desarrolladores realizar el diseño estructural del sistema atendiendo a aspectos como los tipos de entidades que componen el sistema, los aspectos sociales existentes entre los elementos del sistema, y el despliegue del sistema multiagente.

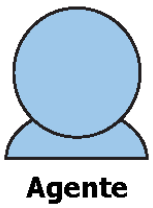


Figura A. 4 Representación de una entidad de tipo agente en AML

- **Entidades del sistema:** Una entidad representa un objeto que existe en el sistema y es independiente de otros objetos. En AML se definen tres tipos: Agentes, recursos y entornos internos del sistema. En la **Figura A.4.** se representa una entidad de tipo agente.

- **Aspectos Sociales:** A través de estos aspectos sociales, AML define diversos elementos para el modelado de las relaciones organizacionales de los sistemas multiagente, incluyendo la estructura de las mismas, características sociales entre entidades del sistema y ciertos aspectos de su comportamiento.

- **Despliegue y movilidad:** Una de las características de AML es que permite el modelado del despliegue de un sistema multiagente en un entorno físico especializado para agentes, en el cual se tiene en cuenta la capacidad de movilidad de estos agentes y su comportamiento.

A.2.2. Modelado del Comportamiento

AML permite realizar el modelado del comportamiento del sistema y de sus componentes en cuanto a la capacidad de razonamiento de los mismos sobre eventos u operaciones a realizar. Además, a través de AML es posible modelar las interacciones existentes entre entidades o grupos de entidades a través de protocolos de interacción, para lo que provee una serie de diagramas extendidos de los diagramas de interacción de UML, y permiten modelar las interacciones entre agentes y grupos de agentes a través de mensajes y señales.

Otro aspecto importante dentro de AML es la capacidad que provee para modelar servicios que pueden ser consumidos o proveídos por las entidades del sistema. Los servicios se encapsulan en bloques de funcionalidad que las entidades pueden ofertar o consumir.

A.2.3. Modelado del Contexto

AML provee la capacidad de modelar el entorno en el que se desenvuelven los agentes. Como se muestra en la **Figura A.5.,** y de acuerdo a situaciones, que puedan ocurrir en tiempo de ejecución, se permite el modelado de situaciones particulares en momentos determinados los cuales se denominan contextos.

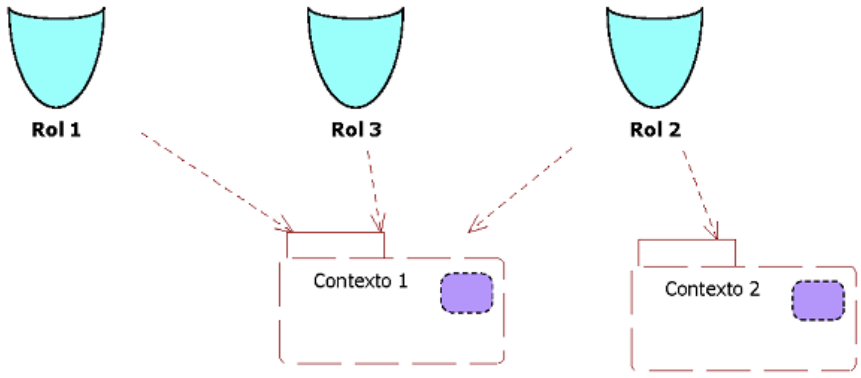


Figura A. 5 Representación del modelado del contexto sobre AML

A.2.4. Tipos de Diagramas Utilizados

AML provee de una serie de diagramas a través de los cuales es posible representar las relaciones entre diversas entidades y las interacciones entre las mismas.

- **Diagrama de Sociedad:** Se utiliza para representar la vista global de la arquitectura del sistema multiagente en términos organizativos y las relaciones existentes entre las diferentes unidades organizativas. En la **Figura A.6.** se muestra un ejemplo de diagrama de sociedad donde pueden verse las relaciones existentes entre los diferentes roles y las diferentes unidades organizativas a las que pertenecen.

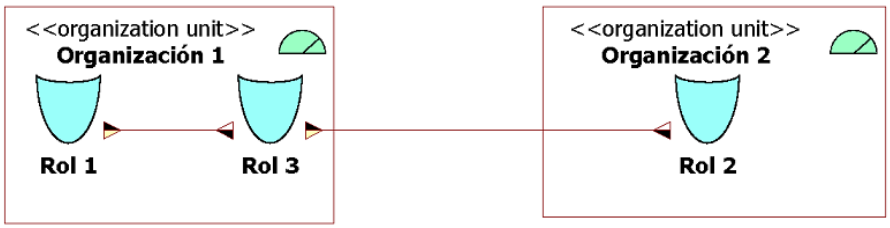


Figura A. 6 Ejemplo de diagrama de sociedad sobre AML

- **Diagrama de Despliegue:** A través de este diagrama se muestra la situación real del sistema, con las diferentes entidades que entran en juego en tiempo de ejecución, sus relaciones y los recursos necesarios para llevar a cabo sus fines.

- **Diagrama de Comportamientos:** Muestra los comportamientos asociados a los agentes, de forma que estos comportamientos definen la funcionalidad de dichos agentes. Estos comportamientos son especificados uno por uno dando a conocer los parámetros necesarios en su ejecución. Este tipo de diagrama es empleado para modelar los comportamientos. En la **Figura A.7.** se muestra un ejemplo de diagrama de comportamientos donde se aprecian dos comportamientos asociados a un agente, que posteriormente son especificados por de forma unitaria a través de sus atributos y operaciones.

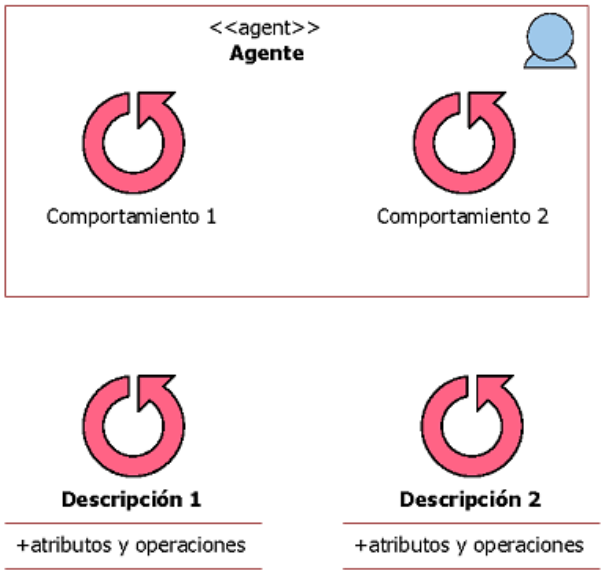


Figura A. 7 Diagrama de comportamiento de un agente sobre AML

- **Diagrama de Secuencia de Protocolos:** Especifica los protocolos de interacción en forma de diagrama de secuencia, mostrando los patrones de comunicación correspondientes a los mensajes que son utilizados en la comunicación de agentes o roles. A través de este tipo de diagramas se aprecia el comportamiento de los agentes o roles en cuanto a su interacción se refiere.

- **Diagrama de Colaboración de Protocolos:** Especifica los protocolos de interacción en forma de diagrama de colaboración. Este tipo de diagrama, al igual que el diagrama de secuencia de protocolos, puede ser aplicado tanto a agentes como a roles dentro del sistema.

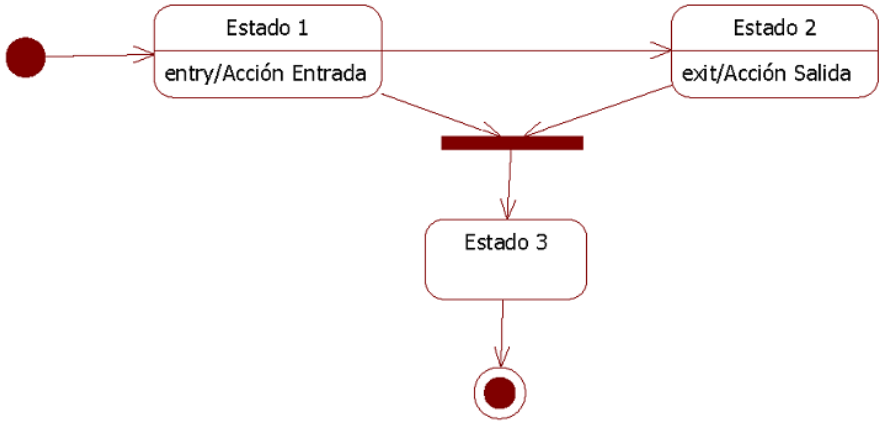


Figura A. 8 Ejemplo de diagrama de estados sobre AML

- **Diagrama de Estados:** Este tipo de diagrama no está incluido como diagrama específico de AML, sino heredado de UML, sin embargo se ha creído conveniente su utilización, ya que a través del mismo puede definirse el comportamiento interno de cada agente, mostrando cada uno de los estados por lo que pasa, y los eventos generados por los mismos, tal y como se muestra en la **Figura A.8.**

ANEXO B: MODELADO CON GAIA Y AML

B.1. MODELADO CON GAIA Y AML

Rol: Comunicaciones

Descripción: Gestiona las comunicaciones desde y hacia el sistema. Selecciona al controlador de comunidad correspondiente. Realiza tareas de control sobre el sistema.

Protocolos y Actividades: Conexión Aplicación, Desconexión Aplicación, Recibir Solicitud, Analizar Solicitud, Enviar Respuesta, Testear Calidad, Reiniciar Calidad, Informar Calidad, Seleccionar Comunidad, Petición Comunidad, Respuesta Comunidad

Permisos:

Lectura: Testear Calidad, Seleccionar Comunidad, Analizar Solicitud

Actualización: Reiniciar Calidad

Responsabilidades:

Comunicaciones = (Conexión Aplicación · ((Recibir Solicitud · Analizar Solicitud · Seleccionar Comunidad · Petición Comunidad · Respuesta Comunidad · Enviar Respuesta) | (Testear Calidad$^{\omega}$ · Reiniciar Calidad*) | Informar Calidad$^{\omega}$) · Desconexión Aplicación)$^{\omega}$

Figura B.1 Representación Gaia del Rol Comunicaciones

Rol: Calidad

Descripción: Gestiona la supervisión del sistema, asumiendo el control de los demás roles y de los servicios. Gestiona estadísticas de funcionamiento del sistema.

Protocolos y Actividades: Recibir Información, Enviar Información, Testear Controlador, Testear Comunicaciones, Testear Servicio, Reiniciar Controlador, Reiniciar Comunicaciones, Reiniciar Servicio, Modificar Estadística, Análisis Estadística, Reiniciar Petición

Permisos:

Lectura: Testear Controlador, Testear Comunicaciones, Testear Servicios

Actualización: Reiniciar Controlador, Reiniciar Comunicador, Reiniciar Servicio, Reiniciar Petición.

Responsabilidades:

Calidad = (((Recibir Información^{+} | Enviar Información^{+}) | ((Testear Controlador$^{\omega}$ | Testear Comunicaciones$^{\omega}$ | Testear Servicio$^{\omega}$) · (Reiniciar Controlador* | Reiniciar Comunicaciones*| Reiniciar Servicios*)) | Reiniciar Petición*) · Modificar Estadística^{+} · Análisis Estadística$^{+)\omega}$

Figura B.2 Representación Gaia del Rol Calidad

Rol: Controlador

Descripción: Gestiona las Comunidades Inteligentes Especializadas compuestas por el los roles planificador y ejecutor. Ejerce labores de control del sistema junto a los roles comunicaciones y calidad.

Protocolos y Actividades: Recibir Petición, Enviar Respuesta, Enviar Petición, Recibir Respuesta, Testear Calidad, Reiniciar Calidad, Informar Calidad, Crear Equipo, Destruir Equipo, Testear Equipo, Reiniciar Equipo, Reiniciar Petición

Permisos:

Lectura: Testear Calidad, Testear Equipo

Actualización: Reiniciar Calidad, Reiniciar Equipo, Crear Equipo, Destruir Equipo

Responsabilidades:

Controlador = ((Recibir Petición · (Crear Equipo | (Testear Equipo$^{\omega}$ · Reiniciar Equipo*)) · Enviar Petición · Recibir Respuesta · Destruir Equipo · Enviar Respuesta) | Informar Calidad$^{\omega}$) | Reiniciar Petición | (Testear Calidad$^{\omega}$ · Reiniciar Calidad*))$^{\omega}$

Figura B.3 Representación Gaia del Rol Controlador

Rol: Planificador

Descripción: Gestiona las peticiones de servicios seleccionando los parámetros adecuados y el servicio óptimo de cada una de ellas.

Protocolos y Actividades: Recibir Petición, Enviar Respuesta, Seleccionar Servicio, Seleccionar Parámetros, Enviar Petición, Recibir Respuesta.

Permisos:
Actualización: Seleccionar Parámetros, Seleccionar Servicio, Enviar Petición

Responsabilidades:
Planificador = (Recibir Petición · Seleccionar Servicio · Seleccionar Parámetros* · Enviar Petición · Recibir Respuesta · Enviar Respuesta)$^{\omega}$

Figura B.4 Representación Gaia del Rol Planificador

Rol: Ejecutor

Descripción: Hace efectivas las peticiones a los servicios y devuelve la respuesta.

Protocolos y Actividades: Recibir Petición, Enviar Respuesta, Conexión, Desconexión, Enviar Petición, Recibir Respuesta.

Permisos:
Actualización: Enviar Respuesta

Responsabilidades:
Planificador = (Recibir Petición · Conexión · Enviar Petición · Recibir Respuesta · Desconexión · Enviar Respuesta)$^{\omega}$

Figura B.5 Representación Gaia del Rol Ejecutor

Testear Calidad		
Comunicaciones Controlador	Calidad	ACK
Se envía un mensaje de testeo al rol calidad		ACK

Figura B.6 Testear Calidad

Reiniciar Calidad		
Comunicaciones Controlador	Calidad	Parámetros Reinicio
Se reinicia la instancia que desempeña el rol calidad		Confirmación Reinicio

Figura B.7 Reiniciar Calidad

Informar Calidad		
Comunicaciones Controlador	Calidad	Resultado Actividad
Se informa de la consecución de alguna actividad al rol calidad		

Figura B.8 Informar Calidad

Petición Comunidad		
Comunicaciones	Controlador	Solicitud Servicio
El rol comunicaciones solicita un servicio al rol controlador de una Comunidad Inteligente Especializada		

Figura B.9 Petición Comunidad

Respuesta Comunidad		
Controlador	Comunicaciones	Resultado Servicio
El resultado de la ejecución de un servicio es comunicado al rol comunicaciones		

Figura B.10 Respuesta Comunidad

Recibir Información		
Comunicaciones Controlador	Calidad	Resultado Actividad
El rol calidad recibe información sobre actividades desarrolladas por otros roles del sistema		ACK

Figura B.11 Recibir Información

Enviar Información		
Calidad	Comunicaciones Controlador	Información Control
El rol calidad envía información a otros roles del sistema para corregir algún comportamiento		Acción Correctora

Figura B.12 Enviar Información

Testear Controlador		
Calidad	Controlador	ACK
Se envía un mensaje de testeo al rol controlador		ACK

Figura B.13 Testear Controlador

Testear Comunicaciones		
Calidad	Comunicaciones	ACK
Se envía un mensaje de testeo al rol comunicaciones		ACK

Figura B.14 Testear Comunicaciones

Reiniciar Controlador		
Calidad	Controlador	Parámetros Reinicio
Se reinicia la instancia que desempeña el rol controlador		Confirmación Reinicio

Figura B.15 Reiniciar Controlador

Reiniciar Comunicaciones		
Calidad	Comunicaciones	Parámetros Reinicio
Se reinicia la instancia que desempeña el rol comunicaciones		Confirmación Reinicio

Figura B.16 Reiniciar Controlador

Reiniciar Petición		
Calidad	Controlador	Solicitud Servicio
El rol calidad ordena de nuevo una petición de un servicio		

Figura B.17 Reiniciar Petición

Recibir Petición Comunicaciones		
Comunicaciones	Controlador	Solicitud Servicio
El rol controlador recibe una petición de un servicio por parte del rol comunicaciones		ACK

Figura B.18 Recibir Petición Comunicaciones

Enviar Petición Planificador		
Controlador	Planificador	Solicitud Servicio
El rol controlador envía una petición de servicio al rol planificador de su comunidad		

Figura B.19 Enviar Petición Planificador

Enviar Respuesta Comunicaciones		
Controlador	Comunicaciones	Respuesta Servicio
El rol controlador envía la respuesta de una solicitud de un servicio al rol comunicaciones		

Figura B.20 Enviar Respuesta Comunicaciones

Recibir Respuesta Planificador		
Planificador	Controlador	Respuesta Servicio
El rol controlador recibe una respuesta de un servicio por parte del rol planificador		ACK

Figura B.21 Recibir Respuesta Planificador

Crear Equipo		
Controlador	Planificador Ejecutor	Parámetros Equipo
El rol controlador crea un equipo de trabajo al recibir la solicitud de un servicio		Instancia Equipo

Figura B.22 Crear Equipo

Destruir Equipo		
Controlador	Planificador Ejecutor	Parámetros Equipo
El rol controlador destruye un equipo de trabajo al finalizar la ejecución de una solicitud de servicio		

Figura B.23 Crear Equipo

Testear Equipo		
Controlador	Planificador Ejecutor	ACK
Se envía un mensaje de testeo a los roles planificador y ejecutor de un equipo determinado		ACK

Figura B.24 Testear Equipo

Reiniciar Equipo		
Controlador	Planificador Ejecutor	Parámetros Reinicio
Se reinician las instancias que desempeñan los roles planificador y ejecutor de un equipo		Confirmación Reinicio

Figura B.25 Reiniciar Equipo

Reiniciar Petición Controlador		
Calidad	Controlador	Solicitud Servicio
El rol controlador vuelve a solicitar la ejecución de un servicio por orden del rol calidad		Confirmación Reinicio

Figura B.26 Reiniciar Petición Controlador

Recibir Petición Controlador		
Controlador	Planificador	Solicitud Servicio
El rol planificador recibe una petición de un servicio por parte del rol controlador		ACK

Figura B.27 Recibir Petición Controlador

Enviar Petición Ejecutor		
Planificador	Ejecutor	Solicitud Servicio
El rol planificador envía una petición de servicio al rol ejecutor de su comunidad		

Figura B.28 Enviar Petición Ejecutor

Enviar Respuesta Controlador		
Planificador	Controlador	Respuesta Servicio
El rol planificador envía la respuesta de una solicitud de un servicio al rol controlador		

Figura B.29 Enviar Respuesta Controlador

Recibir Respuesta Ejecutor		
Ejecutor	Planificador	Respuesta Servicio
El rol planificador recibe una respuesta de un servicio por parte del rol ejecutor		ACK

Figura B.30 Recibir Respuesta Ejecutor

Recibir Petición Planificador		
Planificador	Ejecutor	Solicitud Servicio
El rol ejecutor recibe una petición de un servicio por parte del rol planificador		ACK

Figura B.31 Recibir Petición Planificador

Enviar Respuesta Planificador		
Ejecutor	Planificador	Respuesta Servicio
El rol ejecutor envía la respuesta de una solicitud de un servicio al rol planificador		

Figura B.32 Enviar Respuesta Planificador

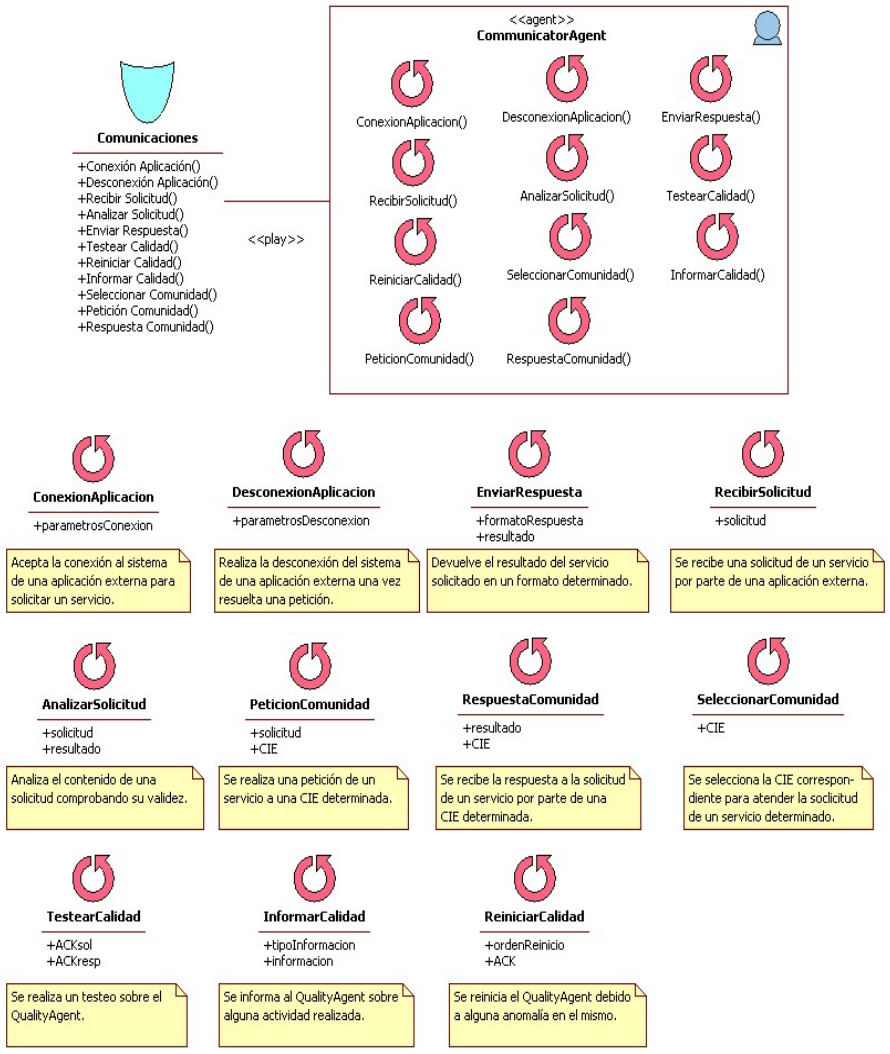


Figura B.33 Diagrama de Comportamiento del CommunicatorAgent

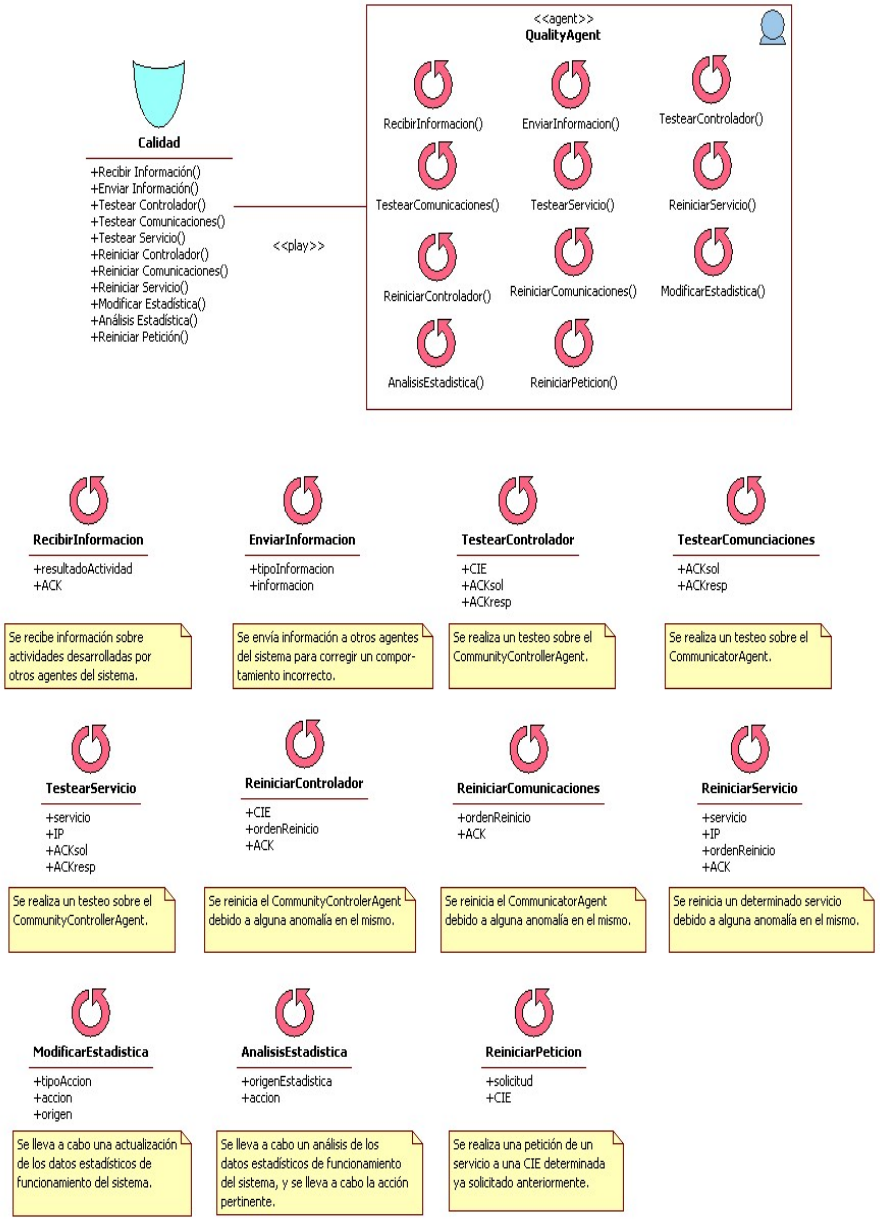


Figura B.34 Diagrama de Comportamiento del QualityAgent

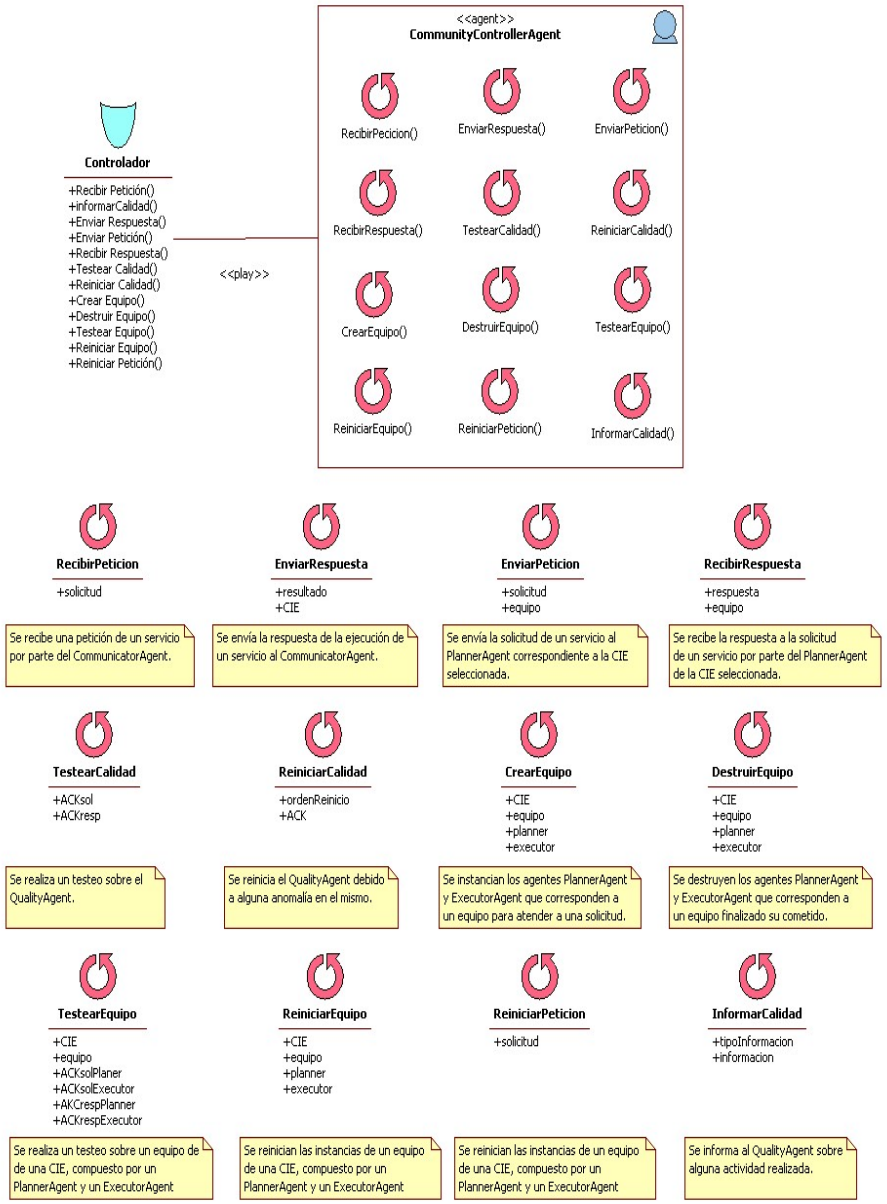


Figura B.35 Diagrama de Comportamiento del CommunityControllerAgent

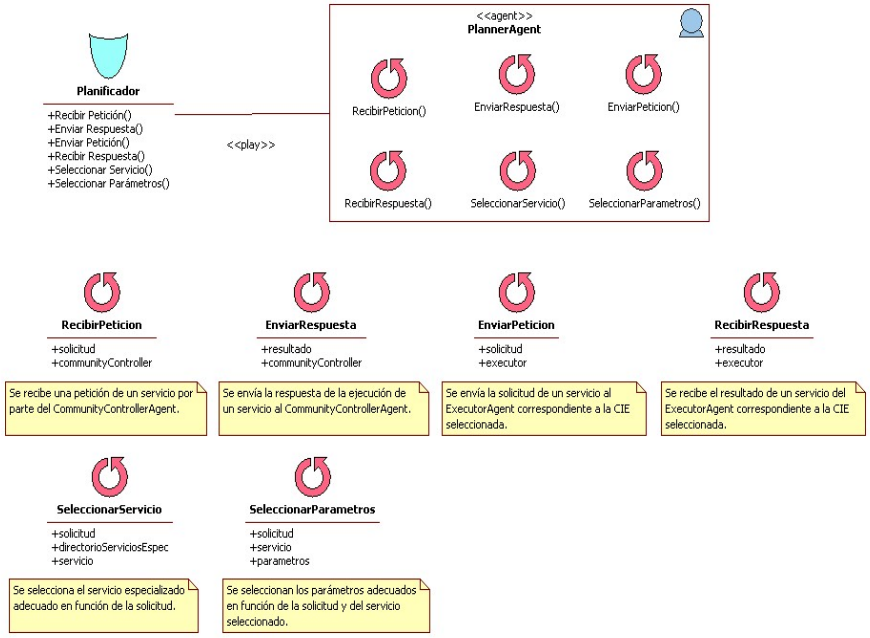


Figura B.36 Diagrama de Comportamiento del PlannerAgent

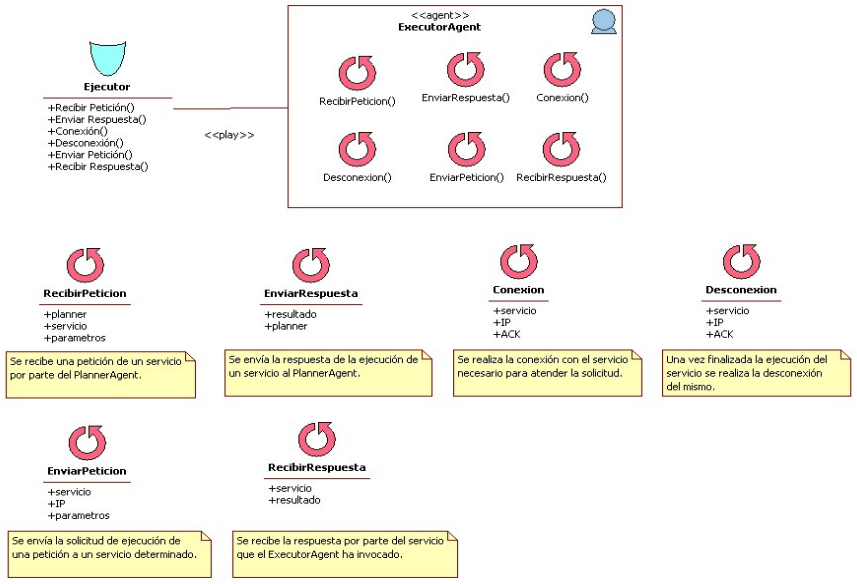


Figura B.37 Diagrama de Comportamiento del ExecutorAgent

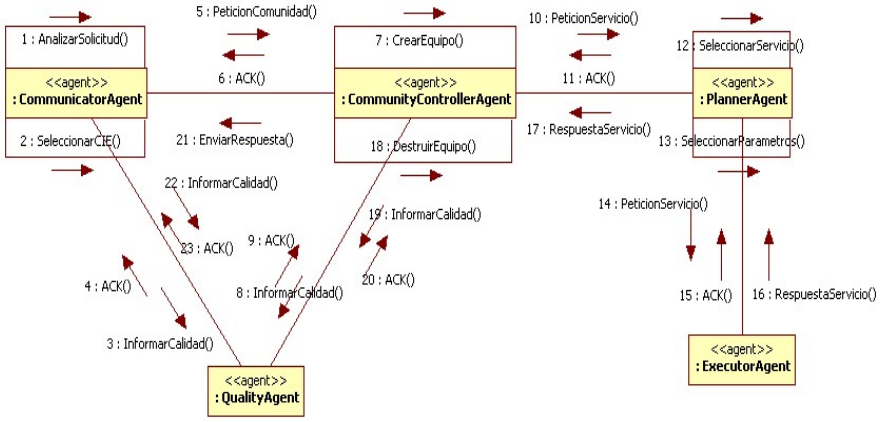


Figura B.38 Diagrama de Colaboración de Protocolos para una solicitud exitosa completa

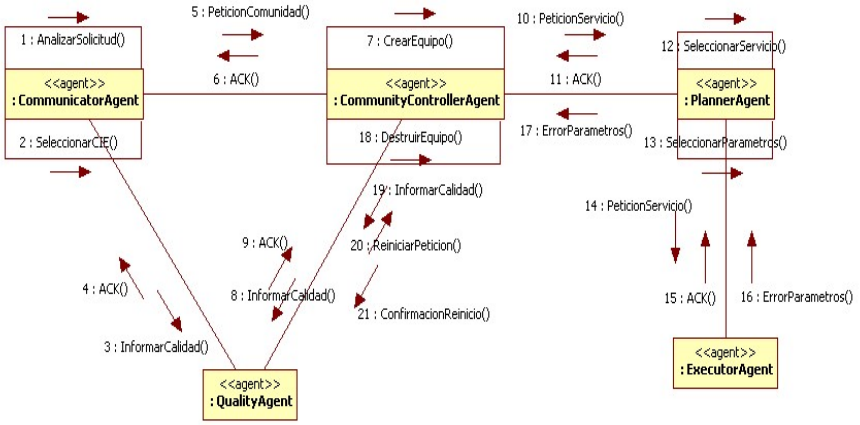


Figura B.39 Diagrama de Colaboración de Protocolos para una solicitud con error en los parámetros del servicio de una Comunidad Inteligente Especializada

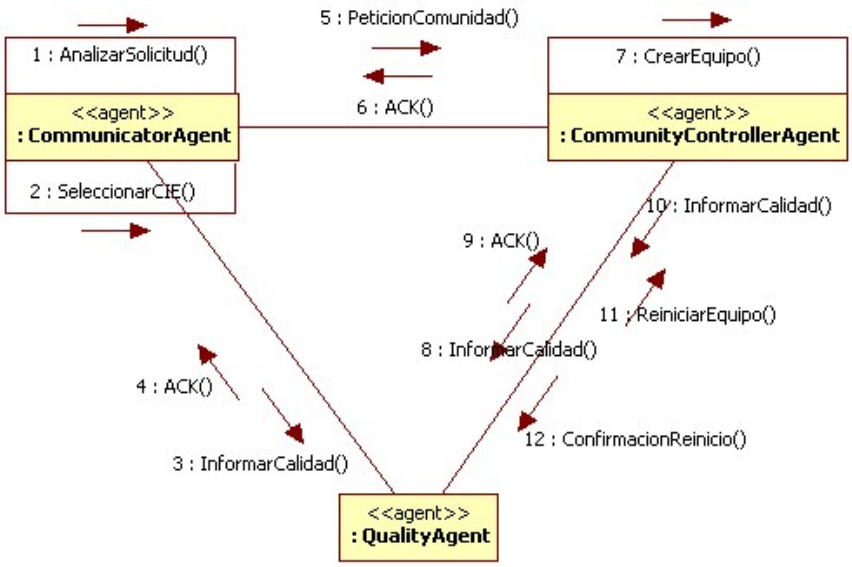


Figura B.40 Diagrama de Colaboración de Protocolos para una solicitud con error en la creación de un Equipo de una Comunidad Inteligente Especializada

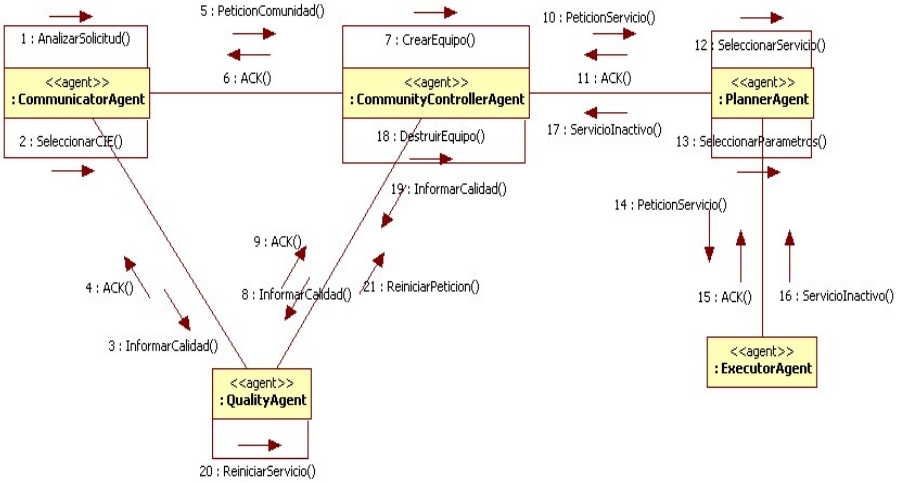


Figura B.41 Diagrama de Colaboración de Protocolos para una solicitud con error por inactividad del servicio de una Comunidad Inteligente Especializada

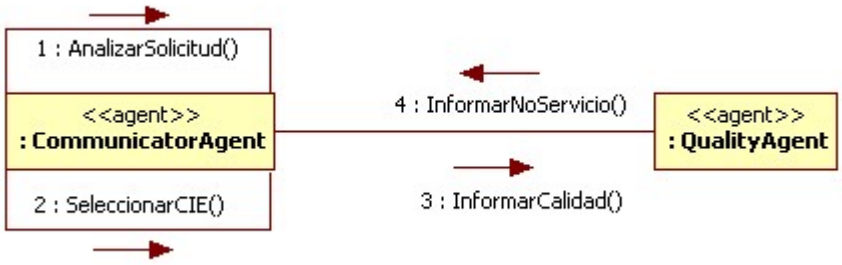


Figura B.42 Diagrama de Colaboración de Protocolos para una solicitud con error por no existir un servicio adecuado para una solicitud

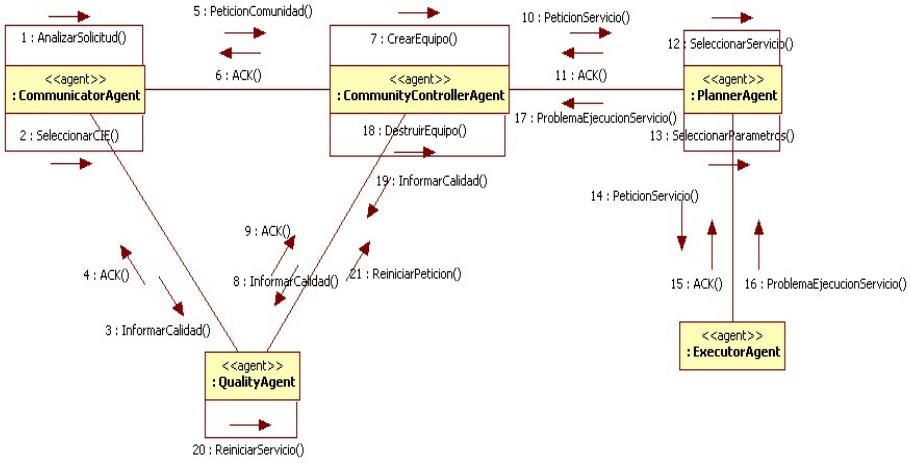


Figura B.43 Diagrama de Colaboración de Protocolos para una solicitud con error por problema de ejecución del Servicio de una Comunidad Inteligente Especializada

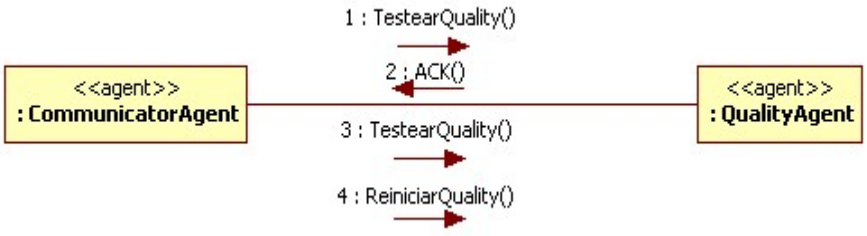


Figura B.44 Diagrama de Colaboración de Protocolos para el testeo del QualityAgent

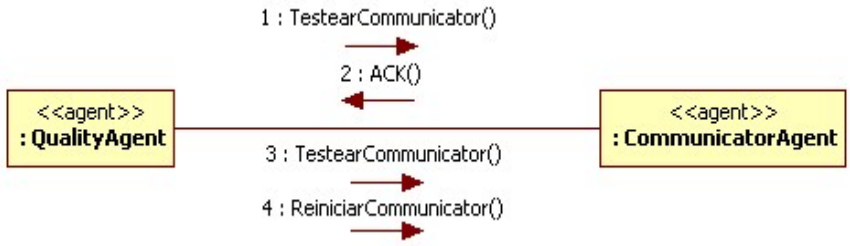


Figura B.45 Diagrama de Colaboración de Protocolos para el testeo del CommunicatorAgent

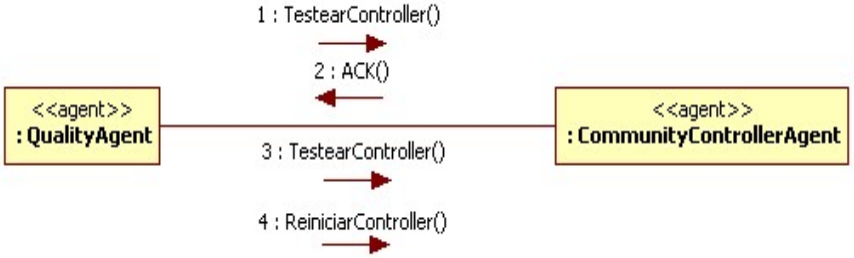


Figura B.46 Diagrama de Colaboración de Protocolos para el testeo del ControllerAgent

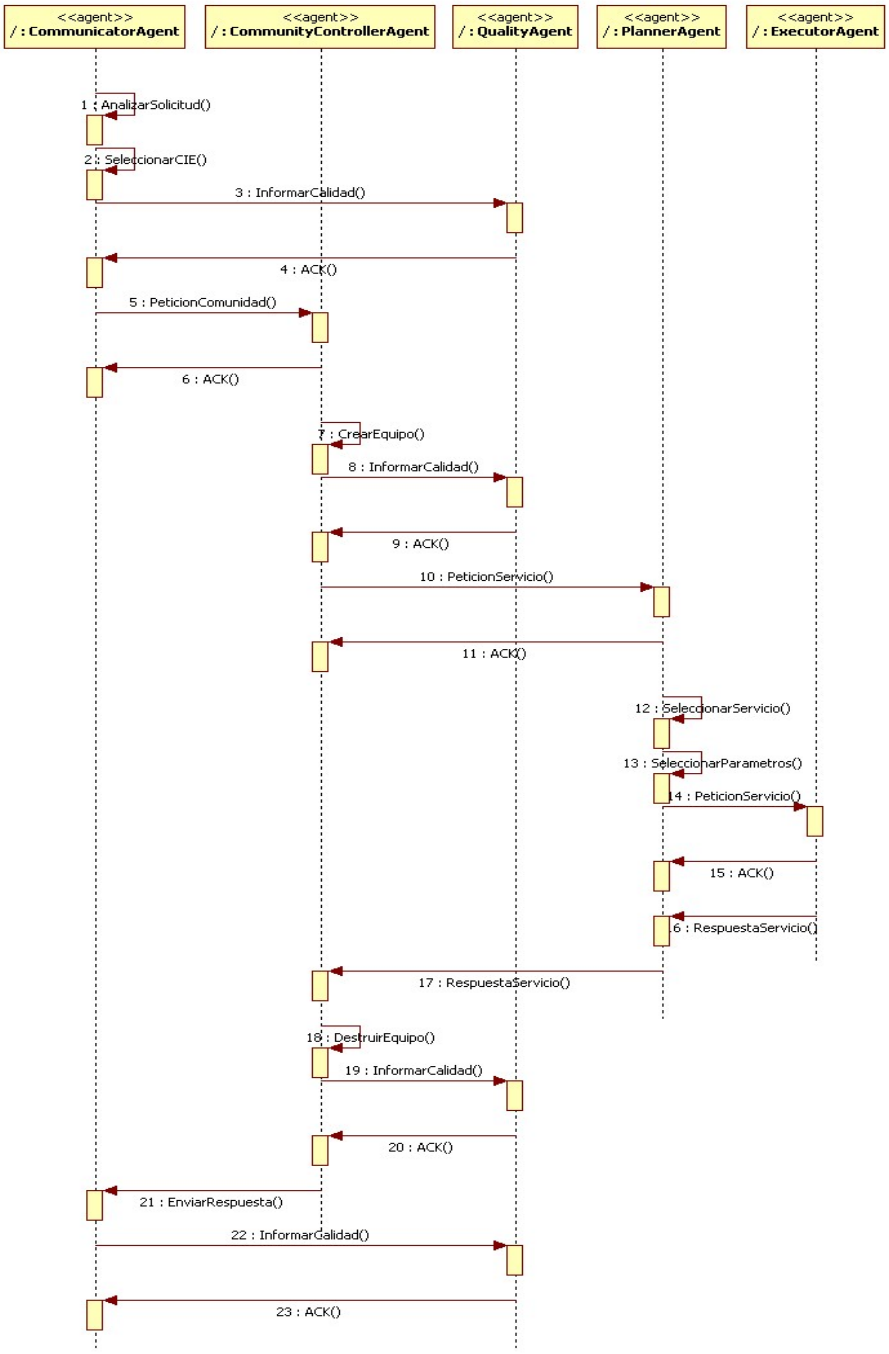


Figura B.47 Diagrama de Secuencia de Protocolos para una solicitud exitosa completa

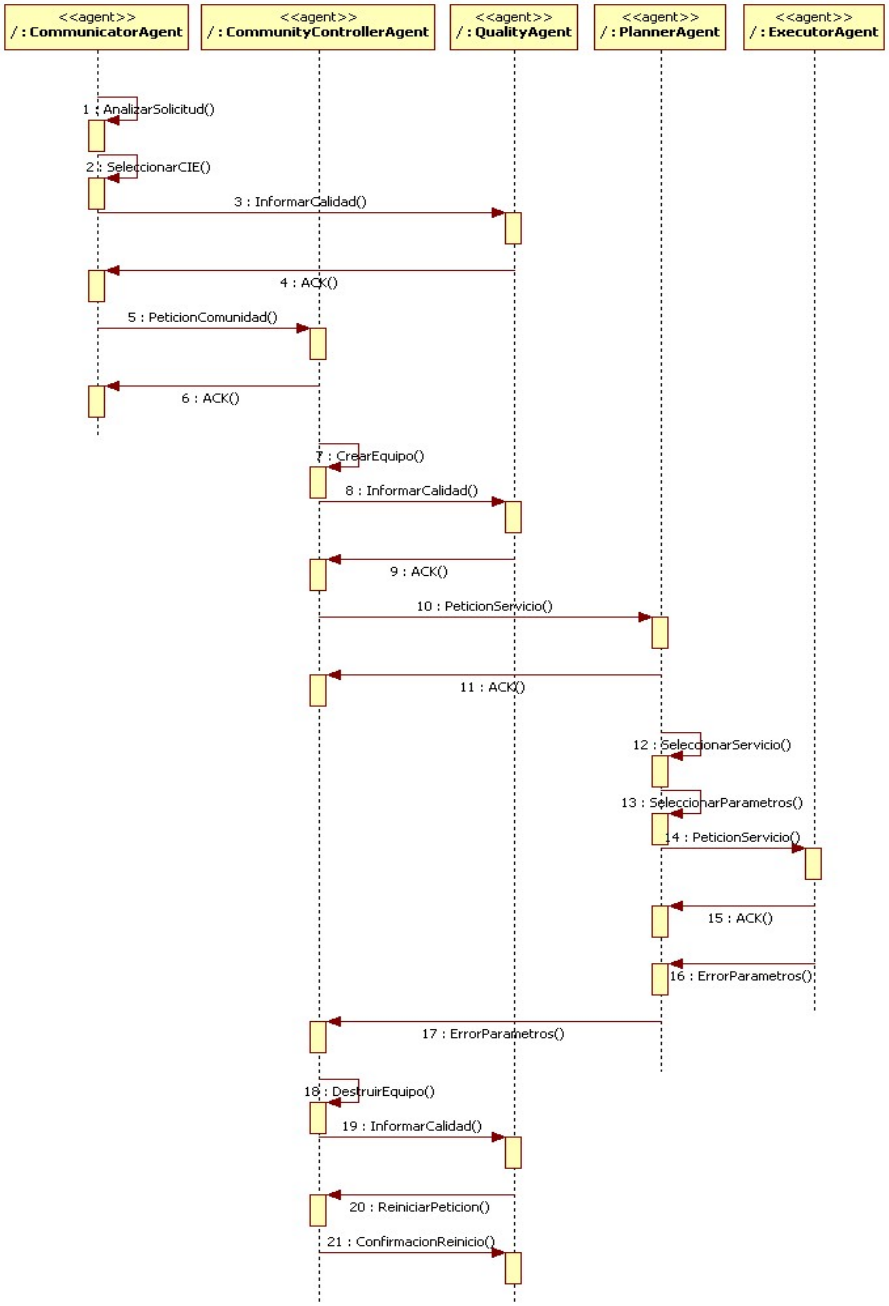


Figura B.48 Diagrama de Secuencia de Protocolos para una solicitud con error en los parámetros del servicio de una Comunidad Inteligente Especializada

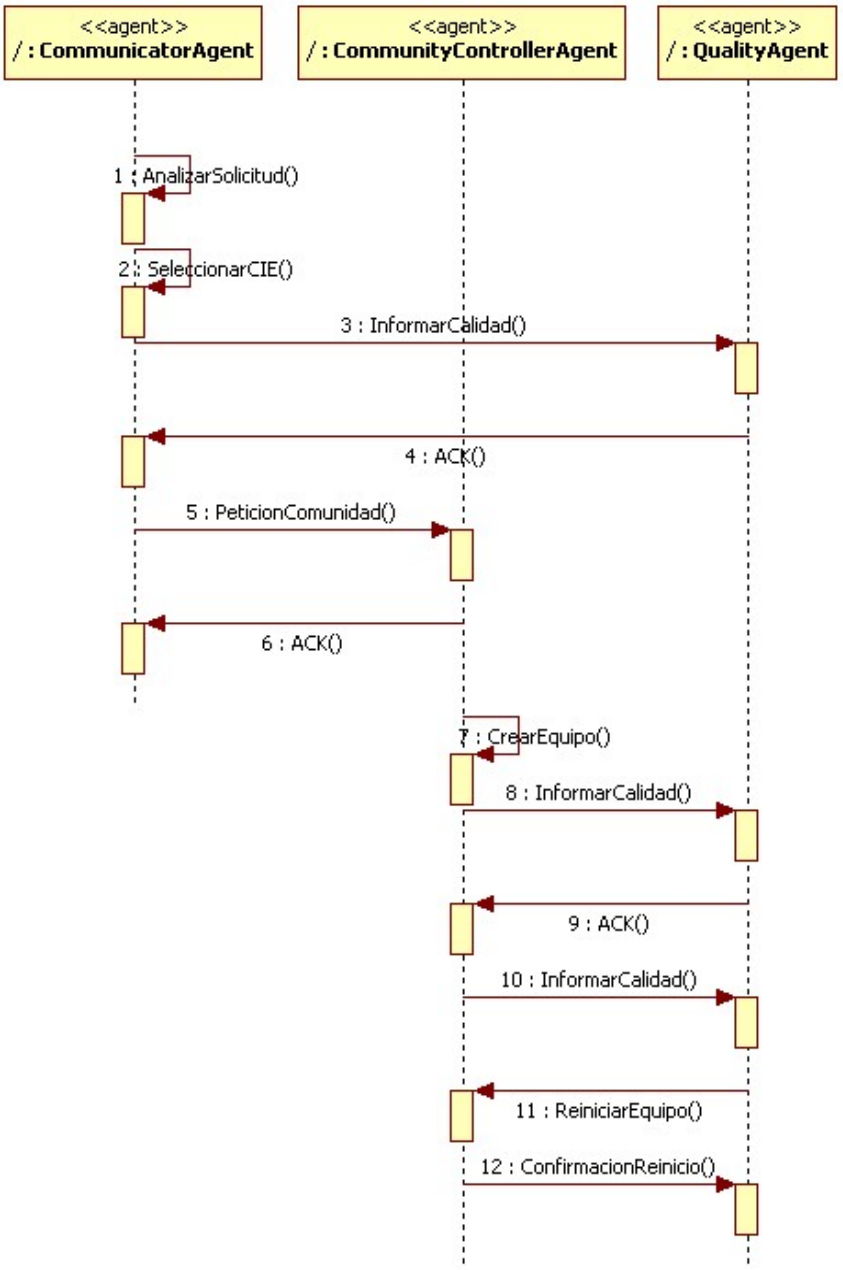


Figura B.49 Diagrama de Secuencia de Protocolos para una solicitud con error en la creación de un Equipo de una Comunidad Inteligente Especializada

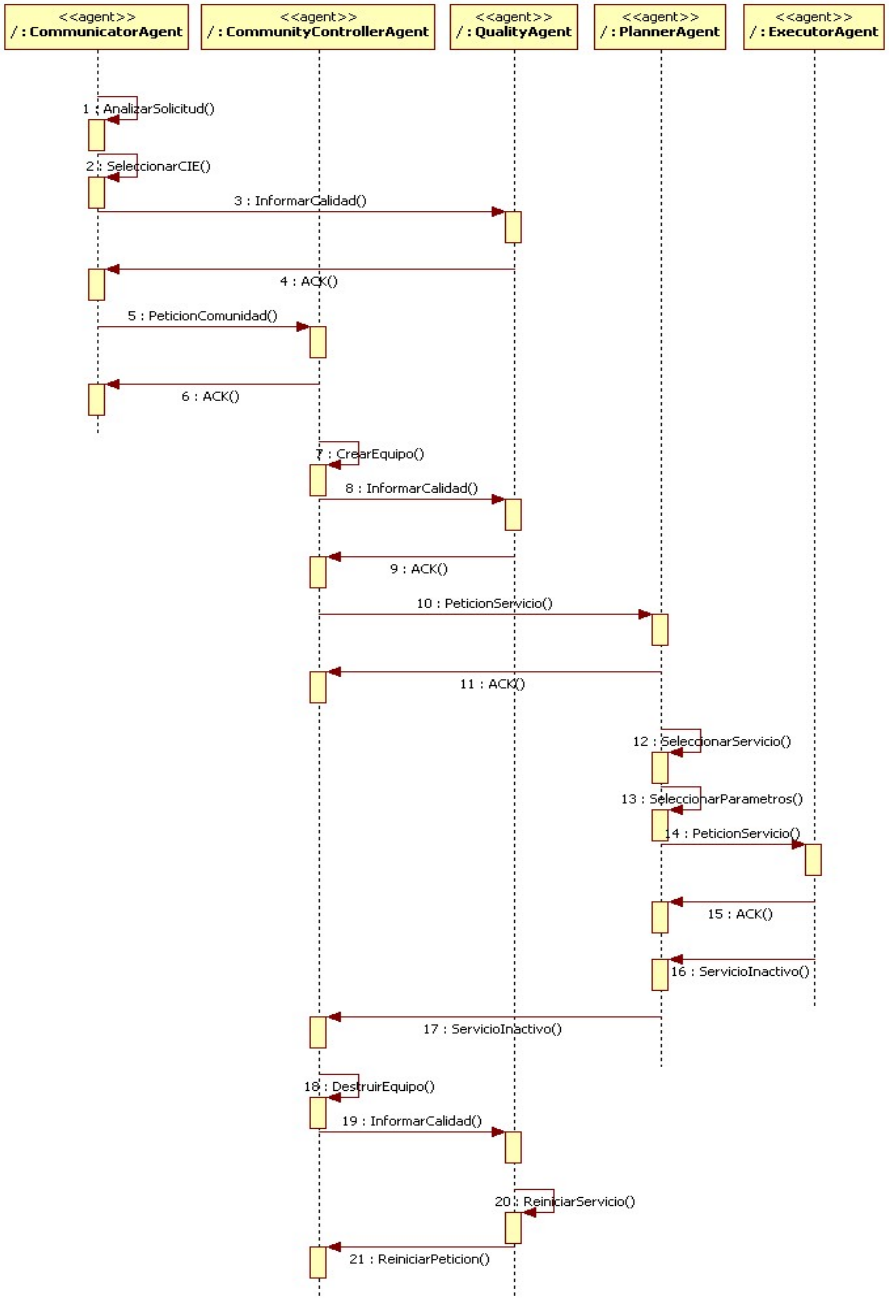


Figura B.50 Diagrama de Secuencia de Protocolos para una solicitud con error por inactividad del servicio de una Comunidad Inteligente Especializada

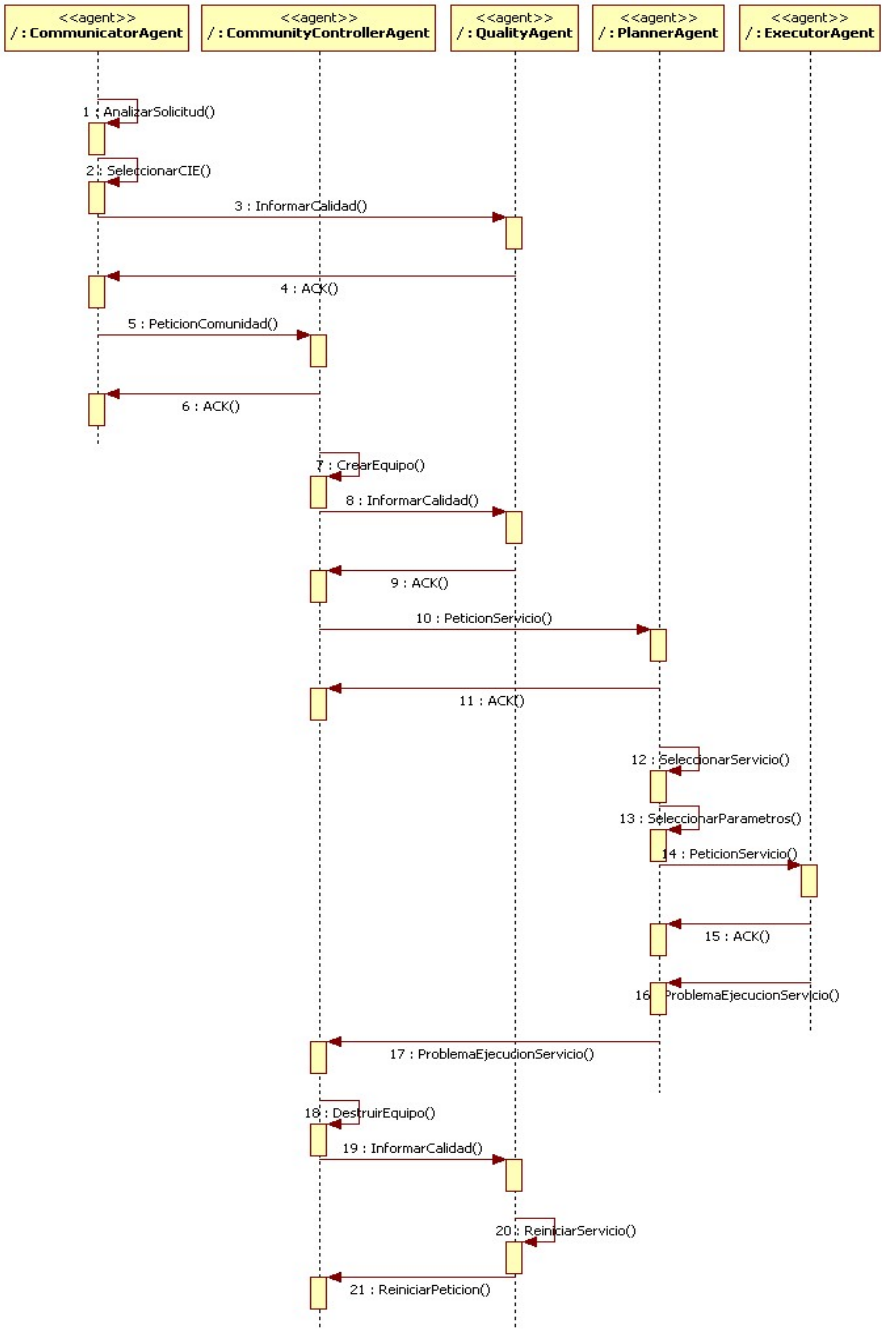


Figura B.51 Diagrama de Secuencia de Protocolos para una solicitud con error por problema de ejecución del Servicio de una Comunidad Inteligente Especializada

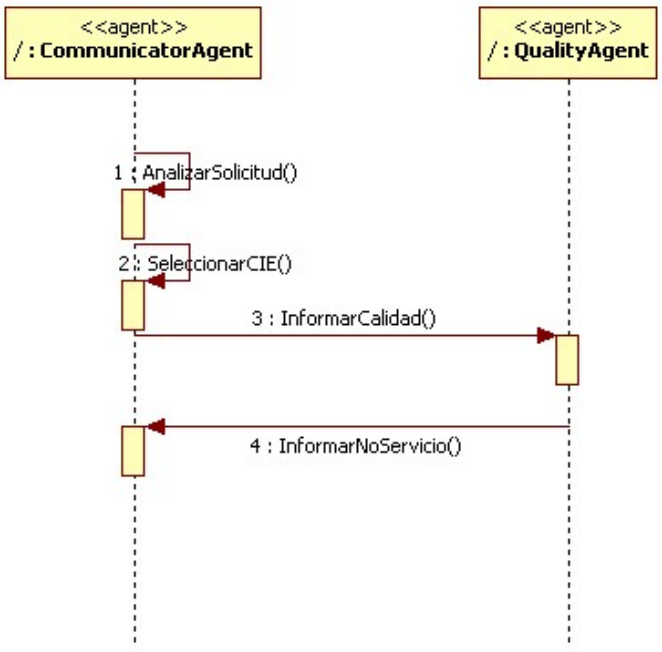


Figura B.52 Diagrama de Secuencia de Protocolos para una solicitud con error por no existir un servicio adecuado para una solicitud

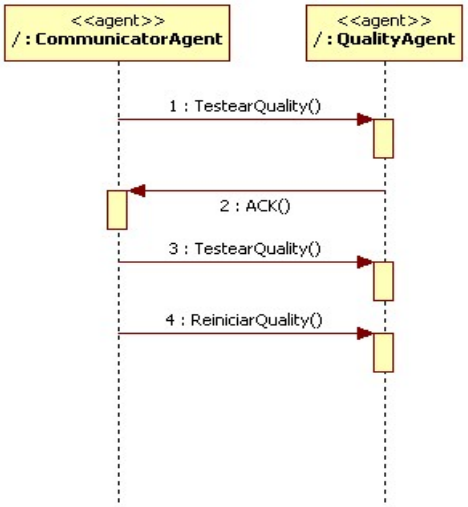


Figura B.53 Diagrama de Secuencia de Protocolos para el testeo del QualityAgent

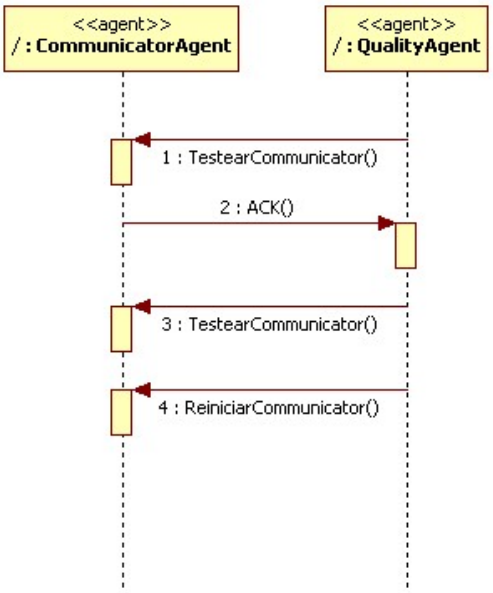


Figura B.54 Diagrama de Secuencia de Protocolos para el testeo del CommunicatorAgent

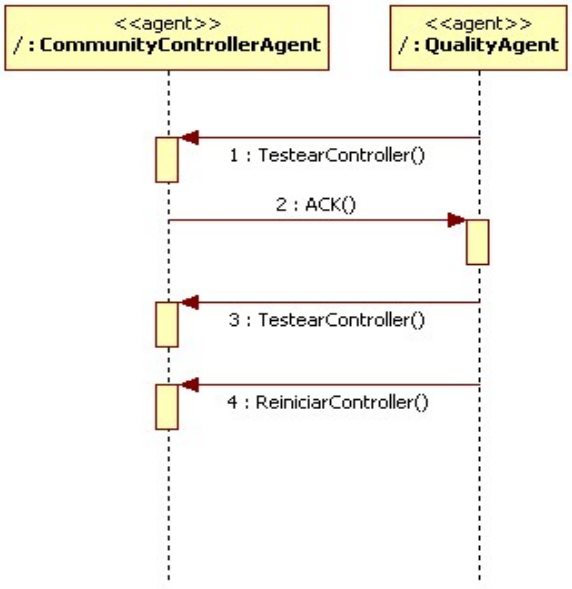


Figura B.55 Diagrama de Secuencia de Protocolos para el testeo del CommunityControllerAgent

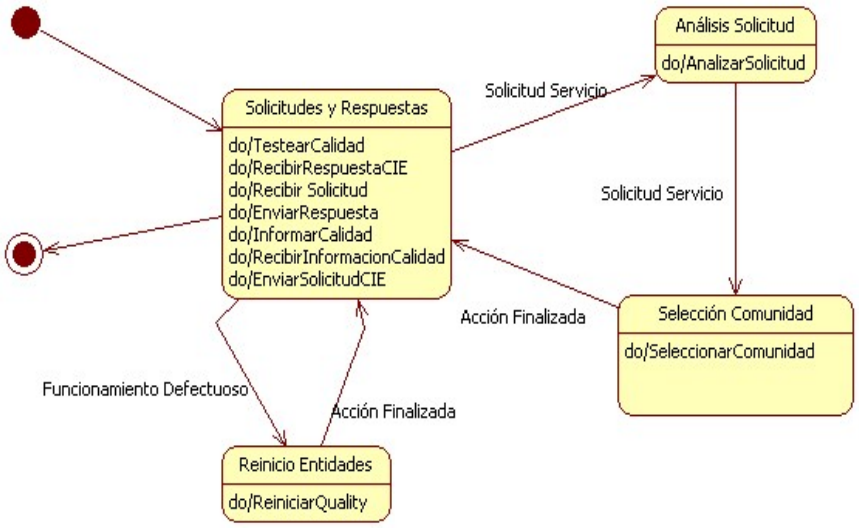


Figura B.56 Diagrama de Estados del CommunicatorAgent

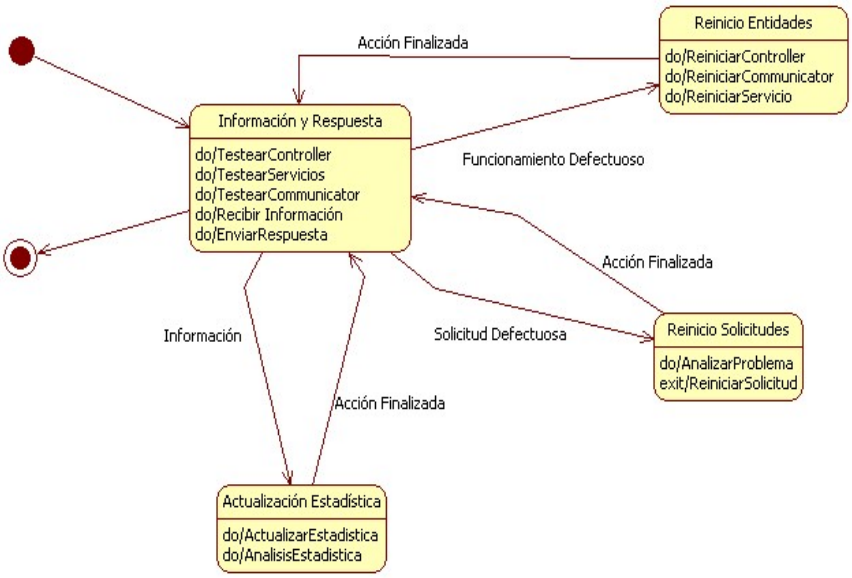


Figura B.57 Diagrama de Estados del QualityAgent

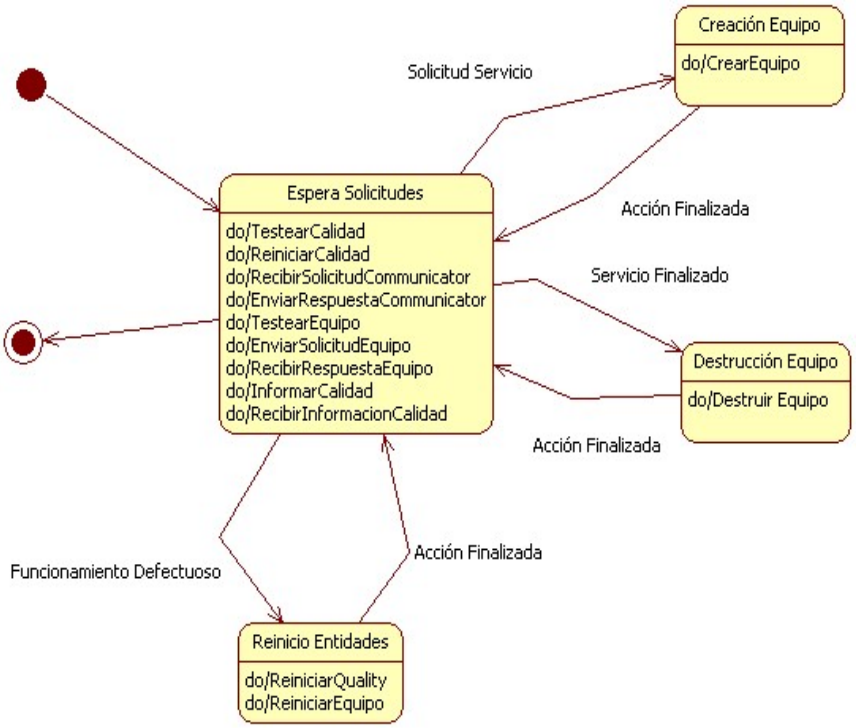


Figura B.58 Diagrama de Estados del CommunityControllerAgent

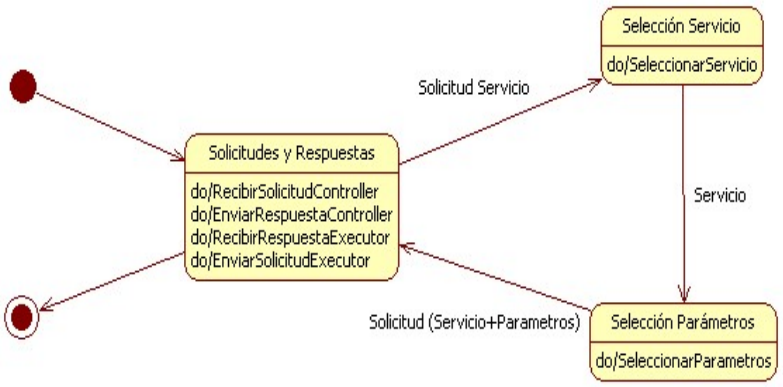


Figura B.59 Diagrama de Estados del PlannerAgent

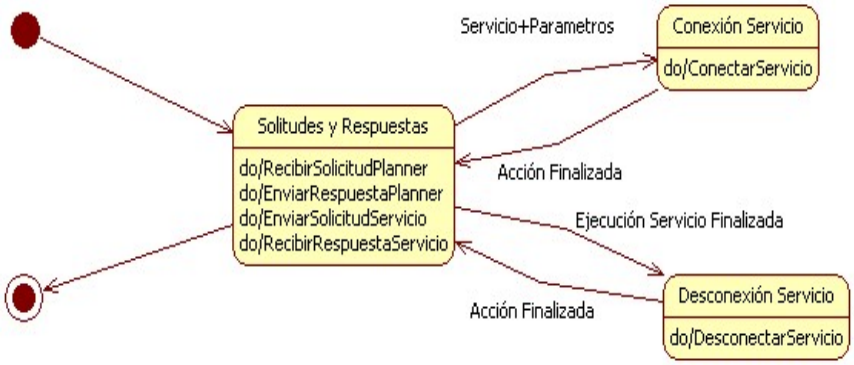


Figura B.60 Diagrama de Estados del ExecutorAgent