TESIS DOCTORAL

Arquitectura Multiagente para Entornos de Inteligencia Ambiental

VNiVERSiDAD D SALAMANCA

Departamento de Informática y Automática

Facultad de Ciencias

AUTOR
Dante Israel Tapia Martínez

DIRECTORES
Dr. D. Juan Manuel Corchado Rodríguez
Dr. D. Javier Bajo Pérez

TESIS DOCTORAL

Arquitectura Multiagente para Entornos de Inteligencia Ambiental

Universidad de Salamanca
Departamento de Informática y Automática

AUTOR
Dante Israel Tapia Martínez

DIRECTORES
Dr. D. Juan Manuel Corchado Rodríguez
Dr. D. Javier Bajo Pérez

DEPARTAMENTO DE INFORMÁTICA Y AUTOMÁTICA

FACULTAD DE CIENCIAS

TESIS DOCTORAL

ARQUITECTURA MULTIAGENTE PARA ENTORNOS DE INTELIGENCIA AMBIENTAL

AUTOR

D. Dante Israel Tapia Martínez

DIRECTORES

Dr. D. Juan Manuel Corchado Rodríguez

Dr. D. Javier Bajo Pérez

Enero de 2009

La memoria titulada “ARQUITECTURA MULTIAGENTE PARA ENTORNOS DE INTELIGENCIA AMBIENTAL” que presenta D. Dante Israel Tapia Martínez 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, Profesor Titular del Departamento de Informática y Automática de la Universidad de Salamanca, y por el profesor Dr. D. Javier Bajo Pérez, profesor Adjunto de la Escuela Universitaria de Informática de la Universidad Pontificia de Salamanca.

Salamanca, enero de 2009

Los Directores

Fdo: Dr. D. Juan M. Corchado Rodríguez
Profesor Titular de Universidad
Informática y Automática
Universidad de Salamanca

Fdo: D. Javier Bajo Pérez
Profesor Adjunto
Escuela Universitaria de Informática
Universidad Pontificia de Salamanca

El Graduando

Fdo. D. Dante I. Tapia Martínez

Resumen

La Inteligencia Ambiental (AmI: *Ambient Intelligence*) es un área multidisciplinar emergente que propone nuevas formas de interacción entre las personas y la tecnología, haciendo que esta se adapte a las necesidades de los individuos y al entorno que los rodea. La Inteligencia Ambiental propone tres conceptos fundamentales: computación ubicua, comunicación ubicua e interfaces de usuario inteligentes.

La importancia que la Inteligencia Ambiental ha adquirido en años recientes requiere el desarrollo de nuevas soluciones innovadoras. En este sentido, el desarrollo de software basado en la Inteligencia Ambiental exige crear aplicaciones cada vez más complejas y flexibles. Por tal motivo, existe una tendencia hacia la reutilización de los recursos y a compartir una misma plataforma o arquitectura. Esto se debe a que en algunos casos, los sistemas requieren funcionalidades similares implementadas como parte de otros sistemas, las cuales no siempre son compatibles entre sí.

Esta memoria presenta FUSION@ (*Flexible User and ServIces Oriented multi-ageNt Architecture*), una nueva arquitectura que integra la tecnología de agentes con la filosofía de las arquitecturas orientada a servicios (SOA), para facilitar y optimizar el desarrollo de sistemas basados en el paradigma de la Inteligencia Ambiental. FUSION@ define un conjunto de componentes que optimizan el desarrollo de sistemas distribuidos, adaptables, ubicuos, y con cierto grado de inteligencia. Esto se logra modelando las funcionalidades de los sistemas y de los propios agentes como aplicaciones y servicios independientes, los cuales pueden comunicarse de forma distribuida, independientemente de restricciones de tiempo y localización.

Si bien el conjunto de la sociedad se beneficiará de los avances tecnológicos que proporcione la Inteligencia Ambiental, serán especialmente las personas dependientes y la creciente población de la tercera edad quienes más verán mejorada su calidad de vida, principalmente la asistencia médica. Por tal motivo y para probar la validez de la arquitectura FUSION@, se ha definido un caso de estudio que consiste en el desarrollo del sistema ALZ-MAS, un sistema multiagente enfocado en mejorar los cuidados y la asistencia de ancianos y personas que presenten algún grado de dependencia, especialmente pacientes con la enfermedad de Alzheimer dentro de residencias geriátricas.

Abstract

Ambient Intelligence (AmI) is an emerging multidisciplinary area that proposes new ways of interaction between people and technology, making it suited to the needs of individuals and the environment that surrounds them. Ambient Intelligence proposes three essential concepts: ubiquitous computing, ubiquitous communication and intelligent user interfaces. The significance that Ambient Intelligence has acquired in recent years requires the development of new innovative solutions. In this sense, the development of AmI-based software requires the creation of increasingly complex and flexible applications. For that reason, there is a trend toward the reusing of resources and sharing the same platform or architecture. This is due that in some cases, the systems require similar functionalities implemented as part of other systems, which are not always compatible with each other.

This work presents a *Flexible and User Services Oriented Multi-agent Architecture* (FUSION@). This is a new architecture that integrates intelligent agents with a services oriented approach to facilitate and optimize the development of AmI-based systems. FUSION@ defines a set of components that optimize the development of distributed, adaptable, ubiquitous and intelligent systems. This is accomplished by modelling the functionalities of the systems and the agents as independent applications and services, which can communicate in a distributed way, regardless of restrictions of time and location.

While the whole of society will benefit from the technological advances provided by the Ambient Intelligence, those most likely to benefit from these advances will be the elderly and people with disabilities, whose daily lives, with particular regard to health care, will be most enhanced. A case study has been

defined to test the validity of FUSION@. The case study consists in the development of ALZ-MAS, a multiagent system aimed at improving the care and assistance for elderly and people with disabilities, especially patients with Alzheimer's disease within geriatric residences.

Agradecimientos

Deseo expresar mi más sincero agradecimiento a las personas que me han apoyado durante la realización de esta tesis.

Al Dr. Juan Manuel Corchado Rodríguez, quien me ha guiado siempre con buenas ideas y que, gracias a su incansable forma de trabajar, me ha inspirado aún más en cada etapa de esta investigación.

Al Dr. Javier Bajo Pérez, de quien he aprendido mucho y me ha ayudado de forma invaluable desde que inicie con esta investigación, sobre todo en la difusión de resultados.

A Alberto Saavedra Lorenzo, quien me ha dado la oportunidad de trasladar los conocimientos adquiridos desde el ámbito académico al empresarial.

A mis padres y hermanos: Dante, Beatriz, Ignacio y Liliana, a quienes extraño cada día estando tan lejos de casa.

Un especial agradecimiento a Blanca, por su comprensión y por haberme apoyado incondicionalmente, sobrellevando la distancia y los largos periodos en los que no hemos estado juntos.

Gracias a mis compañeros del doctorado y a los miembros del grupo de investigación BISITE, en particular a Fran y Sara, por ayudarme en diversas líneas de investigación complementarias y por los buenos momentos compartidos.

Finalmente, darle las gracias a España, un país que me ha recibido con los brazos abiertos y al que espero retribuirle mucho más de lo que me ha dado.

Esta investigación ha sido parcialmente financiada por los proyectos:

- *ALIADO (ALzheimer Intelligent Ambient DOmotic system) FIT-350300-2007-84, otorgado por el Ministerio de Industria Turismo y Comercio.*
- *GERMAP (Plataforma Inteligente para la Gestión Integral de Residencias Geriátricas) 137/2007, otorgado por el Ministerio de Trabajo y Asuntos Sociales, IMSERSO.*

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

Los constantes avances científicos y tecnológicos están propiciando que la sociedad esté inmersa en una dinámica innovadora que afecta cada vez más a la vida de las personas. Ello ha favorecido el desarrollo de interfaces y sistemas intuitivos y con cierto grado de inteligencia, capaces de reconocer y responder a las necesidades de los individuos, de una manera discreta y a menudo imperceptible (Aarts, *et al*., 2002) (Ducatel, *et al*., 2001a) (Ducatel, *et al*., 2001b). Por ello, parece necesario considerar a las personas como el centro de los desarrollos para crear entornos tecnológicamente complejos e inteligentes para ámbitos médicos, domésticos, públicos, académicos, etc. (Verhaegh, *et al*., 2006) (Schmidt, 2005) (Susperregi, *et al*., 2004).

La Inteligencia Ambiental (AmI: *Ambient Intelligence*) es un área multidisciplinar emergente (ISTAG, 2003) (Ducatel, *et al*., 2001b) (Aarts, *et al*., 2002) (Carretero, *et al*., 2005) (Tapia, *et al*., 2007a) (Tapia, *et al*., 2007c) basada en la computación ubicua (Weiser, 1993) (Weiser, 1995) y que influye en el

diseño de protocolos, las comunicaciones, la integración de sistemas, y el desarrollo de nuevos dispositivos (Mukherjee, *et al*., 2006) (Reynolds, 2006). La Inteligencia Ambiental propone nuevas formas de interacción entre las personas y la tecnología, haciendo que esta se adapte a las necesidades de los individuos y al entorno que los rodea (Aarts, *et al*., 2003a) (Carretero, *et al*., 2005). La AmI puede verse como un modelo de interacción en el que las personas están rodeadas de un entorno inteligente, consciente de su presencia, sensible al contexto y capaz de adaptarse a sus necesidades (Vázquez, *et al*., 2005). Este tipo de interacción se logra a través de tecnología embebida, no invasiva y transparente para los usuarios, con el objetivo de facilitar sus actividades diarias (Mukherjee, *et al*., 2006) (Aarts, *et al*., 2006) (Anastasopoulos, *et al*., 2005) (Haya, *et al*., 2005) (Ducatel, *et al*., 2001b) (Emiliani, *et al*., 2005).

Uno de los pilares de esta investigación es el uso de agentes y sistemas multiagente, los cuales han sido aplicados con éxito en diversos desarrollos basados en la Inteligencia Ambiental, en áreas como la educación, la cultura, el ocio y entretenimiento, la medicina, la robótica, etc. (Tapia, *et al*., 2006a) (Tapia, *et al*., 2007d) (Sadeh, *et al*., 2005) (Bajo, *et al*., 2006c) (Bajo, *et al*., 2006b) (Schön, *et al*., 2005) (Angulo, *et al*., 2004). Un agente puede describirse como un sistema computacional que se sitúa en algún entorno y es capaz de actuar de forma autónoma en dicho entorno para alcanzar sus objetivos de diseño (Wooldridge, 2002). Ampliando un poco más esta definición, tenemos que, un agente es cualquier cosa capaz de percibir su entorno a través de sensores y responder según su función en el mismo entorno a través de actuadores, asumiendo que cada agente puede percibir sus propias acciones y aprender de la experiencia para definir su comportamiento (Russell, *et al*., 1995). Por su parte, un sistema multiagente se define como cualquier sistema compuesto de múltiples agentes autónomos con capacidades incompletas para resolver un problema global, en donde no existe un sistema de control global, los datos son descentralizados y la computación es asíncrona (Wooldridge, 2002) (Jennings, *et al*., 1998) (ANA-MAS, 2005). Los agentes poseen características tales como autonomía, razonamiento,

reactividad, habilidades sociales, pro-actividad y movilidad, entre otras. Estas características hacen que resulten adecuados para ser utilizados en la construcción de sistemas basados en el paradigma de la Inteligencia Ambiental (Zaslavsky, 2004) (Grill, *et al*., 2005) (Tapia, *et al*., 2007c). Además, el rápido avance de las tecnologías para el control y la automatización hace posible integrar dispositivos que proporcionen a los agentes la capacidad para percibir el estado del entorno y reaccionar físicamente sobre éste de acuerdo a las necesidades de los usuarios (Zaslavsky, 2004).

1.1. Hipótesis y Objetivos

El objetivo final de esta investigación es el de diseñar una arquitectura que sea aplicable al desarrollo de entornos inteligentes basados en el paradigma de la Inteligencia Ambiental. La arquitectura debe proponer funcionalidades racionales capaces de ser ejecutadas en entornos dinámicos y distribuidos, y que además faciliten la interacción entre los usuarios y el entorno. Estas funcionalidades deber poder ser ejecutadas en dispositivos con reducida capacidad de almacenamiento y procesamiento. Además, la arquitectura debe proporcionar mecanismos que permitan modelar problemas de Inteligencia Ambiental en términos de agentes y sistemas multiagente, ya que son considerados como fundamentales en el desarrollo de entornos inteligentes. La tecnología de agentes juega un papel muy importante en el desarrollo de sistemas capaces de afrontar los requerimientos de la Inteligencia Ambiental, en especial la computación ubicua, la comunicación ubicua y el desarrollo de interfaces de usuario inteligentes.

De esta forma surge FUSION@ (*Flexible User and ServIces Oriented multi-ageNt Architecture*), una arquitectura que integra la tecnología de agentes y una filosofía orientada a servicios. FUSION@ define la utilización de un conjunto de tecnologías, tales como RFID, ZigBee y Wi-Fi, que permiten obtener información del contexto y actuar físicamente sobre éste de forma automática y en tiempo de ejecución. La elección de tecnologías se basa en conceptos propuestos por la Inteligencia Ambiental, como la ubicuidad, la sensibilidad al contexto, la inteligencia, la movilidad, etc. Por tal motivo, la tecnología situada en el entorno, ya sea software o hardware, es capaz de comunicarse de forma distribuida y ubicua, permitiendo a los usuarios acceder a los distintos recursos del sistema sin importar su localización física. En este sentido, los avances en la computación móvil facilitan la obtención de información del contexto y la capacidad de reacción ante los estímulos de formas más sencillas e innovadoras.

La hipótesis que se defiende en el marco de este proyecto de investigación es que la arquitectura FUSION@ facilita el desarrollo de sistemas multiagente de forma eficiente, permitiendo que estos tengan una mayor escalabilidad y capacidad para la reutilización de recursos. Además, FUSION@ permite extraer y modelar las funcionalidades de los agentes como servicios independientes, obteniendo agentes con baja carga computacional, lo cual hace que esta arquitectura sea única en su tipo. Asimismo, un enfoque distribuido dota a la arquitectura de una elevada capacidad de recuperación ante errores, así como una mayor flexibilidad para adaptar su comportamiento en tiempo de ejecución.

Para alcanzar los objetivos propuestos, ha sido necesario analizar el estado del arte de la Inteligencia Ambiental, la tecnología de agentes, modelos de razonamiento asociados con los agentes, modelos de arquitecturas orientados a servicios, metodologías para el análisis y diseño de sistemas multiagente, así como diversas tecnologías compatibles con tales objetivos. Los objetivos que han guiado esta investigación son los siguientes:

- Realizar un análisis de los problemas concretos de aplicación de la Inteligencia Ambiental en entornos de dependencia para detectar las carencias que se presentan en estos entornos.
- Buscar mecanismos formales útiles para la resolución de problemas relativos a la toma de decisiones de forma automática en entornos dinámicos en tiempo de ejecución. En este sentido, los agentes cuentan con la capacidad para generar y perseguir objetivos en relación a sus propios intereses, de forma que pueden generar soluciones de forma autónoma.
- Realizar un estudio de las tecnologías inalámbricas capaces de convivir con agentes inteligentes e integrarse en el entorno de manera ubicua para los usuarios.
- Implementar la teoría de agentes en la Inteligencia Ambiental para proponer una arquitectura multiagente distribuida aplicable a la construcción de entornos inteligentes. La arquitectura debe ser flexible y adaptable a las necesidades tanto de los usuarios como del escenario de aplicación.
- Desarrollar el prototipo de un sistema para un escenario de dependencia específico, ubicando a los usuarios en el centro del desarrollo.
    - Formalizar el modelo del sistema empleando herramientas para la ingeniería del software orientada a agentes (AOSE: *Agent-Oriented Software Engineering*), de forma que permita avanzar rápidamente hacia el proceso de implementación.
- Evaluar empíricamente los resultados obtenidos en entornos reales de aplicación.

Cabe destacar que la investigación presentada en este trabajo de tesis doctoral no se ha enfocado en crear únicamente servicios para los escenarios tecnológicos, sino que aspira a lograr espacios inteligentes ubicuos, capaces de cubrir diversos ámbitos en los que se desarrolla la vida de los usuarios. Este es el

concepto definido por el ISTAG (*Information Society Technologies Advisory Group*) como "Espacio de Inteligencia Ambiental" (ISTAG, 2003).

1.2. Motivación

La Inteligencia Ambiental ofrece un gran potencial para mejorar la calidad de vida de las personas y simplificar el uso de la tecnológica a través de una diversidad de servicios personalizados. Asimismo, proporciona a las personas nuevas formas más fáciles y eficientes de comunicación e interacción con otros individuos y sistemas (Carretero, *et al*., 2005) (Ducatel, *et al*., 2001a). Sin embargo, el adecuado desarrollo de sistemas y productos que cumplan claramente con la visión de la AmI (Ducatel, *et al*., 2001a) resulta complicado y no siempre satisfactorio. Para lograrlo, se requiere un desarrollo conjunto de modelos, técnicas y tecnología basados en servicios, donde la mejora en la interacción entre los usuarios y el entorno sea el producto resultante de una constante innovación que permita acercarse cada vez más a dicha visión (Ducatel, *et al*., 2001a).

El desarrollo de software basado en la Inteligencia Ambiental exige crear aplicaciones cada vez más complejas y flexibles. Por tal motivo, existe una tendencia hacia la reutilización de los recursos y a compartir una misma plataforma o arquitectura. En algunos casos, las aplicaciones requieren de funcionalidades similares implementadas como parte de otros sistemas, sin embargo dichas funcionalidades no siempre son compatibles entre sí. En este punto, los desarrolladores se enfrentan a esta tesitura mediante dos opciones:

1. Reutilizar funcionalidades ya implementada en otros sistemas, labor difícil de realizar debido a que éstas no fueron diseñadas para ser

reutilizadas en otros desarrollos, encontrándose implementadas sobre plataformas y/o tecnologías incompatibles entre ellas.

2. Re-implementar las funcionalidades requeridas, lo que implica más tiempo de desarrollo, aunque es en la mayoría de los casos la opción más fácil y segura.

Aunque la primera opción es la más acertada a largo plazo, la segunda opción es la más escogida en los desarrollos. Esto conlleva a tener funcionalidades replicadas, así como una mayor dificultad de migración de los sistemas internos o nucleares de las aplicaciones. Además, al no existir una estrategia de integración de aplicaciones, se generan múltiples puntos de error que pueden afectar al rendimiento de los sistemas. Este es un modelo poco flexible y escalable, con una pobre respuesta al cambio, en el cual las aplicaciones siguen siendo concebidas desde un principio como islas independientes, con el consiguiente coste económico y temporal.

Una alternativa para ayudar a resolver esta problemática, es hacer uso de arquitecturas funcionales distribuidas. Una arquitectura funcional define la estructura lógica y física de los componentes que integran un sistema, así como las interacciones entre dichos componentes (Anastasopoulos, *et al*., 2005) (González-Bedia, 2004). Las arquitecturas funcionales clásicas se caracterizan por buscar una modularidad y estructura orientada al propio sistema (ANA-MAS, 2005). Por el contrario, las arquitecturas funcionales modernas, como SOA (*Service-Oriented Architecture*), consideran la integración y el rendimiento como aspectos que deben ser tomados en cuenta cuando las funcionalidades son creadas fuera del sistema, es decir, como servicios externos ligados al sistema. Las arquitecturas distribuidas buscan una interoperabilidad entre diferentes sistemas, distribución de recursos e independencia de los lenguajes de programación (Cerami, 2002). Los servicios son integrados por medio de protocolos de comunicación que deben ser utilizados por las aplicaciones para compartir recursos en la red de servicios (Ardissono, *et al*., 2004). La compatibilidad y manejo de los mensajes son algunos de los elementos más

importantes en este tipo de desarrollos (WS-I, 2007) (W3C, 2004) (Ardissono, *et al*., 2004).

Una de las propuestas con mayor aceptación y más difundidas entre las arquitecturas funcionales modernas es la utilización de sistemas multiagente. Estos sistemas son capaces de distribuir los recursos y reducir la carga de tareas en puntos específicos, ya que la excesiva centralización de los servicios afecta de forma negativa en la funcionalidad de los sistemas, sobrecargándolos o limitando sus capacidades (Ardissono, *et al*., 2004) (Voos, 2006). Un sistema multiagente se obtiene cuando se divide un sistema complejo en varios agentes especializados para realizar una tarea específica. Los agentes deben ser capaces de cooperar entre ellos para resolver una meta común (ANA-MAS, 2005) (Wooldridge, 2002) (Jennings, *et al*., 1995), pudiéndose manejar todo el conjunto como un problema de un solo agente (Bahadori, *et al*., 2003a) (Bahadori, *et al*., 2003b). Mediante el desarrollo de agentes se reduce el esfuerzo de programar tareas múltiples, ya que sólo es necesario especificar objetivos globales para que los agentes cooperen entre ellos y así lograr los objetivos señalados. Así, el sistema es capaz de generar conocimiento y experiencia (Jayaputera, *et al*., 2007) (Pecora, *et al*., 2007). Además, una arquitectura distribuida basada en agentes, los cuales se ejecutan bajo demanda, permite mover el código a lugares donde las acciones son requeridas. De esta forma, es posible obtener respuestas en tiempo de ejecución, autonomía, continuidad de los servicios, así como mayores niveles de flexibilidad y escalabilidad que las arquitecturas centralizadas (Camarinha-Matos, *et al*., 2007) (Ardissono, *et al*., 2004) (Voos, 2006).

Aunque los agentes son capaces de afrontar los requerimientos de los desarrollos basados en la Inteligencia Ambiental, las herramientas para el desarrollo de agentes presentan unas posibilidades de comunicación limitadas y no permiten una integración sencilla con funcionalidades externas, es decir, con otros sistemas o programas desarrollados fuera de su estructura (Bellifemine, *et al*., 1999) (Martin, *et al*., 1999) (Sycara, *et al*., 2003). Por otra parte, las arquitecturas orientadas a servicios (SOA) intentan mejorar la distribución de los

recursos disponibles, facilitar la reutilización de funcionalidades y optimizar la compatibilidad entre distintas plataformas. Sin embargo, las arquitecturas orientadas a servicios no proporcionan los mecanismos suficientes de interacción ni de computación inteligente que requieren los desarrollos basados en la Inteligencia Ambiental. De esta forma, surge la necesidad de desarrollar la arquitectura FUSION@. Desarrollar una arquitectura distribuida supone definir de forma detallada cada uno de los componentes de dicha arquitectura, así como describir las interacciones que se producen entre los componentes de la arquitectura para alcanzar los objetivos finales del sistema distribuido. FUSION@ propone una solución en la cual, la tecnología de agentes y una filosofía orientada a servicios se fusionan en un intento por aprovechar las ventajas y fortalezas y reducir las principales debilidades de estos modelos. FUSION@ proporciona un marco que permite crear interfaces inteligentes y cubrir las necesidades de computación y comunicación ubicua en escenarios de Inteligencia Ambiental.

Existen diversos desarrollos que proporcionan un soporte adecuado para diseñar o desarrollar sistemas multiagente distribuidos con una filosofía SOA (Ardissono, *et al*., 2004) (Cerami, 2002) (Camarinha-Matos, *et al*., 2007) (Bonino da Silva, *et al*., 2007) (Ricci, *et al*., 2007) (Shafiq, *et al*., 2006) (Li, *et al*., 2004) (Liu, 2007) (Walton, 2006), sin embargo, no resuelven por sí solos las necesidades de los sistemas basados en la Inteligencia Ambiental, además, la mayor parte se encuentran en etapas prematuras de desarrollo y es difícil probar su potencial en escenarios reales.

Es importante señalar que la dificultad al desarrollar una arquitectura multiagente distribuida es mayor que la de una arquitectura centralizada que no utilice agentes, y al no existir herramientas de programación de alto nivel es preciso escribir grandes cantidades de código para crear servicios o clientes (Rigole, *et al*., 2002). También es necesaria una planificación más exhaustiva de los sistemas, lo cual implica un mayor tiempo para su desarrollo y posterior implementación (Wooldridge, *et al*., 2000) (Juan, *et al*., 2002a) (Bresciani, *et al*., 2004). Asimismo, el control sobre los sistemas se reduce debido a que los

agentes requieren una mayor autonomía para solucionar problemas complejos (Halkia, 2003). Por tal motivo, es necesario continuar investigando sobre desarrollos más flexibles, capaces de adaptarse a las necesidades de los usuarios de forma dinámica y en cierto modo inteligente. En este sentido, la arquitectura FUSION@ plantea la utilización de métodos y mecanismos que permitan una integración y comunicación entre agentes, servicios y aplicaciones, además de facilitar el desarrollo de sistemas multiagente distribuidos, modelando las capacidades de los agentes y del propio sistema como funcionalidades externas, es decir, como servicios y aplicaciones distribuidos.

Si bien el conjunto de la sociedad se beneficiará de los avances tecnológicos que proporcione la Inteligencia Ambiental, serán especialmente las personas dependientes y la creciente población de la tercera edad quienes más verán mejorada su calidad de vida. La medicina y los cuidados médicos juegan un rol importante, gracias a las futuras generaciones de productos y sistemas basados en dichos avances (Verhaegh, *et al*., 2006) (Angulo, *et al*., 2004) (Camarinha-Matos, *et al*., 2004) (Cesta, *et al*., 2003) (IMSERSO, 2005) (Carretero, *et al*., 2005) (Emiliani, *et al*., 2005). En este sentido y para poder valorar la arquitectura propuesta, se ha desarrollado un sistema multiagente enfocado en mejorar los cuidados y la asistencia de ancianos y personas que presenten algún grado de dependencia. Se trata de la construcción de un entorno inteligente, en el que se hace necesaria la utilización de tecnologías inalámbricas y dispositivos móviles, y en el que se pueda comprobar y medir la validez de la arquitectura.

1.3. Metodología de Investigación

El proceso de investigación seguido en este trabajo de tesis doctoral hace uso de la metodología investigación-acción (*Action-Research*). Esta metodología comienza por identificar el problema para así formular una hipótesis en la cual se basará el desarrollo. Posteriormente, se realiza una recopilación, organización y análisis de información, continuando con el diseño de una propuesta enfocada a solucionar el problema. Finalmente, se formulan las conclusiones respectivas tras evaluar los resultados obtenidos de la investigación. Para poder seguir este modelo de investigación se han definido seis actividades, las cuales se complementan para lograr los objetivos planteados. A continuación, se describen brevemente estas actividades:

1. **Definición del problema y descripción de sus características.** Esta actividad consiste en presentar la problemática, definir sus características, proponer una hipótesis para solucionar total o parcialmente dicha problemática, así como en plantear los objetivos a cumplir para lograrlo.
2. **Actualización y revisión constante e incremental del estado del arte.** Se analiza el estado del arte de las áreas, tecnologías y desarrollos relacionados con la presente investigación, para obtener un marco teórico sustentable que permita enriquecer el conocimiento y mejorar el proceso de desarrollo.
3. **Diseño y desarrollo gradual e iterativo del modelo de la propuesta.** Partiendo de 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 a través del desarrollo de sistemas prototipo.** Se formalizan las funcionalidades, componentes, comportamientos, interacciones, etc. de prototipos para ser implementados en escenarios de aplicación específico, dentro del ámbito de la problemática, experimentando con dichos prototipos para obtener diversos datos que servirán para evaluar la solución propuesta.
5. **Análisis de resultados y formulación de conclusiones.** Consiste en realizar un análisis y evaluación de los resultados obtenidos durante el desarrollo de la investigación, además de formular las conclusiones correspondientes en base a la hipótesis planteada y los objetivos definidos.
6. **Diseminación constante del conocimiento, resultados y experiencias a la comunidad científica.** Esta actividad consiste en presentar diversas publicaciones en revistas, congresos, talleres, etc., dando a conocer los avances y resultados parciales de la investigación, así como la experiencia adquirida durante el proceso de desarrollo.

1.4. Estructura de la Memoria

Para probar la hipótesis de partida y alcanzar los objetivos establecidos, se ha estructurado esta memoria en siete capítulos y diversos apéndices.

La primera parte de esta memoria corresponde al capítulo 1, en el cual se hace una introducción a este trabajo de investigación. Se describe la problemática en torno al desarrollo de sistemas basados en la Inteligencia Ambiental, así como de la necesidad de obtener una mayor flexibilidad para los modelos actuales, al reutilizar funcionalidades y distribuir los recursos. Se

presentan los objetivos, la hipótesis y la motivación que han llevado al desarrollo de la arquitectura FUSION@ (*Flexible User and ServIces Oriented multi-ageNt Architecture*). Finalmente, se detalla la metodología de investigación aplicada y se realiza una breve descripción de la estructura de esta memoria.

La segunda parte presenta el estado del arte e integra los capítulos dos y tres de esta memoria. En el capítulo 2, se analiza el estado del arte de la Inteligencia Ambiental. Se describen los aspectos más importantes y se presentan algunas de las áreas relacionadas con esta nueva área multidisciplinar. Se analiza la problemática en torno a las personas dependientes y al continuo envejecimiento de la población, describiendo algunos de los desarrollos que incorporan sistemas multiagente en entornos automatizados y de dependencia. Finalmente, se analizan las tecnologías Wi-Fi (*Wireless Fidelity*), ZigBee e identificación por radiofrecuencia (RFID), todas ellas con el potencial para aportar a FUSION@ la infraestructura tecnológica necesaria de cara a su implementación en escenarios reales. Por otro lado, en el capítulo 3, se analiza el estado del arte de la tecnología de agentes. Se describen los distintos tipos de agentes y modelos de arquitecturas de agentes, así como algunos mecanismos de razonamiento capaces de integrarse con dichos agentes. Asimismo, se estudian los trabajos y tendencias existentes en el campo de la integración de sistemas multiagente y de arquitecturas distribuidas orientadas a servicios (SOA: *Service-Oriented Architecture*). Finalmente, se analizan algunas herramientas para la ingeniería del software orientada a agentes (AOSE: *Agent-Oriented Software Engineering*).

La tercera parte de esta memoria corresponde al capítulo 4, en el cual, se presenta en detalle la arquitectura FUSION@. Se definen las principales funcionalidades que desempeña y los componentes que la integran, desde los tipos de agentes y servicios utilizados, hasta el tipo de comunicación entre ellos, así como las tecnologías base para explotar su potencial.

La cuarta parte, correspondiente al capítulo 5, presenta el caso de estudio a través del sistema ALZ-MAS, un sistema multiagente que hace uso de la

arquitectura FUSION@. ALZ-MAS se enfoca en mejorar los cuidados y la asistencia de ancianos y personas que presenten algún grado de dependencia. Se hace uso de herramientas AOSE para definir los principales componentes que integran el sistema.

La quinta parte de la memoria corresponde al capítulo 6, en donde se analizan los resultados obtenidos durante todo el proceso de investigación que han llevado al desarrollo e implementación de la arquitectura FUSION@. A partir de los resultados obtenidos se presentan las conclusiones y las contribuciones de la investigación, así como el trabajo a futuro y las posibles líneas de desarrollo a seguir basándose en esta investigación.

Finalmente, se presenta un capítulo recopilatorio redactado en inglés y se finaliza con el listado de las reseñas bibliográficas que han servido como referencia para el desarrollo de este trabajo de tesis doctoral, además de los apéndices que permiten detallar algunos aspectos importantes sobre diversas etapas del desarrollo realizado.

Capítulo

2

Inteligencia Ambiental

Las tecnologías de información y comunicación están integradas en prácticamente todos los ámbitos de nuestras vidas, facilitando las tareas diarias y mejorando la calidad de vida (Miori, *et al*., 2005) (Aarts, *et al*., 2002, 2003a). No obstante, cada vez es mayor la necesidad de utilizar técnicas y conceptos contemplados en el ámbito de la Inteligencia Ambiental (*Ambient Intelligence*).

Independientemente de las invocaciones que desde la perspectiva europea se hacen para conseguir una mayor sensibilidad social que inspire el desarrollo de la Inteligencia Ambiental, lo cierto es que su impulso principal se produce desde la tecnología (ISTAG, 2003) (Carretero, *et al*., 2005) (Mukherjee, *et al*., 2006). Por este motivo, es importante tener en cuenta el punto de vista de la demanda, articulando la concepción y puesta en funcionamiento de servicios integrados para las personas, atendiendo a las necesidades reales de la población. Dentro de estas necesidades, cada vez son más importantes los

soportes para facilitar una vida independiente y garantizar un envejecimiento saludable, así como para la integración laboral y social.

Los avances basados en la Inteligencia Ambiental tendrán profundas consecuencias en el tipo y la funcionalidad de productos y servicios emergentes, así como en la manera que las personas interactuarán con éstos, creando nuevos desafíos para el desarrollo de nuevas tecnologías de la información (Aarts, 2005) (Anastasopoulos, *et al*., 2005) (Emiliani, *et al*., 2005). Es especialmente importante la creación de interfaces que proporcionen una interacción humano-sistema lo más similar posible a la que realizan las personas entre sí (Carretero, *et al*., 2005). En este sentido, el desarrollo de agentes inteligentes será una pieza esencial para analizar datos de sensores distribuidos (Pecora, *et al*., 2007) y para que éstos sean capaces de razonar individualmente, pero trabajando de forma conjunta para analizar situaciones complejas, logrando altos niveles de interacción con los humanos (Bahadori, *et al*., 2003b) (Bahadori, *et al*., 2003a) (Pecora, *et al*., 2007).

La gran cantidad de dispositivos de los que se dispone en la actualidad, junto a su multifuncionalidad, ha generado la necesidad de interconexión (Aarts, *et al*., 2002) (Chavira, *et al*., 2007). Las redes inalámbricas aportan una infraestructura capaz de soportar las necesidades de comunicación distribuida e incrementan la movilidad, la flexibilidad y la eficiencia de los usuarios. Este tipo de redes facilitan que los servicios (programas, datos, equipos, etc.) estén disponibles para cualquier equipo de la red que lo requiera, sin importar la localización física del recurso y del usuario (Bénédet, *et al*., 2004) (Sun Microsystems, 2000). Los dispositivos para el control y la automatización permiten obtener información sobre el entorno y reaccionar físicamente sobre éste, expandiendo las capacidades de los usuarios y automatizando acciones cotidianas. Por su parte, la identificación por radio frecuencia (RFID: *Radio Frequency IDentification*) proporciona una individualización a través de un único número ID (Garfinkel, *et al*., 2005) (USDC, 2005) (Tapia, *et al*., 2007f), logrando identificar y localizar usuarios de manera automática.

Este capítulo está estructurado de la siguiente manera: Primeramente, se hace una introducción a la Inteligencia Ambiental, describiendo los elementos más importantes que componen esta área, así como sus principales aportes a la sociedad; Posteriormente, se presenta un análisis sobre el rol de la Inteligencia Ambiental en entornos de dependencia; Después, se describen algunas de las principales áreas relacionadas con la Inteligencia Ambiental, entre las que destaca domótica, un área que aporta la infraestructura tecnológica necesaria para que la AmI desarrolle gran parte de su potencial; Subsiguientemente, se presenta un breve análisis sobre la importancia de las tecnologías inalámbricas en desarrollos basados en la Inteligencia Ambiental; Finalmente, se presentan las conclusiones correspondientes a este capítulo.

2.1. Introducción a la Inteligencia Ambiental

El término "Inteligencia Ambiental" (AmI: *Ambient Intelligence*) surge en 1999 como una propuesta realizada por el *Information Society Technology Programme Advisory Group* (ISTAG) de la Comunidad Europea (ISTAG, 2003) (Haya, *et al*., 2005) (Richter, *et al*., 2004) (Aarts, *et al*., 2002). Esta propuesta se basa en los conceptos planteados por la computación ubicua, la cual tiene inferencia en áreas como la Inteligencia Artificial, la domótica, los agentes inteligentes, etc. (Carretero, *et al*., 2005) (Ducatel, *et al*., 2001b). La AmI influye en el diseño de protocolos, comunicaciones, integración de sistemas, dispositivos, etc. (Reynolds, 2006) (Aarts, *et al*., 2003b), haciendo que la tecnología se adapte a las necesidades de los usuarios, y no los usuarios a la tecnología.

La Inteligencia Ambiental se describe como un modelo de interacción (Vázquez, *et al*., 2005) en el que las personas están rodeadas de un entorno

inteligente, consciente de su presencia, sensible al contexto y capaz de adaptarse a sus necesidades (Aarts, *et al*., 2002) (Carretero, *et al*., 2005). Este tipo de interacción se logra a través de tecnología embebida, no invasiva y transparente para los usuarios (Anastasopoulos, *et al*., 2005) (Haya, *et al*., 2005), con el objetivo de facilitar sus actividades diarias (Ducatel, *et al*., 2001a) (Emiliani, *et al*., 2005).

Las principales características que debe tener un sistema basado en la AmI son (Ducatel, *et al*., 2001b) (Aarts, *et al*., 2002) (Haya, *et al*., 2005) (Vázquez, *et al*., 2005) (Schmidt, 2005) (Anastasopoulos, *et al*., 2005):

- La computación, comunicación e información deben ser ubicuas y transparentes para los usuarios, empleando dispositivos embebidos en objetos cotidianos.
- El entorno debe ser sensible al contexto, con la capacidad de percibir los estímulos externos mediante el uso de sensores.
- El entorno debe contar con cierto grado de inteligencia, aprendiendo y actualizándose automáticamente para adaptarse a las necesidades de los usuarios.
- La interacción humano-sistema debe realizarse de manera natural y no intrusiva.

La Inteligencia Ambiental puede verse como el siguiente paso en la evolución de la sociedad de información (Anastasopoulos, *et al*., 2005) (Ducatel, *et al*., 2001a), proponiendo una visión de los individuos rodeados de dispositivos e interfaces inteligentes mimetizadas en objetos de la vida cotidiana (Emiliani, *et al*., 2005) (Haya, *et al*., 2005) (Aarts, *et al*., 2003a). Para lograr esta visión, es necesario crear entornos con capacidades de computación, comunicación y procesamiento inteligentes al servicio de las personas (Vázquez, *et al*., 2005). La interacción humano-sistema debe realizarse de manera natural, simple, sin esfuerzo para los usuarios y sin implicar una curva de aprendizaje elevada (Richter, *et al*., 2004) (Ducatel, *et al*., 2001a). En un futuro próximo estaremos

aún más rodeados de tecnología y dispositivos. Por este motivo, se deben desarrollar interfaces y sistemas intuitivos e inteligentes, capaces de reconocer y responder a las necesidades de los individuos de una manera discreta y a menudo invisible (Ducatel, *et al*., 2001a) (Aarts, *et al*., 2006). En este sentido, es muy importante considerar a las personas como el centro del desarrollo al crear entornos tecnológicamente complejos en ámbitos médicos, domésticos, públicos, académicos, etc. (Schmidt, 2005) (Susperregi, *et al*., 2004).

La Inteligencia Ambiental plantea una nueva forma de interactuar entre las personas y la tecnología (Susperregi, *et al*., 2004) (Aarts, *et al*., 2002). Esta última es la que se adapta a los individuos y a su contexto, disponiendo de una gama de dispositivos interactivos capaces de afrontar las exigencias y requerimientos de los usuarios (Emiliani, *et al*., 2005) (Aarts, *et al*., 2003a). Además, la tecnología debe actuar de forma autónoma y facilitar la realización de tareas diarias. Todo ello supone profundas consecuencias en el tipo y la funcionalidad de productos y servicios emergentes, creando nuevos desafíos (Anastasopoulos, *et al*., 2005) para el desarrollo de nuevas tecnologías de la información (Emiliani, *et al*., 2005), principalmente en la creación de interfaces que proporcionen una interacción humano-sistema lo más similar posible a la que realizan las personas entre sí (Carretero, *et al*., 2005).

Los sistemas basados en la AmI mejorarán la calidad de vida de los usuarios, ofreciendo nuevos servicios y vías de comunicación más fáciles y eficientes para interactuar con otros usuarios y sistemas. Para lograr la visión propuesta por la Inteligencia Ambiental se requiere un desarrollo conjunto de modelos, técnicas y tecnología basados en servicios, donde la mejora en la interacción entre los usuarios y el entorno sea el producto resultante de una constante innovación que permita acercarse cada vez más a dicha visión (Emiliani, *et al*., 2005) (Aarts, *et al*., 2006).

Como se muestra en la **Figura 2.1**, un sistema basado en la Inteligencia Ambiental está compuesto por un conjunto de actores humanos y mecanismos

adaptables. Estos elementos colaboran de forma distribuida, son capaces de proporcionar servicios personalizados bajo demanda, y estimulan al usuario a través de su entorno, según las necesidades y características en un momento determinado.

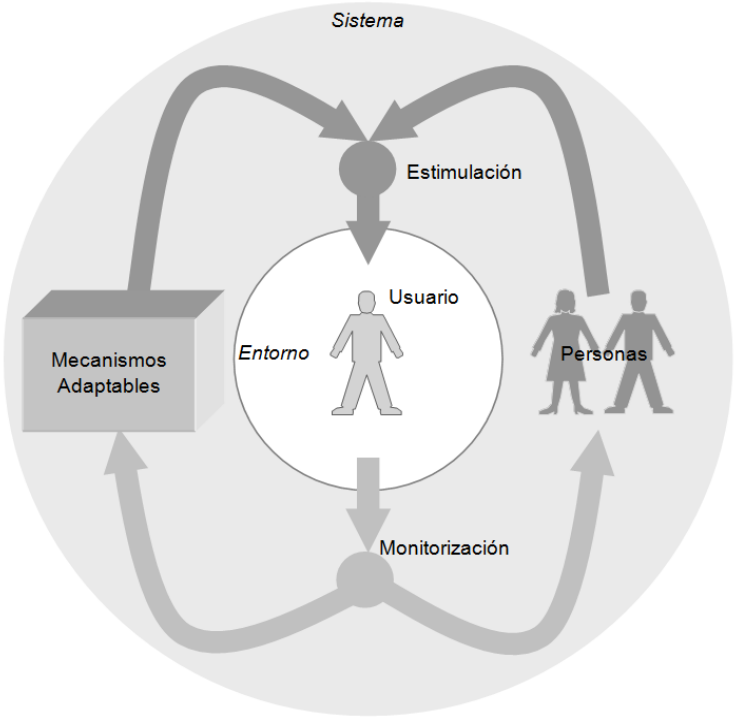

Figura 2.1. Esquema general de un sistema basado en la Inteligencia Ambiental

Un aspecto importante que se debe tomar en cuenta al desarrollar sistemas basados en la AmI, son las medidas de seguridad y privacidad, ya que estos sistemas poseen gran cantidad de información acerca de los usuarios y del entorno (Carretero, *et al.*, 2005). La continuidad de los servicios y las propiedades fundamentales de la seguridad en informática (confidencialidad, autenticación, integridad, control de acceso, no repudio, y disponibilidad) deben ser una prioridad como funcionalidad a garantizar para que estas propuestas sean aceptadas socialmente (Camarinha-Matos, *et al.*, 2004).

2.1.1. Escenarios Propuestos

Con la finalidad de visualizar el potencial de la Inteligencia Ambiental en el trabajo y en la vida diaria, el ISTAG ha propuesto cuatro posibles escenarios para el año 2010 (Ducatel, *et al*., 2001a). Cada escenario presenta distintas características económicas, sociales y culturales, pero todos ellos vistos desde una perspectiva humana. Como se aprecia en la **Figura 2.2**, los escenarios se clasifican en dos ejes, en un intento por trazar los caminos para el desarrollo de la AmI, definiendo las características y tecnologías aplicada en cada caso.

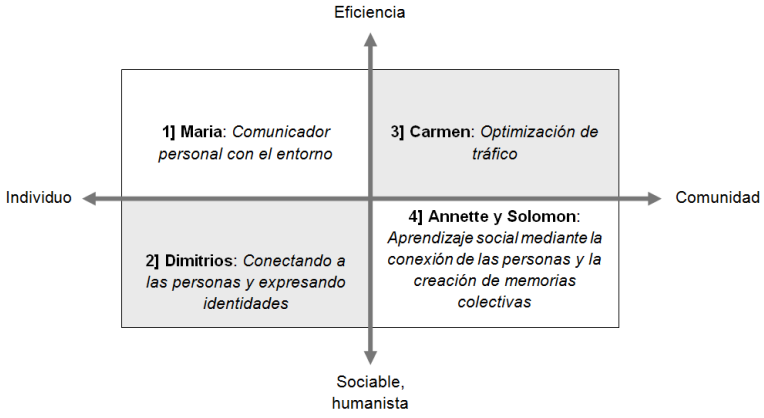

Figura 2.2. Escenarios propuestos por el ISTAG

A continuación, se describen brevemente los cuatro escenarios propuestos por el ISTAG:

1. **María.** Este escenario es una extrapolación del uso y desarrollo de dispositivos móviles, como ordenadores portátiles, teléfonos móviles y PDAs (*Personal Digital Assistant*). La Inteligencia Ambiental se centra en el soporte tecnológico que María obtiene para facilitar sus tareas. Se

describe una situación laboral típica de María, una mujer que trabaja en un entorno con alto nivel de estrés. Para llevar a cabo sus múltiples actividades, María lleva en su muñeca un pequeño dispositivo de comunicación personalizado, llamado P-Com, que le facilita la realización automática de una gran cantidad de tareas. Algunos de los servicios que María puede disfrutar son:

- *Personalización de servicios:* Su coche alquilado la identifica, calcula las rutas óptimas, reserva plaza de aparcamiento, y le permite comunicarse de forma remota con su hija. Además, su habitación de hotel le ofrece vídeo y música individualizada, y María puede controlar servicios, como la iluminación y la temperatura, mediante comandos de voz.
- *Gestión de información:* María recibe ayuda de asistentes virtuales, y además, tiene acceso seguro a bases de datos distribuidas.

2. **Dimitrios.** Este escenario se enfoca a la sociabilidad y el ocio, y muestra posibles alternativas para el uso de entornos personalizados. Dimitrio trabaja en una empresa multinacional. Durante su tiempo de descanso no desea ser molestado, por lo que gestiona el constante tráfico de llamadas y correos electrónicos que recibe, a través de un avatar de sí mismo, llamado D-Me (*Digital Me*, Yo-Digital) y que lleva embebido en su ropa. Entre los principales servicios que ofrece D-Me están:
   - *Aprendizaje automático.* Aprende de las interacciones de Dimitiros con el entorno, y utiliza ese conocimiento para tomar decisiones de forma autónoma.
   - *Mimetización.* Actúa como si fuera el propio Dimitrio, simulando su voz y comportamiento para comunicarse con las personas o incluso con otros D-Me.
   - *Integración con otros servicios.* Interactúa con dispositivos de comunicación, como terminales de videoconferencia, para que Dimitrio pueda comunicarse prácticamente en cualquier lugar.

3. **Carmen.** Este escenario, en comparación con los dos anteriores, requiere una mayor infraestructura tecnológica al desarrollarse en un entorno urbano, así como algunos cambios en la conducta cívica de las personas. El escenario describe un típico día en la vida de Carmen, una persona común que necesita ayuda para organizar sus tareas diarias. Algunas de las principales características de este escenario son:
    - *Compartir recursos.* Carmen puede acceder a un sistema para compartir vehículos con otros usuarios para desplazarse. El sistema se encarga de encontrar conductores que accedan a las peticiones y los pone en contacto. Además, los vehículos cuentan con dispositivos de identificación y tarificación automática, transfiriendo automáticamente el saldo, de la cuenta de Carmen a la del conductor, al finalizar el trayecto. Durante el viaje, el sistema informa al conductor sobre rutas alternativas y el estado del tráfico, incluso alerta sobre posibles accidentes y limita la velocidad de los automóviles cuando se registra mal tiempo.
    - *Automatización de tareas.* Su nevera está conectada a un sistema que gestiona la compra de productos que Carmen ha solicitado, informándola sobre promociones y haciéndole consultas sobre la elección de productos. Además, el sistema se encarga de que todos los productos sean entregados al centro de distribución más próximo a su domicilio.
4. **Annette y Solomon.** Es el escenario más complejo de los presentados por el ISTAG, con los mayores requerimientos tecnológicos y avances sociales. Describe la interacción que se produce entre un grupo de personas, especialmente entre Annette y Solomon, que se reúnen en un evento sobre el medio ambiente. El entorno está dotado con un sistema que es capaz de realizar las siguientes funcionalidades:

- *Personalización de servicios.* El sistema es capaz de reconocer a los asistentes, definiendo sus áreas de interés y sus conocimientos en la temática del evento.
- *Aprendizaje automático.* El sistema se conecta con otros entornos que "conocen" a los nuevos asistentes, en este caso Solomon, para así poder aprender acerca de ellos.
- *Gestión de actividades.* El sistema consulta las agendas de los participantes, les asigna una hora de llegada y la sala en donde tienen que presentarse. Además, propone nuevas reuniones para compartir las experiencias de los asistentes.
- *Gestión de recursos.* Consultando su base de conocimiento, el sistema propone a algunos asistentes, en este caso Annette, realizar una breve presentación en el evento, proporcionándoles múltiples herramientas (como proyectores, simuladores interactivos, etc.). Es posible seguir la presentación mediante videoconferencia.

2.2. La Inteligencia Ambiental y la Dependencia

Uno de los segmentos de la población que más se beneficiarán con la aparición de sistemas basados en la Inteligencia Ambiental serán los ancianos y personas con algún tipo de discapacidad (Kleindienst, *et al*., 2004) (Carretero, *et al*., 2005), contribuyendo a mejorar aspectos importantes en su calidad de vida, principalmente los cuidados médicos (Emiliani, *et al*., 2005).

En la actualidad, existe una creciente necesidad de encontrar vías más efectivas para proveer servicios de asistencia social y cuidados médicos al

número creciente de personas en estos sectores, problemática que se ha convertido en uno de los mayores retos para Europa y su comunidad científica (Nealon, *et al*., 2003) (Pecora, *et al*., 2007) (Cesta, *et al*., 2003) (Camarinha-Matos, *et al*., 2002). Algunos desarrollos como el AAL (*Ambient Assisted Living*), auspiciado por el IST (*Information Society Technologies*) dentro del Séptimo Programa Marco de la Unión Europea (7*th European Framework Programme*), se han enfocado a encontrar nuevas fórmulas para atender a esta problemática desde un punto de vista tecnológico (AAL, 2008).

La dependencia es *"la situación permanente en la que se encuentran las personas que precisan ayudas importantes de otra u otras personas para realizar actividades básicas de la vida diaria"* (IMSERSO, 2005). Estas actividades son: el cuidado personal, las actividades domésticas básicas, la movilidad esencial y el reconocimiento de personas y objetos (Costa-Font, *et al*., 2005). Las situaciones de dependencia se clasifican en tres grados (IMSERSO, 2005):

- **Grado 1: Dependencia moderada.** Personas que necesitan ayuda para realizar una o varias actividades básicas de la vida diaria, al menos una vez al día.

- **Grado 2: Dependencia severa.** Personas que necesitan ayuda para realizar varias actividades básicas de la vida diaria dos o tres veces al día, pero no requieren el apoyo permanente de un cuidador.

- **Grado 3: Gran dependencia.** Personas que necesitan ayuda para realizar distintas actividades básicas de la vida diaria varias veces al día y, por su pérdida total de autonomía mental o física, necesitan la presencia indispensable y continua de otra persona.

En Europa, durante las últimas tres décadas, el número de personas mayores de 60 años aumentó cerca de un 50%. En la actualidad, el 21% de la población de Europa es mayor de 60 años y se estima que dentro de 20 años este porcentaje aumente a un 30%, dando como resultado un total de 100 millones de

ciudadanos (WHO, 2007). Esta situación no es exclusiva de Europa, ya que estudios en otras partes del mundo muestran tendencias parecidas. Tal es el caso de Estados Unidos de América, en donde las personas mayores de 65 años son el segmento de mayor crecimiento en la población (Anderson, 1999). Se prevé que en el año 2020 aproximadamente de 1 de cada 6 ciudadanos estadounidenses se encuentre en este segmento (Camarinha-Matos, *et al*., 2002) y para el 2030 existan más de 69 millones. A estos datos hay que añadir que más del 20% de las personas mayores de 85 años tendrán una capacidad limitada para vivir independientemente, requiriendo monitorización constante y cuidados diarios (Erickson, *et al*., 1995) (Angulo, *et al*., 2004). Estimaciones realizadas por la Organización Mundial de la Salud reflejan que en el año 2025 habrá más de 1.000 millones de personas mayores de 60 años en el mundo. Si esta tendencia continúa, para el año 2050 habrá el doble, con cerca del 80% concentradas en países desarrollados (WHO, 2007). En particular, España será en el 2050 el país "más viejo" de Europa, con un 35% de habitantes mayores de 65 años, y el tercero a nivel mundial, sólo detrás de Japón y Corea (Sancho, *et al*., 2002).

La importancia de desarrollar nuevos y más rentables métodos para suministrar cuidados médicos y asistencia para los ancianos y personas dependientes se acentúa al considerar tales tendencias. Por tal motivo, es necesario desarrollar entornos seguros, discretos y adaptables para mejorar el fomento de la salud y la administración de los servicios de asistencia social (Angulo, *et al*., 2004) (Miksch, *et al*., 1997) (Corchado, *et al*., 2008). Estos desarrollos deben enfocarse especialmente a mejorar la accesibilidad, la localización, la monitorización y los dispositivos de ayuda a la movilidad (IMSERSO, 2005).

Los entornos automatizados se encuentran en el paradigma de la Inteligencia Ambiental, con áreas como la domótica (Carretero, *et al*., 2005) (Ducatel, *et al*., 2001a) (Grill, *et al*., 2005) que proporcionan la infraestructura tecnológica necesaria para lograr una adecuada interacción con el entorno. En años recientes se ha registrado un crecimiento considerable en el desarrollo de

tecnologías de automatización y arquitecturas distribuidas (Carretero, *et al*., 2005) (Richter, *et al*., 2004) (Anastasopoulos, *et al*., 2005) gracias a la aparición de espacios inteligentes y la integración de dispositivos programables a través de redes computacionales (Rigole, *et al*., 2002), impulsando el desarrollo de la computación ubicua (Kleindienst, *et al*., 2004).

Diversos autores (Nealon, *et al*., 2003) (Pecora, *et al*., 2007) (Corchado, *et al*., 2006) consideran que en un futuro próximo, las instituciones para el cuidado de la salud estarán equipadas con sistemas inteligentes capaces de interactuar con humanos. Los entornos automatizados, la domótica, los sistemas multiagente, y las arquitecturas basadas en dispositivos inteligentes, han sido recientemente explorados como sistemas de supervisión de los cuidados médicos para ancianos y personas dependientes (Angulo, *et al*., 2004). Estos sistemas podrían proveer apoyo constante en su vida diaria (Telefónica, 2003) (Cesta, *et al*., 2003), prediciendo situaciones potencialmente peligrosas y entregando un soporte cognitivo y físico para la persona asistida (Bahadori, *et al*., 2003a). Existen desarrollos que hacen uso de agentes en sistemas orientados a la automatización de servicios o para el cuidado y monitorización de personas. Algunos de ellos se describen a continuación:

- *HomeTalk* consiste en el diseño de una plataforma que provee servicios centrales en hogares, desde el punto de vista de los usuarios. Se tiene en cuenta que los servicios más demandados son los de detección (fuego, humo, fugas de agua, etc.), funciones de emergencia, apertura de puertas, y control de electrodomésticos. Esta propuesta se basa en una metodología de usuario central, integrando redes de datos, redes de control, tecnología de reconocimiento de voz, y recolección de información visual a través de cámaras de vídeo (Kleindienst, *et al*., 2004).
- *RoboCare*, apoyado por el Ministerio Italiano de Educación, se enfoca en proporcionar asistencia a ancianos a través de un entorno multiagente, utilizando software, robots, sensores y humanos (RoboCare, 2005)

(Bahadori, *et al*., 2003b). Una parte importante de este proyecto es el desarrollo de software y herramientas de visualización para sistemas con múltiples robots. El proyecto *RoboCare* proyecto apoya directamente el desarrollo de diversas líneas de investigación (RoboCare, 2005), entre ellas: sistemas multiagente, interacción humano-computadora, monitorización y diagnóstico, supervisión contra autonomía, etc.

- El proyecto *TeleCARE* está orientado a proporcionar servicios remotos a personas en edad avanzada. *TeleCARE* desarrolla una aproximación de agentes móviles mediante una plataforma genérica que permite una adición continua de servicios. Entre los principales servicios se encuentran la monitorización y control de electrodomésticos a un bajo coste (Camarinha-Matos, *et al*., 2004).
- AMICO (Susperregi, *et al*., 2004) es un sistema que hace uso de agentes para proporcionar servicios personalizados en entornos industriales, principalmente en la fabricación de materiales. Se utilizan diversas tecnologías, como el reconocimiento de voz o la realidad aumentada para mejorar la interacción de los usuarios con el entorno.
- *Smart Home Technology* (Angulo, *et al*., 2004) es una propuesta centrada en la automatización de servicios del hogar mediante el desarrollo de "unidades inteligentes de hardware" o agentes físicos, entre ellos robots. Los agentes procesan la información mediante técnicas de inteligencia computacional y microprocesadores, además de emplear una comunicación interactiva entre sus componentes, con la finalidad de lograr una mayor adaptabilidad entre los dispositivos y los usuarios.
- Rutishouer, *et al*. (2005) han desarrollado una arquitectura multiagente que facilita el control de edificios comerciales a través de sensores y actuadores. Los agentes utilizan mecanismos de aprendizaje mediante lógica difusa.

- Weyns, *et al*. (2004) han diseñado un modelo genérico para la percepción activa de sistemas multiagente, con la finalidad de percibir, interpretar y filtrar la información recopilada por sensores.
- Rigole, *et al*. (2002) han utilizado diversas herramientas de programación, con características que se adaptan a las necesidades de los sistemas multiagente. Esta propuesta se basa en la creación de agentes bajo demanda, en donde cada agente es un servicio en la red que maneja un conjunto de funcionalidades de automatización en un hogar.

A pesar de los avances en la implementación de agentes en entornos automatizados, aún hace falta mucha investigación y desarrollo en esta área, especialmente en la producción de hardware diseñado para interactuar con dichos agentes (Rigole, *et al*., 2002). Esto supone contar con una gama de dispositivos reducida, recurriendo en primera opción a componentes sin tales características. Por otra parte, las personas en general se oponen a la idea de que un ordenador o un sistema tomen decisiones por ellos. Todo esto, aunado a un mayor coste económico, la complejidad, el tiempo requerido para su desarrollo, y el número limitado de hardware, son los principales obstáculos para la aceptación social y la implementación de este tipo de propuestas. Sin embargo, con la aparición de nuevas tecnologías y desarrollos más maduros, el uso de sistemas multiagente aplicados en entornos cotidianos irá en aumento.

La Inteligencia Ambiental, a través de los conceptos de la computación ubicua, es el acercamiento tecnológico más prometedor para resolver el desafío de desarrollar estrategias que permitan la temprana detección y prevención de problemas en entornos automatizados y de dependencia (Weber, *et al*., 2005) (ISTAG, 2003) (Carretero, *et al*., 2005) (Vázquez, *et al*., 2005).

2.3. Áreas Relacionadas con la Inteligencia Ambiental

Existen diversas áreas íntimamente ligadas a la Inteligencia Ambiental. Cada una de ellas define un conjunto de aspectos que, de forma individual, no alcanzan a cubrir todos los requerimientos de la Inteligencia Ambiental, pero que en conjunto, influyen en gran medida en el tipo de interacción que se produce entre los humanos y la tecnología. A continuación, se describen estas áreas.

2.3.1. Computación Ubicua (*Ubiquitous Computing*)

La Computación Ubicua (*Ubiquitous Computing*), tal vez el área de mayor inferencia en la Inteligencia Ambiental, es un concepto introducido por Mark Weiser a principios de los años 90, cuando aseguraba que aquellos elementos de hardware y software especializados, conectados a través de cable, radio o infrarrojos, serían tan ubicuos que nadie notaría su presencia (Weiser, 1993) (Weiser, 1995).

La principal idea en que se basa la Computación Ubicua es la creación de entornos saturados de tecnología, sistemas informáticos y comunicaciones, integrados con los humanos de forma que se produzca una interacción sencilla, transparente y no intrusiva, pudiendo entenderse como una evolución de los sistemas distribuidos (Weiser, 1995) (Reynolds, 2006) (Lyytinen, *et al*., 2002). La tecnología pretende ser invisible, pero sin embargo debe integrarse con la infraestructura, soportar movilidad y ha de ser proactiva.

Uno de los principales objetivos de la Computación Ubicua es hacer "desaparecer" los dispositivos computacionales, situándolos en un segundo plano (Reynolds, 2006) (Lyytinen, *et al*., 2002). Para lograrlo, es necesario desarrollar dispositivos y sistemas que se mezclen en objetos de la vida cotidiana (ropa, zapatos, relojes, electrodomésticos, automóviles, etc.), hasta el punto de crear una extensa red de dispositivos que no sea percibida por los usuarios. Esto permite que las personas se centren en las tareas que deben hacer, no en las herramientas que utilizan. En este sentido, las herramientas deben pasar desapercibidas, tener una movilidad sencilla, y estar siempre conectadas y disponibles.

Algunas de las herramientas que tienen un gran impacto en la Computación Ubicua son (Freeman, *et al*., 2005):

- Circuitos integrados en aplicaciones específicas (ASIC: A*pplication-Specific Integrated Circuitry*)
- Reconocimiento del habla y gestos
- Sistemas en Chips (SoC: S*ystems on a Chip*)
- Interfaces perspicaces
- Materiales inteligentes
- Transistores flexibles
- Procesadores re-configurables
- Sistemas micro electromecánicos (MEMS: *Micro Electro Mechanical Systems*)

La Computación Ubicua es posible gracias al avance de la tecnología, el desarrollo de dispositivos cada vez más pequeños y potentes, así como de nuevos avances en la gestión de la información.

2.3.2. Computación Pervasiva (*Pervasive Computing*)

Es un área que generalmente se incluye dentro de la Computación Ubicua (Weiser, 1993), incluso algunos autores la nombran como tal (Satyanarayan, 2001). Sin embargo, el objetivo de la Computación Pervasiva es buscar una computación omnipresente, no tanto la movilidad, a través de servicios y aplicaciones que utilicen elementos computacionales que puedan verse y comunicarse entre sí (Lyytinen, *et al*., 2002).

Su orientación es hacia la interacción e inteligencia de los sistemas, abarcando disciplinas como la computación distribuida, computación móvil, redes de sensores, HCI (*Human Computer Interaction*) e Inteligencia Artificial (Satyanarayan, 2001).

Los dispositivos pervasivos se caracterizan por su pequeño tamaño y por estar embebidos en otros dispositivos u objetos, como teléfonos móviles, electrodomésticos, ropa, etc.

2.3.3. Computación Sensible (*Sentient Computing*)

La Computación Sensible es una disciplina de software que intenta hacer posible la Computación Ubicua mediante dispositivos computacionales integrados en el entorno y que proporcionan servicios con capacidades sensoriales que pueden "sentir" lo que pasa en el entorno (Addlesee, *et al*., 2001) (Hopper, 2000).

Promueve una computación con conciencia de su entorno, de forma que pueda cambiar su comportamiento en función de la información obtenida por

sensores pro-activos y adaptables a la situación en un momento determinado (Addlesee, *et al*., 2001) (Hopper, 2000).

Los objetivos de la Computación Sensible son:

- Construir espacios sensibles e inteligentes (*Sentient Spaces*), es decir, entornos computacionales que sienten y reaccionan.
- Mejorar la interacción entre los usuarios y el ordenador, por ejemplo, sustituyendo los tradicionales ratón y teclado.
- Materializar el concepto de Computación Ubicua.

Para lograr estos objetivos, es necesario obtener información del contexto y sistemas de reglas que modelen las reacciones de los ordenadores a los estímulos del entorno.

2.3.4. Computación Sensible al Contexto (*Context-Aware Computing*)

Actúa en función de las características del entorno que rodea al usuario en un momento determinado. Para ello, es necesario tener conciencia del contexto, por lo que su descripción debe ser modelada y almacenada. La información sobre el contexto puede ser actualizada en el momento que el sistema lo estime oportuno, una vez interpretada la información recibida.

"El "contexto" es cualquier información que puede ser utilizada para caracterizar la situación de una entidad (persona, lugar, objeto)" (Dey, *et al*., 2000). Esta información es importante para definir la interacción entre los usuarios y la tecnología que los rodea. Algunos atributos del contexto son:

- Identidad (identificación)
- Información espacial (localización, orientación, velocidad y aceleración)
- Detalles temporales (hora, fecha, etc.)

- Situación ambiental (temperatura, luz, ruido, etc.)
- Interacción social (con quién estamos y quién está cerca)
- Recursos cercanos (dispositivos y terminales accesibles)
- Capacidades de los recursos (tamaño, sonido, vídeo, etc.)
- Medidas fisiológicas (presión sanguínea, pulso cardiaco, respiración, etc.)
- Actividad (habla, lectura, caminar, correr)

La información del contexto se obtiene a partir de un conjunto de dispositivos y sensores que permiten identificar y medir diversos aspectos tanto del entorno como de los usuarios. Sin embargo, dicha información es, en algunos casos, difícil de utilizar (Dey, *et al*., 2000), ya que los datos son adquiridos por dispositivos no convencionales y obtenidos de fuentes diferentes, distribuidas y heterogéneas. Además, los datos deben ser abstraídos para dar sentido a las aplicaciones, y se requiere una detección y reacción en tiempo de ejecución debido a su naturaleza dinámica. Por estos motivos, es necesario desarrollar sensores que sean capaces de adquirir datos con diferentes características, y que además interpreten esos datos para hacerlos más comprensibles de cara a la interacción con los usuarios.

2.3.5. Creatividad Computacional (*Computational Creativity*)

Se enfoca en el desarrollo de sistemas que aporten soluciones novedosas, inesperadas y útiles para los usuarios, influyendo en sus acciones y en la interacción con el entono (Duch, 2006) (Ritchie, 2006) (O'Donoghue, *et al*., 2005).

La Creatividad Computacional se asocia con el descubrimiento de nuevos conocimientos y soluciones a través de sistemas capaces de interactuar con su entorno de forma automática, de aprender y adaptarse a dicho entorno, así como

de encontrar soluciones teniendo un conocimiento limitado del problema (Langen, *et al*., 2004).

2.3.6. Domótica

La domótica juega un papel muy importante en el paradigma de la Inteligencia Ambiental, ya que aporta la infraestructura tecnológica necesaria para lograr una adecuada interacción entre los usuarios y el entorno. Los dispositivos para el control y la automatización permiten obtener información sobre el entorno y reaccionar físicamente sobre éste, expandiendo las capacidades de los usuarios y automatizando acciones cotidianas.

Para adentrarse en el ámbito de la domótica, es necesario definir primero el significado de la palabra para después entender las implicaciones, especificaciones, beneficios, etc. que presenta. El diccionario de la Real Academia Española señala que la palabra domótica proviene del latín *domus* (casa) y del término *informática*, siendo el *"conjunto de sistemas que automatizan las diferentes instalaciones de la vivienda"* (RAE, 2008). Lapine, *et al.* (2000) definen a la domótica como *"la disciplina que estudia el desarrollo de infraestructuras inteligentes en casas y edificios, así como también las tecnologías de información para soportarlas"*. González *et al.* (2001) señalan que *"es la tecnología de automatización aplicada al manejo técnico de casas y edificios, y su principal objetivo es incrementar la calidad de vida"*, mientras que Mateos, *et al.* (2001) la describen como *"la tecnología para desarrollar e implementar la automatización de instalaciones comunes en una casa o edificio, siendo sus principales objetivos la seguridad, el ahorro de energía, el confort y las comunicaciones".* Existe un gran número de definiciones y opiniones sobre el término domótica, pero hay una parte esencial en donde la mayoría de los autores están de acuerdo: la domótica

es una tecnología que surge para mejorar el confort, la seguridad y el ahorro energético de un edificio o vivienda.

La domótica no es un producto ni un servicio, sino la integración de éstos mediante la implementación de sistemas que permitan aprovechar sus características (CEDOM, 2006), como se muestra en la **Figura 2.3**, liberando a los usuarios de tareas rutinarias y proveyendo un control óptimo de recursos (Molina, *et al*., 2004).

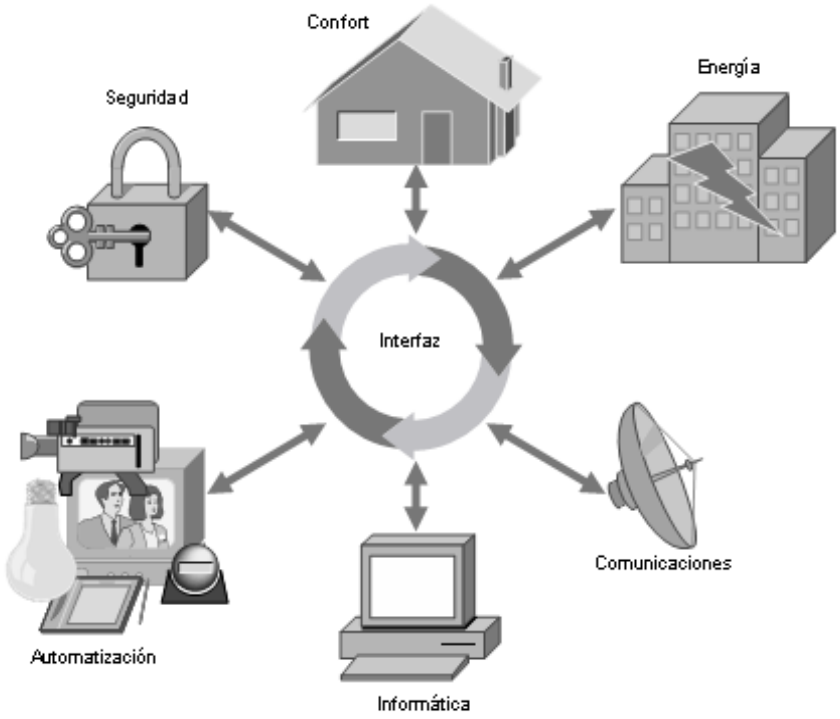

Figura 2.3. Integración de servicios en la domótica

El desarrollo de un sistema domótico implica el manejo de diferentes dispositivos, tales como sensores, alarmas, reguladores, etc., así como la elección del lugar en donde se colocarán los dispositivos (Bravo, *et al*., 2000). El diseño y desarrollo de este tipo de sistemas se realiza de forma independiente a los

dispositivos de control que sean utilizados, y se dividen en varias etapas (González, *et al*., 2001):

- Especificación funcional.
- Selección y localización de los dispositivos.
- Selección de la arquitectura del sistema.
- Configuración de la interfaz de control.
- Instalación e inicialización.

Existen dos formas de transmitir información dentro de una red domótica: cableada o inalámbrica. La elección depende de la tasa de transferencia de datos deseada, la facilidad en la instalación, y el coste de implementación (Telefónica, 2003). En viviendas ya construidas, a diferencia de las viviendas de nueva construcción, se utilizan preferentemente sistemas basados en corrientes portadoras (PLC: *PowerLine Communications*) o sistemas inalámbricos, ya que aportan un menor impacto estructural y facilidad en su instalación (Alcatel, 2005c).

Los dispositivos de control pueden clasificarse en tres categorías (Molina, *et al*., 2004):

1. **Sensores (o receptores).** Recopilan información sobre el entorno.
2. **Actuadores.** Reciben la orden de activarse o desactivarse para realizar diversas acciones.
3. **Controladores.** Procesan la información proveniente de los sensores y envían órdenes a los actuadores.

Anteriormente, sobre todo a inicios de la década de los años 90, las arquitecturas centralizadas, en donde todo el control se realizaba desde un solo dispositivo, eran la base de las instalaciones domóticas. Como consecuencia, se tenía muy poca flexibilidad, costes elevados, así como un alto grado de dificultad en su configuración e instalación. Por estos motivos, la implementación de un sistema domótico resultaba una tarea demasiado compleja, que además, no

justificaba los altos costes con los beneficios brindados (Mainardi, *et al*., 2005). Hoy en día es posible utilizar arquitecturas distribuidas en proyectos de automatización debido a la disminución de precios en la electrónica, dando como resultado la aparición de más y mejores dispositivos y sistemas capaces de compartir información con cada uno de sus componentes (Telefónica, 2003).

Aunque la domótica está bien difundida en los países industrializados, aplicándose exitosamente en escuelas, hospitales, edificios públicos, hogares, etc. (Mainardi, *et al*., 2005), la carencia de un estándar único y la escasa interoperabilidad entre las diferentes tecnologías representan el principal obstáculo para el crecimiento de este mercado (Telefónica, 2003). Sin embargo, constantemente se realizan desarrollos y aparecen nuevas tecnologías tratando de resolver esta problemática (Miori, *et al*., 2005).

2.3.6.1. Estándares de Control y Automatización

Una red de control puede considerarse, en esencia, un grupo de nodos con uno o más sensores o actuadores y cierta habilidad computacional, que se comunican sobre uno o más medios físicos utilizando un protocolo estandarizado (IEA-DSM, 1996). Este tipo de redes enlazan dispositivos de manera distribuida, reemplazando los controladores centrales y solucionando los problemas de conectividad de los sistemas centralizados, expandiendo las posibilidades de comunicación y obteniendo información automáticamente para actuar sobre el entorno más rápidamente (Bénédet, *et al*., 2004).

Los diversos tipos de sistemas de comunicación empleados en la domótica se clasifican dependiendo del canal que utilizan para la transmisión de datos (Fernández de Palencia, *et al*., 2001). En los sistemas tradicionales el emisor y el receptor están unidos físicamente, por lo que se requiere que exista una infraestructura previa en la edificación para poder implementarse. Por otra

parte, se encuentran los sistemas que operan por radio frecuencia, los cuales presentan la gran ventaja de no necesitar cableado, aunque tienen la desventaja de contar con un alcance limitado y ser susceptibles a interferencias.

Existe una gran cantidad de tecnologías, protocolos, estándares y servicios para a la comunicación e integración de dispositivos. Algunos desarrollos se centran en un enfoque software, como es el caso de Jini, UPnP (*Universal Plug and Play*), OSGI (*Open Services Gateway Initiative*), etc. (Alcatel, 2005b), mientras que otros se centran en el desarrollo de hardware para el control y automatización en entornos domóticos. Actualmente, entre las principales tecnologías para el control y automatización se encuentran X10, LonWorks, KNX y ZigBee. En el Apéndice A de esta memoria se presenta un análisis detallado de estas cuatro tecnologías.

La tecnología X10 (Apéndice A.1) es sencilla de implementar y operar (Tweed, et al., 2000). Además, el protocolo de comunicación no es propietario, permitiendo a cualquier fabricante producir dispositivos X10 con tasas muy bajas sobre los circuitos (Telefónica, 2003). La principal desventaja de X10 es contar con un único medio de físico transmisión de datos (corrientes portadoras), a menudo considerado como obsoleto, haciendo de X10 incapaz de competir con tecnologías más robustas y que presentan una mayor gama de soluciones y servicios (Tweed, et al., 2000).

LonWorks (Apéndice A.2) proporciona una gran variedad de potentes componentes hardware y software, que van desde pequeñas redes embebidas en dispositivos sencillos, hasta grandes sistemas industriales. LonWorks ha sido implementada con éxito en oficinas, hoteles e industrias, gracias a su robustez y fiabilidad (Bénédet, et al., 2004) (LonMark España, 2007). Su principal desventaja es su alto coste, motivo por el cual no ha logrado tener una gran penetración en el mercado doméstico, ya que actualmente existen alternativas más económicas que cuentan con servicios similares (Alcatel, 2005a). A

diferencia de Estados Unidos, LonWorks está poco introducido en Europa (Alcatel, 2005a).

KNX (Apéndice A.3) es una tecnología emergente que busca satisfacer cualquier demanda en sistemas de control y automatización, pretendiendo llenar el hueco que existe entre las tecnologías X10 y LonWorks, al ofrecer servicios tan robustos y diversos como esta última, pero sin olvidar el mercado doméstico (KNX, 2006). La principal desventaja es que, al ser un estándar nuevo, con el proceso de integración finalizado apenas en el año 2006, aún hace falta desarrollar más dispositivos que permitan apreciar el verdadero potencial de KNX.

ZigBee (Apéndice A.4) es un estándar de comunicación inalámbrica desarrollado por la ZigBee Alliance, enfocado a la automatización y el control remoto de aplicaciones (ZigBee, 2006) (Freescale, 2007) (Kinney, 2003) (Ergen, 2004) (Poole, 2004). ZigBee presenta una serie de características que lo hacen adecuado para su implementación en entornos de Inteligencia Ambiental. Las principales ventajas de esta tecnología son su flexibilidad, su bajo coste y su bajo consumo de energía, entre otras (Poole, 2004). ZigBee se basa en un estándar internacional, el IEEE 802.15.4, por lo que la interoperabilidad entre dispositivos de diferentes fabricantes está garantizada (ZigBee, 2006) (Ergen, 2004) (Freescale, 2007).

2.4. Tecnología Inalámbrica

Para lograr el suficiente dinamismo y distribución de recursos en los desarrollos basados en la Inteligencia Ambiental, es necesaria una adecuada infraestructura tecnológica, sobre todo en el ámbito de las comunicaciones. Los avances en

Internet y la informática distribuida han permitido el desarrollo de herramientas más eficaces para acceder a gran cantidad de información de forma remota. Actualmente existen diversas tecnologías inalámbricas como GPRS (*General Packet Radio Service*), UMTS (*Universal Mobile Telecommunications System*), Bluetooth, Wi-Fi (*Wireless Fidelity*), etc., implementadas en dispositivos cada día más comunes, como teléfonos móviles y asistentes personales. Estas tecnologías amplían las posibilidades de comunicación para los sistemas distribuidos, permitiendo una mayor flexibilidad para el desarrollo de aplicaciones sobre entornos inalámbricos.

Por otra parte, plataformas como Java ME (*Micro Edition*) de Sun Microsystems, o el sistema operativo Symbian, proporcionan el soporte necesario para el desarrollo de aplicaciones capaces de acceder a servicios remotos a través de dispositivos inalámbricos (Sun Microsystems, 2007) (Symbian, 2003). A pesar de las limitadas capacidades de procesamiento y memoria de estos dispositivos, diversos desarrollos, entre ellos LEAP (*Light Extensible Agent Platform*) (Bergenti, *et al*., 2001), basado en la plataforma JADE (*Java Agent Development Framework*) (Bellifemine, *et al*., 1999), permiten ejecutar agentes en dispositivos móviles, con el suficiente soporte para satisfacer las necesidades de comunicación de un sistema multiagente (Manmoud, 2001) (Bajo, *et al*., 2006d). El uso de agentes en dispositivos móviles ha sido implementado de forma satisfactoria en diversos escenarios (Corchado, *et al*., 2008) (Bajo, *et al*., 2006c) (Tapia, *et al*., 2007g) (Sadeh, *et al*., 2005) (Bätzold, *et al*., 2003) (Angulo, *et al*., 2004) (Schön, *et al*., 2005).

Una red es un sistema de transmisión de datos que permite compartir recursos e información (Bénédet, *et al*., 2004). En el pasado, las redes funcionaban de forma local, pero ahora Internet y las tecnologías inalámbricas permiten a las redes operar de forma global y distribuir más ampliamente los servicios (Bénédet, *et al*., 2004). Hoy en día es mucho más frecuente el uso de redes inalámbricas (Cisco Systems, 2004) (U.S. Robotics, 2003), en especial bajo la tecnología Wi-Fi, analizada en profundidad en el Apéndice B.1 de esta

memoria. Una infraestructura Wi-Fi incrementa la movilidad, la flexibilidad y la eficiencia de los usuarios, ya que permiten que todos los programas, datos y equipos estén disponibles para cualquier otro equipo de la red que lo requiera, sin importar la localización física del recurso y del usuario (Sun Microsystems, 2000) (Bénédet, *et al*., 2004). Una de las principales ventajas de las redes inalámbricas es su menor coste de implementación, presentando ahorro tanto en los componentes así como en la instalación de la infraestructura (Hewlett-Packard, 2002).

Uno de los requisitos para que la inteligencia Ambiental demuestre todo su potencial, radica en la necesidad de que el entorno reconozca la presencia de las personas, ubicándolas en un contexto, tanto geográfico, como de actividad (Susperregi, *et al*., 2004). Por tal motivo, la identificación de los usuarios es una pieza clave para una adecuada personalización de servicios e interacción con el entorno. La Identificación por radiofrecuencia (RFID: *Radio Frequency IDentification*), analizada en profundidad en el Apéndice B.2 de esta memoria, presenta una serie de características que la sitúan como una de las tecnologías con mayor crecimiento a nivel mundial. RFID ha sido empleada con éxito en cadenas de distribución, localización de artículos y personas, sistemas de pago, etc., contando con el respaldo de numerosas empresas y organismos internacionales (Tapia, *et al*., 2007f). La flexibilidad que presenta para integrarse con prácticamente cualquier producto o dispositivo, desde teléfonos móviles hasta ropa, incluso en implantes en animales y personas, es una de las principales ventajas que aporta la tecnología RFID.

En lo que respecta a los agentes y a los sistemas multiagente, estas tecnologías aumentan las capacidades propias de los agentes. En particular, Wi-Fi facilita la comunicación y coordinación que los agentes necesitan para resolver problemas de forma distribuida a través de dispositivos móviles (Bajo, *et al*., 2006d). Por su parte, ZigBee desempeña un papel fundamental en la interacción con el entorno, proporcionando a los agentes la capacidad de percibir el contexto y reaccionar físicamente sobre éste de acuerdo a las necesidades de los usuarios,

lo que se traduce en una mejor interacción entre los usuarios y el entorno (Tapia, *et al*., 2007d). Asimismo, RFID aporta una de las piezas más importantes en la Inteligencia Ambiental: la identificación de los usuarios. Esta permite a los agentes personalizar los servicios del sistema hacia los usuarios dependiendo de su perfil (Corchado, *et al*., 2008).

2.5. Conclusiones

La creciente influencia de la Inteligencia Ambiental en áreas tan diversas como la Inteligencia Artificial, los agentes y sistemas multiagente, la domótica, las telecomunicaciones, el diseño de sistemas, etc., así como los constantes avances tecnológicos, hacen necesario el desarrollo de nuevas y mejores formas de interacción entre las personas y su entorno, en donde la tecnología sea la que se adapte a las necesidades y características de los usuarios y su entorno.

En este capítulo se ha realizado una revisión del estado del arte de la Inteligencia Ambiental, analizando tanto sus objetivos como las tendencias existentes en la actualidad. Asimismo, se han analizado áreas y tecnologías capaces de impulsar nuevas formas de interacción y comunicación compatibles con la visión de la Inteligencia Ambiental.

Los diversos desarrollos realizados hasta ahora, representan un gran avance para afrontar los retos y alcanzar los objetivos que plantea la Inteligencia Ambiental. Sin embargo, es necesario contar con desarrollos más flexibles, capaces de adaptarse a las necesidades de los usuarios de forma dinámica y en cierto modo inteligente. En este sentido, una de las áreas más prometedoras para potenciar el desarrollo de la Inteligencia Ambiental es la tecnología de agentes, la cual se presenta en detalle en el capítulo 3 de esta memoria.

El éxito de la Inteligencia Ambiental como un desarrollo en la futura trayectoria tecnológica y económica, tiene que verse como una fuerza positiva para el progreso social y político de Europa (Ducatel, *et al*., 2001b).

Capítulo

3

Tecnología de Agentes

Los agentes y los sistemas multiagente cuentan con características importantes, entre las que destacan la autonomía, la reactividad, la proactividad, el aprendizaje, así como una comunicación ubicua, distribuida, y en cierto modo inteligente entre sus elementos. Haciendo uso de estas características, es posible cubrir gran parte de los requerimientos que plantean los desarrollos basados en la Inteligencia Ambiental, especialmente en el diseño de sistemas capaces de adaptarse a las necesidades de los usuarios de forma ubicua, autónoma y dinámica.

El grupo de investigación BISITE (Biomedicina, Sistemas Inteligentes y Tecnología Educativa) de la Universidad de Salamanca, cuenta con una amplia experiencia en las áreas de la Inteligencia Artificial y el desarrollo de sistemas multiagente. Por tal motivo, esta tesis doctoral sigue una línea continuista en la investigación y aplicación de estos sistemas, extendiendo sus posibilidades hacia otras áreas, en especial la Inteligencia Ambiental.

Antes de profundizar en la tecnología de agentes, es conveniente definir lo que es un "*agente*", aunque ello sea todavía tema de discusión, por lo que su definición se presenta a partir de los puntos de vista de distintos autores, indicando las principales características que un agente debe tener para que sea considerado como tal. Basándose en sus características, o en su entorno de aplicación, se han creado distintas clasificaciones para los agentes, tales como agentes de interfaz, agentes basados en metas, etc. Por otra parte, un sistema multiagente se define como cualquier sistema compuesto de múltiples agentes autónomos con capacidades incompletas para resolver un problema global, en donde no existe un sistema de control global, los datos son descentralizados y la computación es asíncrona (Wooldridge, 2002) (Jennings, *et al*., 1998) (ANA-MAS, 2005).

Las arquitecturas para la construcción de agentes especifican cómo se descomponen los agentes en un conjunto de módulos que interactúan entre sí, para lograr la funcionalidad requerida. Las arquitecturas deliberativas utilizan modelos de representación simbólica del conocimiento basados en la planificación, partiendo de un estado inicial y un conjunto de planes con un objetivo a satisfacer. Dentro de las arquitecturas deliberativas se encuentra el modelo BDI (*Belief, Desire, Intention*), en el cual, la estructura interna de los agentes y sus capacidades de elección se basan en aptitudes mentales, utilizando creencias, deseos, e intenciones (Bratman, 1987) (Georgeff, *et al*., 1998). Sin embargo, incluso estas capacidades pueden no ser suficientes para afrontar los retos que presentan los desarrollos basados en la Inteligencia Ambiental. Una alternativa consiste en modelar las características de los agentes (Wooldridge, *et al*., 1995), dotándolos de mecanismos para solucionar problemas de forma dinámica y aprender de forma autónoma. Así, los agentes son capaces de aprender partiendo de un conocimiento previo, de interactuar de forma independiente, tanto con el entorno como con los usuarios del sistema, y de adaptarse a las necesidades del contexto (Corchado, *et al*., 2001) (Corchado, *et al*., 2003) (Tapia, *et al*., 2007a).

Debido a los rápidos y constantes avances tecnológicos y el desarrollo de sistemas cada vez más complejos, entre ellos los basados en la Inteligencia Ambiental, existe una tendencia por la reutilización de funcionalidades y compatibilidad entre sistemas y plataformas. Aunque los sistemas multiagente son una alternativa para lograrlo, no siempre cubren las necesidades reales de los sistemas distribuidos. Por ello, diversos desarrollos han considerado la integración de agentes y arquitecturas funcionales modernas, como es el caso de SOA (*Service-Oriented Architecture*). Estos desarrollos intentan mejorar la distribución de los recursos disponibles, facilitar la reutilización de funcionalidades y optimizar la compatibilidad entre distintas plataformas.

Por otra parte, el desarrollo de un sistema multiagente distribuido puede resultar un proceso extenso y delicado. Durante este proceso, es conveniente utilizar herramientas para la ingeniería del software orientada a agentes (AOSE: *Agent-Oriented Software Engineering*), entre las que destacan la metodología Gaia y el lenguaje de modelado SysML. Estas herramientas facilitan y mejoran el proceso de ingeniería, consiguiendo así modelos más detallados y cercanos a la implementación de los sistemas multiagente.

La estructura de este capítulo se compone de seis secciones. En la sección 3.1 se realiza una breve introducción a los agentes y sistemas multiagente. En la sección 3.2 se definen las principales arquitecturas de agentes que existen en la actualidad. La sección 3.3 profundiza en los agentes deliberativos BDI y en sus capacidades de razonamiento. La sección 3.4 está enfocada en presentar la integración de sistemas multiagente con arquitecturas orientadas a agentes. En la sección 3.5 se describen algunas de las principales herramientas para la ingeniería del software orientada a agentes. Finalmente, la sección 3.6 presenta las conclusiones correspondientes a este capítulo.

3.1. Conceptos Generales: Agentes y Sistemas Multiagente

El concepto de agente tiene su principal origen en la inteligencia artificial, evolucionando como una entidad computacional aislada gracias a la influencia de la ingeniería de software, superando así las limitaciones de las metodologías orientadas a objetos. La principal diferencia entre los conceptos de agente y de objeto es la autonomía que poseen los agentes. Los agentes son capaces de tomar decisiones, reaccionar ante estímulos externos, cambiar su propio comportamiento y adaptarse a las necesidades del entorno.

La definición del término *"agente"* es todavía tema de discusión, ya que se asocia a un gran número de disciplinas, desde la psicología, hasta las orientadas a la computación, tales como la inteligencia artificial, la ingeniería de software y las bases de datos, entre otras, por lo que se hace difícil realizar una definición con una visión global independiente del área de influencia. Wooldridge define un agente como un sistema computacional que se sitúa en algún entorno y es capaz de actuar de forma autónoma en dicho entorno para alcanzar sus objetivos de diseño (Wooldridge, 2002). En cambio, Russell, *et al*. (1995) consideran que la noción de un agente aparece como una herramienta para analizar sistemas, no una caracterización absoluta que divida el mundo en agentes y no agentes. Para este último autor, un agente es cualquier cosa capaz de percibir su entorno a través de sensores y responder según su función en el mismo entorno a través de actuadores, asumiendo que cada agente puede percibir sus propias acciones y aprender de la experiencia para definir su comportamiento.

Debido a que existen grandes diferencias y discusión a la hora de definir lo que es un agente, se ha optado por definir una serie de características que éstos deben cumplir, las cuales se describen a continuación:

- **Autonomía.** Actuar sin la necesidad de intervenciones externas, ya sean humanos u otros agentes, y tener alguna clase de control sobre sus acciones y su estado interno.
- **Situación.** Situarse dentro de un entorno, ya sea real o virtual.
- **Reactividad.** Percibir su entorno y actuar sobre éste con la capacidad de adaptarse a sus necesidades.
- **Pro-Actividad o Racionalidad.** Tomar la iniciativa para definir metas y planes que les permitan alcanzar sus objetivos.
- **Habilidad social.** Interactuar con otros agentes, incluso con humanos.
- **Inteligencia:** Rodearse de conocimiento (creencias, deseos, intenciones y metas).
- **Organización.** Organizarse dentro de sociedades que siguen unas estructuras similares a las definidas en sociedades humanas o ecológicas.
- **Aprendizaje.** Habilidad de adaptarse progresivamente a cambios en entornos dinámicos, mediante técnicas de aprendizaje.

Existen distintos tipos de agentes, clasificados por sus características o por su entorno de aplicación (Franklin, *et al*., 1996) (Russell, *et al*., 1995) (Brenner, *et al*., 1998) (Maes, 1994). Las clasificaciones más comunes son las siguientes:

- **Interacción con el usuario**
  - *Agentes de interfaz.* Permiten la interacción con el usuario a través de comandos.
  - *Agentes autónomos.* Aunque interactúan con el usuario, el agente decide si es necesario realizar modificaciones en el entorno debido a cambios de comportamiento del usuario.
- **Movilidad**
  - *Estáticos.* Se colocan dentro de un sistema o una red, siendo incapaces de realizar tareas fuera de éstos.

  - *Móviles.* Son capaces de migrar entre plataformas o entre hosts dentro una red, eligiendo de forma autónoma el momento y el destino, para una vez realizada su tarea, regresar a su origen.

- **Modelos biológicos**
  - *Nivel de reino.* Se dividen en robóticos y computacionales; estos últimos pueden ser agentes software o agentes de vida artificial.
  - *Nivel de clase.* Son agentes software enfocados a tareas específicas u ociosas, como los virus informáticos.
- **Tipo de programa utilizado para su implementación**
  - *Reflejo simple:* Actúan basándose en reglas en donde su condición concuerde con la situación actual, la cual está definida por la percepción.
  - *Reflejo con estado interno:* Mantienen información actualizada de su entorno independientemente de sus acciones y de como éstas afectan al mismo entorno.
  - *Basados en metas:* Requieren información detallada sobre las metas para elegir las acciones que le permitan alcanzarlas.
  - *Basados en utilidad:* Permiten tomar decisiones racionales cuando para satisfacer ciertas metas se presentan conflictos o si se tienen varias metas sin la certeza de lograr alguna.
- **Software**
  - *De interfaz o asistentes personales.* Reducen el trabajo del usuario y facilitan la interacción con el sistema.
  - *De Internet.* Se utilizan para el filtrado de información.

Una vez descritos los principales requisitos que debe cumplir un agente y las características de los diferentes tipos de agentes que existen, es necesario definir lo que es un sistema multiagente (MAS: *Multi-Agent System*). Un sistema multiagente es básicamente una red de entidades enfocadas a resolver problemas, y que trabajan de manera conjunta para encontrar respuestas a los problemas que están más allá de las capacidades o del conocimiento individuales

de cada entidad (Durfee, *et al*., 1989). Una definición más general y actualizada describe un sistema multiagente como cualquier sistema compuesto de múltiples componentes autónomos que presentan las siguientes características (Jennings, *et al*., 1998):

- Cada agente tiene capacidades incompletas para resolver un problema.
- No existe un sistema de control global.
- Los datos son descentralizados.
- La computación es asíncrona.

En un sistema multiagente, los datos se encuentran organizados de forma distribuida y no existe un sistema de control global. De esta forma, cada agente se enfoca en su propia conducta, tomando la iniciativa guiado por sus objetivos y decidiendo dinámicamente las tareas que debe realizar o asignar a otros agentes. Por tal motivo, es necesario que los agentes trabajen de forma coordinada, principalmente a través de mecanismos de negociación, para alcanzar sus objetivos (Ossowski, *et al*., 1998).

3.2. Tipos de Arquitecturas de Agentes

Las arquitecturas para la construcción de agentes especifican cómo se descomponen los agentes en un conjunto de módulos que interactúan entre sí para lograr la funcionalidad requerida. Entre las principales tenemos las siguientes, diferenciadas en el modelo de razonamiento que utilizan:

- **Reactivas.** Carecen de razonamiento simbólico complejo y de conocimiento o representación de su entorno, por lo que sus mecanismos de comunicación con otros agentes son muy básicos. Los

agentes que utilizan este tipo de arquitectura reciben estímulos de su entorno y reaccionan ante ellos modificando sus comportamientos y el mismo entorno.

- **Deliberativas.** Utilizan modelos de representación simbólica del conocimiento basados en la planificación. Los agentes deliberativos emplean mecanismos de comunicación complejos y contienen un modelo simbólico del entorno. Toman decisiones utilizando razonamiento lógico basado en la concordancia de patrones y en la manipulación simbólica, partiendo de un estado inicial y un conjunto de planes con un objetivo a satisfacer.
- **Híbridas.** Son arquitecturas intermedias entre las dos anteriores. Los agentes de este tipo incluyen comportamientos reactivos y deliberativos, generando un ciclo percepción-decisión-acción. El comportamiento reactivo se utiliza para reaccionar ante eventos que no requieran decisiones complejas sobre ciertas acciones.

Dentro del paradigma de la Inteligencia Ambiental, cada tipo de agente cuenta con características distintas para cada escenario de aplicación en el que se desempeñe. Por ejemplo, en un entorno rodeado de sensores y en el cual el tiempo de reacción ante los estímulos sea lo más importante, los agentes reactivos son la opción más recomendable. Sin embargo, en ciertos escenarios puede ser necesario que los agentes sean capaces de tomar decisiones más complejas y de forma dinámica, por lo que el uso de agentes deliberativos o híbridos resulta más conveniente.

3.2.1. Arquitectura Deliberativa BDI

Dentro de las arquitecturas deliberativas, descritas anteriormente, se encuentran las intencionales, en las cuales, el modelo más utilizado hoy en día es el modelo

BDI (*Belief, Desire, Intention*). En este modelo, la estructura interna de los agentes y sus capacidades de elección se basan en aptitudes mentales, como son creencias, deseos, e intenciones (Bratman, 1987) (Georgeff, *et al*., 1998).

- Las creencias representan la parte informacional del sistema. Son la representación, con valores o expresiones, del estado del entorno y de los estados internos del agente. Aunque representan información imperfecta, son esenciales para recordar eventos pasados y para mejorar la percepción con el entorno. Las creencias resultan aún más necesarias cuando los recursos del sistema son limitados y se deben calcular acciones recurrentes continuamente. Es posible modificar la información almacenada cuando se detecten cambios en el entorno (Rao, *et al*., 1995).
- Los deseos (u objetivos) consisten en aptitudes motivacionales del sistema. Pueden ser simplemente el valor de una variable, un registro o una expresión, pero que representa algún estado final deseado. Normalmente, los agentes cuentan con más de un objetivo, así que el sistema debe tener información acerca de los objetivos y las prioridades de cada uno (Rao, *et al*., 1995).
- Las intenciones constituyen la parte premeditada o deliberada del sistema. Son un conjunto de caminos de ejecución (*threads*) que pueden interrumpirse al recibir información que involucre cambios en el entorno (Kinny, *et al*., 1991). Las intenciones permiten modificar ciertas acciones para alcanzar los objetivos. El sistema almacena las intenciones y planes actuales para utilizarlos en situaciones futuras (Rao, *et al*., 1995).

Además de las creencias, los deseos y las intenciones, es común definir un conjunto de planes, es decir, una secuencia de acciones (Bajo, *et al*., 2006c) (Corchado, *et al*., 2008). Si un agente tiene la intención de alcanzar un objetivo, debe tener la intención de ejecutar el plan que permita alcanzarlo, y además, debe creer que ese plan logrará dicho objetivo (Cavedon, *et al*., 1996). Por lo

tanto, la arquitectura básica de un agente BDI, mostrada en la **Figura 3.1**, es un conjunto de creencias, planes deseos e intenciones, procesadas por un intérprete que puede desempeñar diversas funciones, entre ellas el filtrado, el razonamiento, la deliberación, etc.

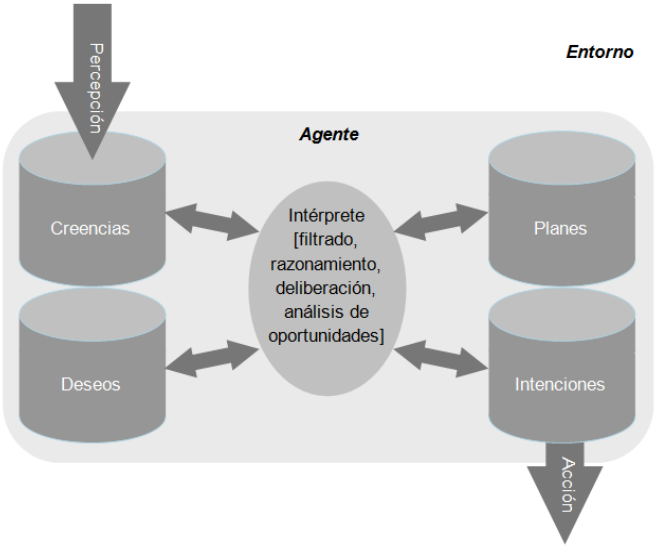

Figura 3.1. Arquitectura básica de un agente BDI

Los agentes deliberativos BDI se modelan utilizando una estructura abstracta, basada en la lógica de situaciones (o mundos posibles), denominada árbol temporal con múltiples futuros y un solo pasado (Rao, *et al*., 1991). En este árbol, cada nodo es una situación, y las ramas son 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:

- **Compatibilidad entre creencias y objetivos.** Si el agente desea alcanzar un objetivo, debe creer que dicho objetivo es cierto.
- **Compatibilidad entre objetivos e intenciones.** Antes de que el agente se enfoque en una intención, éste debe haberla formulado como deseo.
- **Las intenciones conducen a acciones.** Es necesario que el agente ejecute las intenciones a través de acciones simples.
- **Relación entre creencias e intenciones.** El agente debe creer en sus propias intenciones.
- **Relación entre creencias y objetivos.** El agente debe conocer sus objetivos y deseos.
- **No hay retrasos infinitos.** Cuando un agente adopta una intención para alcanzar un objetivo, debe continuar hasta un determinado momento finito.

Las intenciones actuales del agente influyen en sus decisiones sobre futuras intenciones. Existen varios tipos de agentes, dependiendo de cómo afecten las intenciones pasadas a las futuras (Rao, *et al*., 1991):

- **Ciego.** El agente mantiene sus intenciones hasta alcanzar su objetivo, rechazando las creencias o deseos que interfieran negativamente.
- **Firme.** El agente mantiene sus intenciones mientras cree que tiene opciones de alcanzarlas.
- **Imparcial.** El agente mantiene sus intenciones mientras éstas se corresponden con sus deseos.

Las creencias, los deseos y las intenciones se almacenan separadamente en listas y se trabaja con secuencias de eventos en una estructura de datos lineal con comportamiento FIFO (primero en entrar primero en salir). La arquitectura BDI realiza repetidamente una serie de pasos, cada uno de ellos con una duración limitada y con posibilidad de reacción ante el entorno. Al comenzar el ciclo, se lee la cola de eventos y se devuelve una lista de opciones, en donde se seleccionan aquellas que se deben adoptar, y se añaden a la cola de intenciones. A

continuación, se ejecutan todas las intenciones que impliquen la realización de una acción simple, y se comprueba si existen nuevos eventos en el entorno para incorporarlos a la cola de eventos. Por último, el agente 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 han sido exitosamente utilizados en diversos sistemas desarrollos por el grupo de investigación BISITE (Corchado, *et al*., 2003, 2005, 2008) (Bajo, *et al*., 2006c, 2006d) (Tapia, *et al*., 2007b). Estos agentes proporcionan soluciones en entornos dinámicos e inciertos. Además, son capaces de enfrentarse a problemas del mundo real, incluso cuando sólo cuenten con una visión parcial del problema y con un número limitado de recursos (Georgeff, *et al*., 1998). Desde el punto de vista de la Inteligencia Ambiental, los agentes deliberativos BDI son capaces de proporcionar altos niveles de adaptabilidad, tanto a las condiciones del entorno como de las necesidades de los usuarios. Esto resulta importante de cara a individualizar las acciones que puedan ser tomadas en cuenta para afrontar distintas situaciones de forma dinámica y autónoma.

La utilización de agentes deliberativos BDI es una pieza fundamental en el desarrollo de la arquitectura FUSION@, la cual se describe a detalle en el capítulo 4 de esta memoria.

3.2.2. Agentes BDI con Capacidades de Razonamiento

Los agentes deliberativos BDI cuentan con un conjunto de características apropiadas para el desarrollo de sistemas basados en la Inteligencia Ambiental. Sin embargo, estos desarrollos pueden requerir que los agentes cuenten con mayores grados de adaptación, aprendizaje y autonomía respecto al modelo BDI (Bratman, *et al*., 1988). Para lograrlo, es necesario modelar las características de los agentes (Wooldridge, *et al*., 1995) para dotarlos de mecanismos que permitan

solucionar problemas de forma dinámica y aprender de forma autónoma. Algunos de estos mecanismos son los sistemas de razonamiento basado en casos (CBR: *Case-Based Reasoning*) (Aamodt, *et al*., 1994) y de planificación basada en casos (CBP: *Case-Based Planning*).

Un CBR es un sistema dinámico con aprendizaje incremental que utiliza como base el razonamiento humano (Joh, 1997) (Klein, *et al*., 1988a) (Klein, *et al*., 1988b) (Ross, 1989). Los sistemas CBR realizan un proceso en el que, para resolver un problema en particular, se adaptan soluciones anteriores a problemas similares, generando conocimiento al almacenar las experiencias previas en una memoria de casos (Riesbeck, *et al*., 1989). Para resolver un problema nuevo, se consulta en la memoria de casos los problemas similares y sus soluciones, y una vez resuelto, se almacena el caso y su solución para ser utilizado en un futuro si se requiere. A los agentes deliberativos BDI que hacen uso de sistemas CBR se les llama agentes CBR-BDI (Corchado, *et al*., 2003) (Corchado, *et al*., 2005) (Bajo, *et al*., 2006d) (Carrascosa, 2004). Estos agentes son capaces de aprender partiendo de un conocimiento previo, de interactuar de forma independiente tanto con el entorno como con los usuarios del sistema, y de adaptarse a las necesidades de dicho entorno (Corchado, *et al*., 2008) (Bajo, *et al*., 2006b) (Tapia, *et al*., 2007c). De esta forma, se obtiene un mayor grado de autonomía, una mayor capacidad para la resolución de problemas, una mejor adaptación a los cambios en el contexto y además, se facilita el aprendizaje al identificar posibles soluciones basándose en experiencias pasadas (Bratman, 1987) (Bresciani, *et al*., 2002) (Carbonell, 1983) (Hammond, 1989).

Los sistemas CBP son una variación de los CBR diseñados específicamente para la elaboración de planes (Corchado, *et al*., 2008) (Cox, *et al*., 2006). Estos sistemas comienzan identificando los roles y las metas de los agentes, siguiendo la pauta de los CBR en el diseño y la implementación de la arquitectura de los agentes (Bresciani, *et al*., 2002) (Carbonell, 1983) (Hammond, 1989). En la planificación basada en casos, la solución propuesta para resolver un problema específico es un plan, por lo tanto, la solución es generada tomando en cuenta los

planes aplicados para resolver un problema similar en el pasado. Los problemas y sus planes correspondientes son almacenados en una memoria de casos. Un agente deliberativo BDI con un sistema CBP se conoce como agente CBP-BDI.

Tanto los agentes CBR-BDI como los agentes CBP-BDI han sido ampliamente estudiados por diversos investigadores asociados al grupo de investigación BISITE, demostrando su efectividad en la resolución de problemas complejos (González-Bedia, *et al*., 2002) (Corchado, *et al*., 2003, 2005, 2008) (Bajo, *et al*., 2006c, 2006d) (Tapia, *et al*., 2007b) y haciéndolos adecuados para ser incorporados en desarrollos de Inteligencia Ambiental. Por tal motivo, este tipo de agentes han sido implementados en el marco de esta investigación, tanto en la arquitectura presentada en el capítulo 4, así como en el caso de estudio descrito en el capítulo 5, ambos pertenecientes a esta memoria.

Al implementar agentes CBR-BDI es necesario definir cada una de las etapas del ciclo de vida de los sistemas CBR junto con los algoritmos que se utilizarán en cada una de ellas, además de una estructura para el concepto de caso utilizando herramientas para la construcción de agentes BDI, tales como Jadex (Pokahr, *et al*., 2003). Todo esto con el objetivo de crear una biblioteca de casos flexible, que implemente características de los agentes CBR-BDI, y que además permita a los desarrolladores adaptar los agentes para resolver problemas concretos e incorporar nuevos algoritmos a las diferentes etapas.

La estructura de un mecanismo CBR se diseña sobre el concepto de caso. Un caso es como una lección aprendida cuando se ha resuelto un determinado problema. Los casos pueden representarse de maneras distintas dependiendo del problema, la estructura, los algoritmos utilizados, etc.

En el ciclo de vida (funcionamiento) de un mecanismo CBR, se definen, se ordenan y se relacionan de acuerdo a un orden temporal los pasos de los que se aprende y se extrae información para resolver un problema específico. Como se aprecia en la **Figura 3.2**, este ciclo se inicia con la llegada de un nuevo caso, identificando el problema y examinando sus principales características, para

después procesarlo de manera secuencial por cada una de las cuatro etapas secuenciales que lo integran (Kolodner, 1993) (Aamodt, *et al*., 1994) (Watson, *et al*., 1994). Las etapas del ciclo CBR se enumeran a continuación:

1. **Recuperación.** Se extraen los casos similares almacenados en la memoria de casos, realizando una comparación de las características principales con las del nuevo caso, de manera que sean seleccionados los más parecidos a este último.
2. **Reutilización.** Se realiza un análisis de los casos extraídos de la etapa anterior para tratar de proponer una solución al problema actual, adaptando los casos extraídos si es necesario.
3. **Revisión.** Se revisa y evalúa la solución propuesta en la etapa anterior, utilizando un sistema de revisión del conocimiento, ya sea automático o bien una persona experta para comprobar si dicha solución es apropiada para el caso actual.
4. **Retención (Aprendizaje).** Consiste en memorizar el conocimiento adquirido, de forma que el caso actual junto con su solución, se almacenan en la memoria de casos.

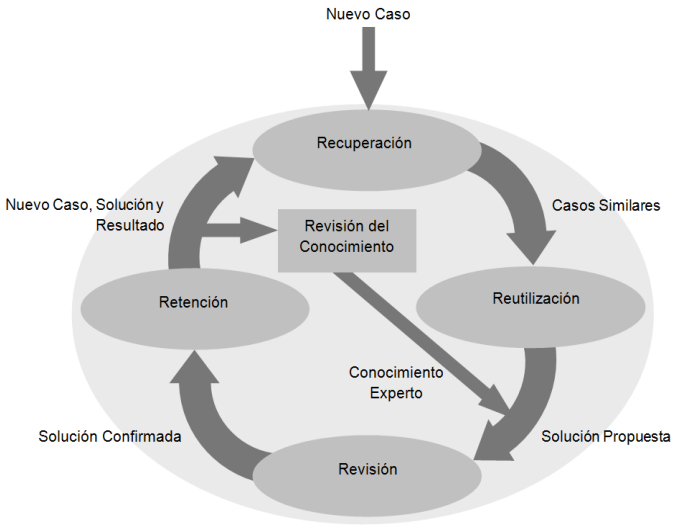

Figura 3.2. Ciclo de vida de un mecanismo CBR

3.4. Sistemas Multiagente y Arquitecturas Orientadas a Servicios

La aparición de la Inteligencia Ambiental implica cambios sustanciales en el diseño de arquitecturas funcionales, por lo que es necesario proporcionar características que permiten una computación y comunicación ubicua, así como una interacción inteligente con los usuarios. En esta sección se examinan algunas las arquitecturas existentes, analizando sus principales problemas y su capacidad para desarrollar sistemas inteligentes basándose en el paradigma de la Inteligencia Ambiental.

En secciones anteriores se presentan los agentes de forma genérica, haciendo énfasis en definir sus principales características y capacidades. Al avanzar hacia el estudio de los sistemas multiagente, resulta necesario analizar los distintos tipos de arquitecturas que éstos pueden implementar, de cara a optimizar diversos aspectos en su funcionamiento, por ejemplo, el tiempo de respuesta, la distribución de recursos, control sobre los agentes, etc.

Una arquitectura funcional define la estructura lógica y física de los componentes que integran un sistema, así como las interacciones entre dichos componentes (Rao, *et al*., 1991) (Franklin, *et al*., 1996) (W3C, 2004). Las arquitecturas funcionales clásicas se caracterizan por buscar una modularidad y estructura orientada al propio sistema (Ana Mas, 2005) (Angulo, *et al*., 2004). Por el contrario, las arquitecturas funcionales modernas, como SOA (*Service-Oriented Architecture*) o CORBA (*Common Object Request Broker Architecture*) consideran la integración y el rendimiento como aspectos que deben ser tomados en cuenta cuando las funcionalidades son creadas fuera del sistema, es decir, como programas externos ligados al sistema (Ricci, *et al*., 2007). Estas

arquitecturas buscan una interoperabilidad entre diferentes sistemas, distribución de recursos e independencia de los lenguajes de programación (Cerami, 2002) (Iverson, 2004) (WS-I, 2007). Las funcionalidades se integran por medio de protocolos de comunicación que deben ser utilizados por las aplicaciones para compartir recursos en la red (Ardissono, *et al*., 2004). La compatibilidad y manejo de los mensajes entre las funcionalidades son algunos de los elementos más importantes y complejos en este tipo de desarrollos (WS-I, 2007) (Newcomer, 2002).

En los sistemas clásicos existe una escasa flexibilidad para adaptarse de manera dinámica a las necesidades y características de los usuarios y el entorno (Camarinha-Matos, *et al*., 2007) (Ardissono, *et al*., 2004) (Voos, 2006).

Uno de los modelos de arquitecturas distribuidas más difundido es CORBA (*Common Object Request Broker Architecture*). CORBA es un estándar definido por la OMG (*Object Management Group*) que proporciona una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos (OMG, 2006). CORBA funciona como un middleware que permite la interoperabilidad entre diferentes aplicaciones escritas en diferentes lenguajes y ejecutadas en diferentes plataformas, lo que es fundamental en computación distribuida. Esto se realiza encapsulando el código escrito en otros lenguajes dentro de paquetes. Los paquetes contienen información necesaria para invocar a los objetos (servicios) y sus capacidades. De esta forma, los objetos pueden ser invocados de forma distribuida. Las capacidades (parámetros y valores de retorno) de los objetos son definidas a través de un lenguaje de definición de interfaces (IDL). CORBA está diseñado para ser independiente de lenguajes de programación y de sistemas operativos. Aunque CORBA es una potente herramienta para el desarrollo de sistemas distribuidos, no ha tenido una aceptación importante entre los desarrolladores, principalmente por problemas de compatibilidad entre especificaciones y su elevada complejidad en su implementación (Xing, et al., 1999), lo que ha llevado

a que los desarrolladores busquen alternativas más fáciles de manejar, como es el caso de SOA y *Web Services* (Cerami, 2002).

Por otra parte, SOA propone un modelo basado en una colección de servicios en el cual está presente un medio de comunicación entre ellos. La comunicación puede involucrar un simple intercambio de datos, dos o más servicios coordinando cierta actividad, etc. Para que una arquitectura orientada a servicios sea efectiva, es necesario definir de forma clara y concisa el término servicio. Así pues, un servicio puede definirse como una función bien definida, auto-contenida y que no depende del contexto o del estado de otros servicios. La comunicación entre los servicios se realiza utilizando estándares, principalmente basados en XML (*Extensible Markup Language*), con el objetivo de crear conexiones robustas y flexibles (St. Laurent, *et al*., 2001). SOA puede ser implementada a través de modelos como los Servicios Web (*Web Services*) (Cerami, 2002) (Newcomer, 2002). En este modelo, los servicios tienen una interfaz definida y conocida a la que se puede acceder a través de Internet, y al igual que una página Web está definida por un URL (*Uniform Resource Locator*), un Servicio Web está definido por un URI (*Uniform Resource Identification*) y por su interfaz, a través de la cual se puede acceder a él. Un Servicio Web puede ser considerado como una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones (Cerami, 2002) (Newcomer, 2002). Las aplicaciones pueden ser desarrolladas con distintos lenguajes de programación y ejecutadas sobre distintas plataformas. La interoperabilidad se consigue mediante la adopción de estándares abiertos (Krechmer, 1998). Las organizaciones OASIS (*Organization for the Advancement of Structured Information Standards*), W3C (*World Wide Web Consortium*) y WS-I (*Web Services Interoperability Organization*) son algunos de los comités responsables de la arquitectura y reglamentación de los Servicios Web y SOA (Cerami, 2002) (Newcomer, 2002). Para mejorar la interoperabilidad entre distintas implementaciones de Servicios Web, se ha creado el organismo WS-I (*Web Services Interoperability Organization*) (WS-I, 2007), fomentando y promoviendo la Interoperabilidad de Servicios Web sobre cualquier plataforma,

aplicaciones y lenguajes de programación. Su intención radica en la integración de estándares para impulsar el avance de los Servicios Web, de una manera estructurada y coherente (Cerami, 2002) (Newcomer, 2002). La **Figura 3.3** muestra la arquitectura básica de los Servicios Web, la cual se compone de un directorio descriptor de servicios (UDDI: *Universal Description Discovery and Integration*), aplicaciones y los propios servicios (Cerami, 2002). La descripción de los servicios y sus funcionalidades son almacenadas en el UDDI. Por su parte, las aplicaciones consultan el UDDI para elegir el servicio que necesitan. Una vez elegido el servicio, las aplicaciones se comunican directamente con éste para invocar sus funcionalidades. La comunicación entre las aplicaciones y los servicios con el UDDI se realiza mediante el lenguaje WSDL (*Web Service Definition Language*), mientras que la comunicación entre las aplicaciones y los servicios se hace a través de SOAP (Cerami, 2002) (Newcomer, 2002) (W3C, 2004).

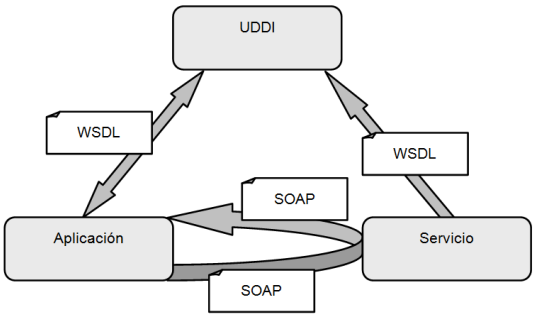

Figura 3.3. Arquitectura de los Servicios Web

Una de las propuestas más difundidas en las arquitecturas funcionales modernas es la utilización de sistemas multiagente. Estos sistemas son capaces de distribuir los recursos y reducir la carga de tareas para la unidad central, ya que la excesiva centralización de los servicios afecta de forma negativa en la funcionalidad de los sistemas, sobrecargándolos o limitando sus capacidades (Ardissono, *et al*., 2004) (Voos, 2006). Los agentes y sistemas multiagente combinan aspectos de las arquitecturas funcionales tanto clásicas como

modernas, tomando en cuenta la modularidad del sistema, la reutilización, la integración y el rendimiento. Sin embargo, la integración no siempre es posible debido principalmente a la incompatibilidad entre las distintas plataformas de agentes (Bellifemine, *et al*., 1999) y los protocolos de comunicación entre los agentes (FIPA, 2005).

Existen diversos desarrollos que intentan lograr una mayor integración e interoperabilidad entre los agentes y sistemas multiagente haciendo uso de Servicios Web y SOA (Ardissono, *et al*., 2004). Algunos de estos desarrollos se centran en la comunicación entre los agentes, mientras que otros buscan más la integración de agentes y servicios distribuidos, especialmente con Servicios Web. Bonino da Silva, *et al*. (2007) proponen la unión de un conjunto de técnicas de agentes con Servicios Web semánticos para diseñar servicios multiagente dinámicos y sensibles al contexto, a través de una capa de control con la cual los agentes descubren servicios y adaptan su comportamiento y capacidades de acuerdo a las descripciones semánticas de los servicios. Ricci, *et al*. (2007) han desarrollado un marco basado en java para crear aplicaciones compatibles con SOA y Servicios Web, los cuales son modelados como agentes. La comunicación entre los agentes y los servicios se realiza a través de lo que ellos llaman "artefactos" y el lenguaje WSDL (*Web Service Definition Language*). Shafiq, *et al*. (2006) proponen una pasarela que permite la interoperabilidad entre Servicios Web y sistemas multiagente. Esta pasarela es de hecho un agente que integra los estándares FIPA (*Foundation for Intelligent Physical Agents*) y las especificaciones del W3C (*The World Wide Web Consortium*), haciendo de traductor entre mensajes ACL, SOAP y WSDL, y combinando los directorios de los Servicios Web y las plataformas de agentes. Li, *et al*. (2004) plantean una alternativa similar, pero centrándose en la representación de los servicios utilizando mensajes SOAP y WSDL para interactuar con los agentes. Liu (2007) presenta una arquitectura multiagente para desarrollar sistemas colaborativos utilizando SOA, componentes de los Servicios Web y protocolos de comunicación para lograr dicha colaboración. Por su parte, Walton (2006) emplea una técnica

para construir sistemas multiagente utilizando Servicios Web, definiendo un lenguaje propio para representar los diálogos entre los agentes. Asimismo, existen marcos que proveen herramientas para desarrollar sistemas basados en SOA y Servicios Web, tal es el caso de Jini (Rigole, *et al*., 2002) que utiliza Java para desarrollar sistemas distribuidos y adaptables para entornos dinámicos, o WebSphere (Ben-Natan, *et al*., 2002) que provee herramientas para diversos sistemas operativos y lenguajes de programación. Sin embargo, estos marcos no son del todo abiertos y se tiene dependencia del lenguaje de programación soportado, por lo que su utilización no es demasiado extendida (Rigole, *et al*., 2002).

Una arquitectura distribuida basada en agentes, los cuales se ejecutan bajo demanda, permite mover el código a lugares donde las acciones son requeridas. Esto permite obtener respuestas en tiempo de ejecución, autonomía, continuidad de los servicios, así como mayores niveles de flexibilidad y escalabilidad que las arquitecturas centralizadas (Camarinha-Matos, *et al*., 2007) (Ardissono, *et al*., 2004) (Voos, 2006). Por otra parte, se reduce el esfuerzo de programar tareas múltiples, ya que sólo es necesario especificar objetivos globales para que los agentes cooperen entre ellos y así lograr los objetivos señalados, siendo el sistema capaz de generar conocimiento y experiencia (Jayaputera, *et al*., 2007) (Pecora, *et al*., 2007). Desafortunadamente, la dificultad al desarrollar una arquitectura multiagente distribuida es mayor que el de una arquitectura centralizada que no utilice agentes, y al no existir herramientas de programación de alto nivel es preciso escribir grandes cantidades de código para crear servicios o clientes (Rigole, *et al*., 2002). También es necesaria una planificación más exhaustiva de los sistemas, lo cual implica un mayor tiempo para su desarrollo y posterior implementación (Wooldridge, *et al*., 2000) (Juan, *et al*., 2002a) (Bresciani, *et al*., 2004). Asimismo, el control sobre los sistemas se reduce debido a que los agentes requieren una mayor autonomía para solucionar problemas complejos (Halkia, 2003).

Como se puede apreciar, los sistemas multiagente son lo suficientemente flexibles para adaptar su estructura a distintos modelos de arquitectura. La arquitectura FUSION@, presentada en detalle en capítulo 4 de esta memoria, se basa en un modelo de arquitectura multiagente distribuida orientada a servicios.

3.5. Herramientas para la Ingeniería del Software Orientada a Agentes

Al diseñar sistemas multiagente se hace necesario utilizar un proceso de ingeniería de software orientado a agentes. Existen distintas metodologías y herramientas de ingeniería orientadas a agentes (AOSE: *Agent-Oriented Software Engineering*), tales como Gaia (Wooldridge, *et al*., 2000), ROADMAP (Juan, *et al*., 2003a) (Juan, *et al*., 2003b) (Juan, *et al*., 2002b), MESSAGE (EURESCOM, 2001), ZEUS (Nwana, *et al*., 1999), MASCommonKADS (Iglesias, *et al*., 1998), AUML (Bauer, *et al*., 2000a) (Bauer, *et al*., 2000b) (Odell, *et al*., 2000) (Odell, *et al*., 2004), Prometheus (Padgham, *et al*., 2002), MaSE (DeLoach, 2001), Tropos (Bresciani, *et al*., 2002) (Bresciani, *et al*., 2004), SysML (SysML.org, 2007), etc., aunque muchas de ellas presenten importantes limitaciones y no existen herramientas de desarrollo especializadas.

La utilización de herramientas AOSE facilita enormemente el desarrollo de sistemas multiagente. En esta investigación se ha optado por hacer uso de las herramientas Gaia y SysML, debido a que su integración permite obtener modelos mucho más cercano a la implementación de los sistemas, de forma relativamente rápida y eficaz.

La metodología Gaia, presentada en detalle en el Apéndice C.1 de esta memoria, es una metodología sencilla y muy general para el análisis y diseño de sistemas multiagente. Gaia tiene como objetivo proporcionar un análisis del sistema y diseñar su estructura a partir de una colección de roles llamada organización. En Gaia no se detalla el sistema lo suficiente como realizar una implementación directa, sino que se busca reducir el nivel de abstracción hasta el punto en el que puedan ser aplicadas técnicas tradicionales. Al finalizar las fases de análisis y de diseño en Gaia se tiene un nivel de abstracción demasiado alto, pero esto puede significar una ventaja para los desarrolladores al no establecer una dependencia con una solución concreta de implementación de agentes.

Por su parte, el lenguaje de modelado SysML (*System Modeling Language*), presentado en detalle en el Apéndice C.2 de esta memoria, es un robusto lenguaje para la ingeniería de sistemas basado en UML (*Unified Modeling Language*) (SysML.org, 2007) (OMG, 2005). Aunque este lenguaje no se considera exclusivamente dentro de las herramientas AOSE, proporciona numerosas características que la hacen adecuada para el diseño y representación de sistemas multiagente complejos, incluso ha logrado desplazar a lenguajes como AUML (AUML.org, 2007). Desafortunadamente, SysML trabaja a un nivel de detalle elevado en sus etapas iniciales, aunque metodologías como Gaia pueden ser utilizadas para ampliar el proceso de ingeniería e iniciar el modelado de forma más sencilla.

3.6. Conclusiones

Actualmente, gran parte de los avances en Inteligencia Ambiental se han centrado en el desarrollo de dispositivos ubicuos, dejando a un lado la

inteligencia de los sistemas. Las características de los agentes deliberativos BDI (*Belief, Desire, Intention*), así como la posibilidad para modelar sus capacidades e integrar mecanismos de razonamiento, hacen que resulten adecuados para ser utilizados en la resolución de problemas en tiempo de ejecución en entornos altamente dinámicos. Como consecuencia, los agentes permiten a los sistemas aprender de las experiencias pasadas y reaccionar de manera diferente de acuerdo a las necesidades de los usuarios y las características del contexto en una situación determinada, requerimientos fundamentales para afrontar los retos que plantea la Inteligencia Ambiental. Por su parte, la combinación de las herramientas para la ingeniería del software Gaia y SysML, permiten obtener modelos de los sistemas multiagente cercanos a la implementación, facilitando la labor de los desarrolladores.

Sin lugar a dudas, los sistemas multiagente representan una interesante alternativa que bien vale la pena explorar para intentar afrontar los retos que presenta la Inteligencia Ambiental, especialmente en el desarrollo de sistemas dinámicos y adaptables a las necesidades de los usuarios.

Los conceptos detallados en este capítulo han permitido diseñar la arquitectura multiagente FUSION@ (*Flexible User and ServIces Oriented multi-ageNt Architecture*), descrita en detalle en el capítulo 4 de esta memoria. FUSION@ combina las ventajas de los sistemas multiagente, las arquitecturas distribuidas y el modelo SOA, satisfaciendo algunas de las necesidades identificadas en el desarrollo de sistemas basados en la Inteligencia Ambiental.

Capítulo

4

Arquitectura FUSION@

Los seres humanos nos encontramos rodeados de elementos tecnológicos que intentan mejorar nuestra calidad de vida y facilitar nuestras actividades diarias. Sin embargo, en muchas ocasiones estos elementos tecnológicos son difíciles de manejar, o bien, las personas carecen de los conocimientos necesarios para utilizarlos de forma adecuada. Ante esta situación, aparece la Inteligencia Ambiental, con el objetivo de adaptar la tecnología a las necesidades de las personas. La Inteligencia Ambiental propone tres conceptos fundamentales: computación ubicua, comunicación ubicua e interfaces de usuario inteligentes. Para conseguir satisfacer estos tres objetivos, es necesario desarrollar nuevas arquitecturas funcionales, capaces de proveer marcos adaptables y compatibles entre sí, permitiendo además acceder a las funcionalidades de los sistemas sin importar las restricciones físicas y temporales.

Una arquitectura funcional define la estructura lógica y física de los componentes que integran un sistema, así como las interacciones entre dichos componentes (Franklin, *et al*., 1996) (Anastasopoulos, *et al*., 2005) (González-Bedia, *et al*., 2002). Si nos centramos en las arquitecturas modernas, los dos principales paradigmas a tener en cuenta son los sistemas multiagente y las arquitecturas orientadas a servicios. Existen distintos desarrollos orientados a integrar agentes y sistemas multiagente con arquitecturas orientadas a servicios, algunos de ellos presentados en el capítulo 3 de esta memoria. Sin embargo, estas herramientas no resuelven por sí solas las necesidades de los desarrollos basados en la Inteligencia Ambiental, además, la mayor parte se encuentran en etapas prematuras de desarrollo y es difícil probar su potencial en escenarios reales.

FUSION@ (*Flexible User and ServIces Oriented multi-ageNt Architecture*) es una nueva arquitectura centrada en el desarrollo de sistemas basados en el paradigma de la Inteligencia Ambiental, basándose en la ubicuidad, la eficiencia y la simplicidad. FUSION@ integra la tecnología de agentes con una filosofía SOA, facilitando el desarrollo de sistemas multiagente flexibles al aprovechar las características de los agentes para modelar las funcionalidades de los sistemas y de los propios agentes como servicios distribuidos e independientes. FUSION@ también define la utilización de un conjunto de tecnologías que permiten obtener información del contexto y actuar físicamente sobre éste de forma automática y en tiempo de ejecución. La elección de tecnologías debe basarse siempre en conceptos propuestos por la Inteligencia Ambiental, como la ubicuidad, la sensibilidad al contexto, la inteligencia, la movilidad, etc. Dentro de estas tecnologías se encuentran las redes inalámbricas, las cuales aportan una infraestructura capaz de soportar las necesidades de comunicación distribuida de los agentes e incrementan la movilidad, la flexibilidad y la eficiencia de los usuarios. Las redes de control y automatización ZigBee, las redes Wi-Fi (*Wireless Fidelity*), y la identificación por radiofrecuencia (RFID: *Radio Frequency*

IDentification), componen un esquema de tecnologías inalámbricas que se integran en el entorno de manera natural para los usuarios.

Este capítulo está estructurado de la siguiente manera: Primeramente se presenta la arquitectura FUSION@, describiendo de forma general los principales elementos que la componen, entre los que destacan la plataforma de agentes, el protocolo de comunicación, y los servicios y aplicaciones. Posteriormente, cada uno de estos elementos se describe en detalle. Asimismo, se presentan algunos de los mecanismos de razonamiento diseñados para tomar decisiones de forma dinámica y en tiempo de ejecución, así como las tecnologías de referencia que permiten a los agentes obtener información del contexto y comunicarse de forma distribuida. Finalmente, se presentan las conclusiones correspondientes a este capítulo.

4.1. Descripción de la Arquitectura

Un sistema basado en la Inteligencia Ambiental está compuesto por un conjunto de actores humanos y mecanismos adaptables. Estos actores colaboran de forma distribuida y son capaces de proveer servicios personalizados bajo demanda. Asimismo, estimulan al usuario a través de su entorno, según las necesidades y características en un momento determinado. Estos hechos han propiciado una creciente necesidad de desarrollar arquitecturas que sean capaces de proveer marcos compatibles y adaptables, permitiendo un acceso a las funcionalidades sin restricciones de tiempo y localización de los recursos.

La utilización de agentes inteligentes permite cubrir gran parte de las necesidades de los entornos de Inteligencia Ambiental, especialmente la comunicación y computación ubicuas. Los agentes proporcionan a los sistemas la

capacidad de adaptarse a las necesidades y características tanto de los usuarios como del propio entorno. Además, los agentes son capaces de razonar individualmente, pero también de trabajar de forma conjunta para analizar situaciones complejas, logrando altos niveles de interacción con los humanos.

Existen diversas plataformas que definen estructuras basadas en agentes para resolver problemas de computación distribuida y facilitar la interacción con los usuarios. Estas plataformas proporcionan herramientas muy útiles para desarrollar agentes y sistemas multiagente. Una de las plataformas de agentes más utilizada dentro del grupo de investigación BISITE es JADE (Bellifemine, *et al*., 1999). Esta plataforma implementa una filosofía de "contenedores" en donde cada agente representa un contenedor (*Container Agent*) y existe un contenedor principal (*Main Container*) que encapsula a todos los contenedores. JADE hace uso del protocolo ACL (*Agent Communication Language*) de FIPA para la comunicación entre los agentes (Bellifemine, et al., 1999). Otra plataforma de agentes ampliamente difundida es OAA (*Open Agent Architecture*) (Martin, *et al*., 1999). En OAA, los facilitadores (*facilitators*) gestionan la comunicación y cooperación entre los agentes utilizando el protocolo de comunicación ICL (*Interagent Communication Language*) (Martin, et al., 1999). Los facilitadores gestionan todas las peticiones tanto de los usuarios como de los agentes. Por su parte, la plataforma RETSINA (Sycara, *et al*., 2003) implementa una red de tareas jerárquica (HTN: *Hierarchical Task Network*) para crear planes (secuencias de acciones) mediante una descomposición de tareas. RETSINA incluye un comunicador (*Communicator*) que administra la comunicación entre los agentes utilizando el protocolo KQML (*Knowledge Query and Manipulation Language*) (Sycara, et al., 2003). Sin embargo, estas plataformas presentan unas posibilidades de comunicación limitadas y no permiten una integración sencilla con funcionalidades externas, es decir, con programas ajenos a cada una de las plataformas. A su vez, las arquitecturas basadas en servicios, tales como SOA o *Web Services*, no proveen suficientes mecanismos de interacción ni de computación inteligente para cubrir las necesidades de los sistemas basados en

la Inteligencia Ambiental, los cuales deben ser dinámicos, flexibles, robustos, escalables, adaptables a los cambios del contexto y sencillos de mantener. Por tal motivo, FUSION@ combina ambos modelos en un intento por aprovechar las ventajas y fortalezas de cada uno de ellos y reducir sus principales debilidades. Aunque FUSION@ ha sido diseñada en torno al paradigma de la Inteligencia Ambiental, no está limitada a este ámbito, por lo que es posible desarrollar sistemas para distintos escenarios.

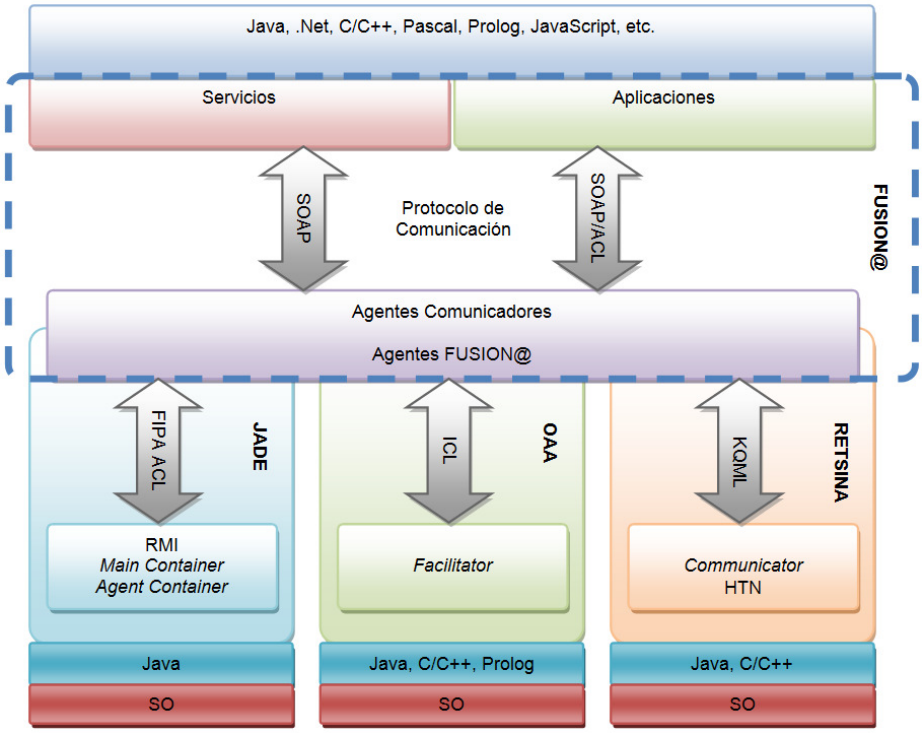

Figura 4.1. FUSION@ sobre distintas plataformas de agentes

FUSION@ es una arquitectura modular en la que los distintos agentes actúan como administradores y coordinadores de todas las funcionalidades de los sistemas y de la propia arquitectura. La arquitectura actúa como un intérprete entre los agentes, los servicios y las aplicaciones, a través de un conjunto de agentes que se ejecutan sobre una plataforma. La **Figura 4.1** muestra cómo

FUSION@ añade nuevas capas a plataformas de agentes existentes, tales como JADE, OAA o RETSINA, ampliando sus capacidades y facilitando la distribución y el manejo de recursos sobre un modelo basado en SOA. FUSION@ define cuatro bloques básicos: Aplicaciones, Servicios, Plataforma de Agentes y Protocolo de Comunicación. Estos cuatro bloques han sido definidos tomando en cuenta el modelo SOA, pero añadiendo el bloque de las aplicaciones, el cual representa una parte fundamental en la Inteligencia Ambiental: la interacción con los usuarios. Las aplicaciones y los servicios no son dependientes de un determinado lenguaje de programación, por lo que los desarrolladores tienen la opción de crearlos de acuerdo a las necesidades de cada proyecto. En las tres siguientes secciones de este capítulo se describen en detalle cada uno de estos bloques.

En FUSION@, los servicios son gestionados y personalizados por agentes basados en el modelo deliberativo BDI. La personalización se basa en las preferencias y el comportamiento de los usuarios, el conocimiento adquirido debido a interacciones previas y en la versatilidad para reaccionar ante una situación determinada. Algunos de estos servicios son: identificación, localización, automatización de tareas, planificación de actividades, monitorización, estimulación del entorno, etc. Estos servicios proporcionan a los usuarios y al propio sistema un mejor conocimiento del contexto en el que se encuentran. Otra característica a destacar es que, gracias a las capacidades de los agentes, los sistemas desarrollados pueden hacer uso de mecanismos de razonamiento o técnicas de aprendizaje para gestionar las funcionalidades de acuerdo a las particularidades del contexto, el cual puede cambiar dinámicamente en el tiempo.

En FUSION@, las principales funcionalidades no se encuentran embebidas en la estructura de los agentes, sino que se modelan como servicios independientes. Los agentes hacen uso de estos servicios para aprender de experiencias pasadas y adaptar su comportamiento de acuerdo a las necesidades y características del contexto. Los agentes, las aplicaciones y los servicios son capaces de comunicarse de forma distribuida, incluso desde dispositivos móviles.

Esto hace posible la utilización de recursos sin importar su ubicación. Cuando un sistema hace uso de FUSION@, los usuarios son capaces de acceder a todas las funcionalidades desde aplicaciones distribuidas que pueden ser ejecutadas desde diversos tipos de dispositivos e interfaces, tales como ordenadores portátiles, teléfonos móviles, PDAs, etc. Todas las peticiones y respuestas son gestionadas por un conjunto de agentes dentro de la plataforma. Los agentes analizan las peticiones enviadas por las aplicaciones para invocar, de forma local o remota, a los servicios necesarios para proveer la funcionalidad requerida. FUSION@ actúa como un intérprete entre las aplicaciones y los servicios, por lo que los usuarios pueden ejecutar funcionalidades programadas en prácticamente cualquier lenguaje de programación.

A través de un ejemplo es posible describir de forma sencilla el enfoque de FUSION@. *"Un usuario necesita calcular la suma de dos números y desea hacerlo desde su dispositivo móvil (por ejemplo, una PDA) conectado de forma remota al sistema. El usuario ejecuta la aplicación que contiene una serie de fórmulas, entre ellas la suma. El usuario selecciona la función deseada, introduce los sumandos y pulsa un botón para obtener el resultado. Cuando el usuario pulsa dicho botón, la aplicación envía una petición a la plataforma de agentes para que ésta encuentre un servicio que procese la petición. Los agentes en la plataforma seleccionan el servicio adecuado y le envían una nueva petición para que procese la función determinada (en este caso, la suma de los dos números). El servicio ejecuta la función y envía el resultado a la plataforma de agentes, que a su vez reenvía el resultado a la aplicación que solicitó el servicio".* Este ejemplo se muestra de forma gráfica en la **Figura 4.2**.

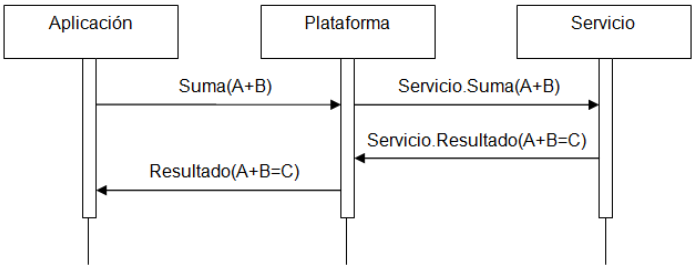

Figura 4.2. Ejemplo de petición de un servicio usando FUSION@

Resulta obvio que invocar un servicio remoto para ejecutar una simple suma de dos números no es la opción más adecuada. Sin embargo, la situación cambia si por ejemplo, en lugar de una suma se trata de un proceso complejo que involucre técnicas de Inteligencia Artificial, como algoritmos genéticos, minería de datos, redes neuronales, etc. De ser así, la limitada capacidad de procesamiento del dispositivo móvil haría imposible su ejecución. En estas circunstancias, el servicio podría ejecutarse en un ordenador convencional y ser invocado de forma remota por el dispositivo del usuario.

4.2. Aplicaciones y Servicios

Las aplicaciones y los servicios conforman la capa de aplicación y presentación de FUSION@. Estos integran las principales funcionalidades del sistema de cara a los usuarios y desarrolladores, por ejemplo las interfaces de usuario, algoritmos matemáticos, obtención de información mediante sensores, etc.

Las aplicaciones consisten en todos aquellos programas que pueden ser utilizados para acceder a las funcionalidades del sistema. Las aplicaciones son

dinámicas y adaptables al contexto, reaccionando de forma distinta ante determinadas situaciones (por ejemplo, si se encuentra en una habitación o en otra, o si el usuario tiene alguna discapacidad) y el tipo de servicio solicitado. Las aplicaciones pueden ser ejecutadas localmente o de forma remota, incluso desde dispositivos móviles con limitada capacidad de procesamiento, debido a que las tareas de cómputo se delegan a los agentes y a los servicios.

Los servicios representan el grueso de las funcionalidades del sistema a nivel de procesamiento, entrega y adquisición de información. Los servicios pueden definirse como conjuntos de actividades que pretenden responder a las necesidades de quien los invoca de forma local o incluso de forma remota. FUSION@ cuenta con un directorio flexible de servicios, de manera que puedan ser modificados, añadidos o eliminados dinámicamente y bajo demanda. Un servicio puede ser prácticamente cualquier programa que ejecute una determinada tarea y que además comparte sus recursos con la plataforma. Estos programas proporcionan métodos para acceder a bases de datos, administrar conexiones, analizar datos, obtener información de dispositivos externos (sensores, lectores, pantallas, etc.), publicar información, o incluso hacer uso de otros servicios para ejecutar una determinada tarea. Por ejemplo, un servicio puede consistir en una simple operación matemática o incluso en un complejo algoritmo de visualización, entre otros. De esta forma, los servicios reciben de las aplicaciones (a través de la plataforma de agentes) los valores que deben ser procesados para después devolver el resultado, liberando a las aplicaciones de gran carga computacional. Los desarrolladores tienen la libertad de utilizar cualquier lenguaje de programación, con el único requerimiento de seguir el protocolo de comunicación definido por FUSION@.

Uno de los objetivos de FUSION@ es lograr que tanto los servicios como las aplicaciones sean reutilizables e independientes del sistema al que ofrecen sus funcionalidades, así como del lenguaje de programación de la plataforma de agentes. La **Figura 4.3** muestra una representación de este enfoque, en donde

una misma aplicación es capaz de hacer peticiones a distintos sistemas y estos a su vez a los servicios disponibles.

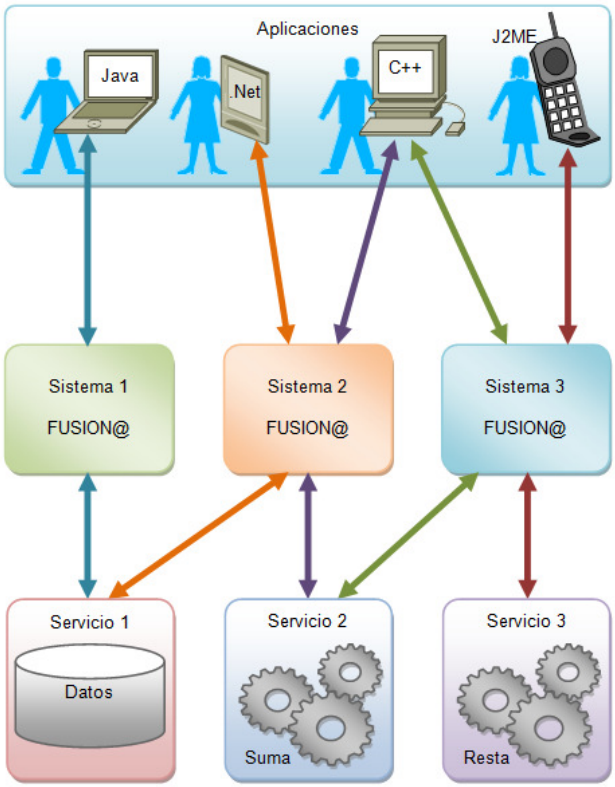

Figura 4.3. Reutilización e independencia de las funcionalidades en FUSION@

La diferenciación entre aplicaciones y servicios se debe principalmente para facilitar a los desarrolladores el distinguir las funcionalidades y requerimientos del sistema en términos de carga computacional y distribución de recursos informáticos. De esta forma, las funcionalidades que requieren una elevada carga computacional deben definirse como servicios, mientras que las funcionalidades que presenten una alta interacción con los usuarios, y que generalmente deban funcionar en tiempo de ejecución, deben plantearse como aplicaciones, aunque la decisión final depende de las necesidades y características de cada sistema.

4.3. Protocolo de Comunicación

FUSION@ implementa un protocolo de comunicación basado en el paso de mensajes. Este protocolo permite a las aplicaciones y a los servicios comunicarse directamente con la plataforma de agentes. El protocolo es totalmente abierto e independiente de cualquier lenguaje de programación y se basa en la especificación SOAP (*Service Oriented Architecture Protocol*) (Cerami, 2002) para encapsular todos los mensajes entre los servicios, las aplicaciones y la plataforma de agentes. La sintaxis básica de un mensaje SOAP es la siguiente:

```
1  <?xml version="1.0"?>
2  <soap:Envelope
3  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
4  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
5  <soap:Header>
6          ...
7  </soap:Header>
8  <soap:Body>
9          ...
13 </soap:Body>
14 </soap:Envelope>
```

El elemento raíz del documento es el elemento sobre (*Envelope*), el cual encapsula todos los elementos del mensaje. El sobre puede contener un elemento encabezado (*Header*) que contiene información sobre el mensaje, por ejemplo, la descripción de quién compuso el mensaje y el destinatario del mismo. El mensaje debe contener un elemento cuerpo (*Body*), el cual contiene los datos propios del mensaje, como puede ser una cadena de caracteres o determinadas variables.

En FUSION@, los mensajes SOAP se utilizan para comunicar los servicios y las aplicaciones con la plataforma de agentes. Sin embargo, la comunicación entre los agentes dentro de la plataforma sigue la especificación propia de cada

plataforma, por ejemplo, ACL en JADE, ICL en OAA, o KQML en RETSINA. En el caso de JADE, los mensajes ACL son objetos, por lo que se hace necesario utilizar protocolos como RMI (*Remote Method Invocation*) para invocar métodos en objetos remotos. La **Tabla 4.1** muestra los parámetros básicos de un mensaje FIPA ACL.

Tabla 4.1. Parámetros básicos de un mensaje FIPA ACL

Parámetro	Categoría
performative	Tipo de comunicación
sender	Participante en la comunicación
receiver	Participante en la comunicación
reply-to	Participante en la comunicación
content	Contenido del mensaje
language	Descripción del contenido
encoding	Descripción del contenido
ontology	Descripción del contenido
protocol	Control de la conversación
conversation-id	Control de la conversación
reply-with	Control de la conversación
in-reply-to	Control de la conversación
reply-by	Control de la conversación

Por ejemplo, la estructura de un mensaje ACL en donde el agente1 informa al agente2 mediante un mensaje con un parámetro *msg* para que el agente2 responda con el contenido de *respuesta*, y que se utilice el lenguaje semántico de FIPA y la ontología *fipa-acl* es la siguiente:

```
1  (inform
2  :sender agente1
3  :receiver agente2
4  :content
5  (mensaje msg)
6  :in-reply-to mensaje
7  :reply-with respuesta
```

```
8   :language fipa-sl
9   :ontology fipa-acl)
```

Las aplicaciones pueden hacer uso de otras plataformas de agentes para comunicarse directamente con los agentes de FUSION@ utilizando mensajes SOAP. Como se muestra en la **Figura 4.4**, el intercambio de mensajes no se realiza de forma directa entre las aplicaciones y los servicios, sino que todos los mensajes deben ser gestionados por la plataforma de agentes. De esta forma se obtiene un mayor control sobre las peticiones al validar todos y cada uno de los mensajes entre las aplicaciones y los servicios. Los agentes en la plataforma se encargan de seleccionar el servicio más óptimo para realizar cada tarea, lo que facilita al usuario especificar el servicio deseado.

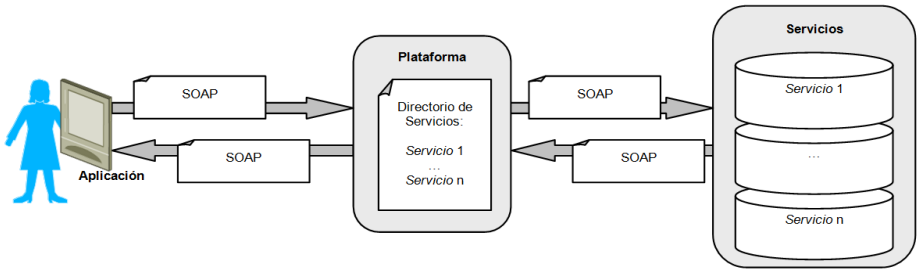

Figura 4.4. Intercambio de mensajes en FUSION@

Cuando la plataforma invoca a un servicio, lo hace enviando un mensaje SOAP a dicho servicio. El mensaje es recibido por el servicio y se crea un nuevo hilo para realizar la tarea. El hilo está asociado al socket, el cual mantiene la comunicación abierta hasta que la tarea sea completada y el resultado haya sido enviado a la plataforma. Este método permite a los servicios procesar múltiples peticiones al mismo tiempo. Sin embargo, puede haber situaciones en donde la multitarea no sea posible, por ejemplo, en servicios que demanden alto grado de procesamiento, lo que implicaría reducir significativamente el rendimiento del servicio. En este caso, la plataforma consulta el estado del servicio, el cual informa a la plataforma de si está ocupado o disponible. Si el servicio se

encuentra procesando una tarea, no podrá recibir nuevas peticiones hasta que haya completado dicha tarea. La plataforma debe entonces buscar otro servicio que pueda procesar la petición, o esperar a que el servicio esté nuevamente disponible.

4.4. Plataforma de Agentes

La plataforma de agentes es el núcleo de FUSION@, integrando un conjunto de agentes deliberativos BDI con comportamientos y funcionalidades independientes. Una característica importante de FUSION@ es que los agentes actúan como controladores y administradores de todas las funcionalidades del sistema, desde aplicaciones, servicios, comunicaciones y rendimiento, hasta las capacidades de razonamiento y toma de decisiones. Los agentes son capaces de modificar su comportamiento de acuerdo con las preferencias de los usuarios, los conocimientos adquiridos de interacciones anteriores, así como las opciones disponibles para responder a una situación determinada.

Toda la información relacionada con los servicios está almacenada en un directorio utilizado por la plataforma para invocar dichos servicios. Este directorio es flexible, por lo que los servicios pueden ser modificados, agregados o eliminados de forma dinámica. Los servicios están siempre en "modo de escucha" para recibir cualquier petición de la plataforma. Es necesario establecer una conexión permanente entre los servicios, las aplicaciones y la plataforma de agentes a través de sockets. Cada servicio debe tener un puerto de escucha para poder recibir las peticiones de la plataforma.

Una de las ventajas de utilizar FUSION@ es que, al no estar restringidas las aplicaciones por un determinado lenguaje de programación, se amplía el abanico

de posibilidades para desarrollar interfaces sobre un mayor número de dispositivos. De esta forma, y como se aprecia en la **Figura 4.5**, los usuarios pueden acceder al resto de las funcionalidades del sistema (servicios remotos y locales) desde ordenadores portátiles, teléfonos móviles, PDAs, etc. La estructura de componentes en FUSION@ es modular, siendo los agentes quienes gestionan todas las funcionalidades y comunicaciones, tanto de los sistemas como de la propia arquitectura.

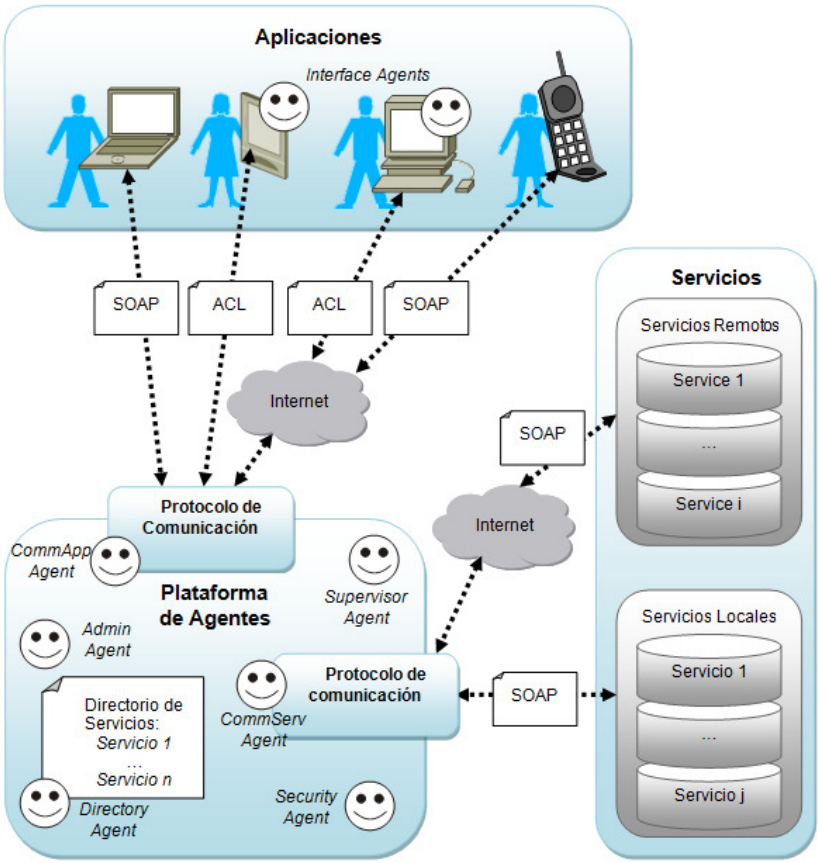

Figura 4.5. Estructura de componentes en FUSION@

A continuación, se describen los distintos tipos de agentes que componen la arquitectura FUSION@.

4.4.1. Arquitectura de Agentes

Los tipos de agente definidos por FUSION@ proporcionan los mecanismos adecuados para la construcción de entornos de Inteligencia Ambiental. En este tipo de entornos, la computación, comunicación e información deben ser ubicuas y transparentes para los usuarios. Asimismo, la interacción humano-sistema debe realizarse de manera natural y no intrusiva. Para lograrlo, es necesario que los sistemas cuenten con cierto grado de inteligencia y sean capaces de aprender de forma automática para adaptarse a las necesidades de los usuarios. A continuación, se describen los distintos tipos de agentes definidos en FUSION@:

- ***CommApp Agent.*** Gestiona todos los mensajes entre las aplicaciones y la plataforma de agentes. Recibe todas las peticiones que las aplicaciones envían a la plataforma y envía las respuestas una vez procesadas por los servicios. Este agente se encuentra siempre "en escucha" para poder recibir todas las peticiones. Las aplicaciones envían peticiones al agente, solicitando un determinado servicio. El agente entonces crea un nuevo hilo de comunicación entre la aplicación y la plataforma. Este hilo permanece abierto hasta que se responda a la petición lanzada por la aplicación. Todos los mensajes son analizados por el *Security Agent* para revisar su estructura y sintaxis. Una vez analizados los mensajes, el *CommAp Agent* reenvía todas las peticiones al *Admin Agent* el cual se encarga de negociar dichas peticiones.
- ***CommServ Agent.*** Gestiona todos los mensajes intercambiados entre los servicios y la plataforma de agentes. Sus funcionalidades son muy similares a las del *CommApp Agent*, pero a la inversa. El *CommServ Agent* está siempre "en escucha", esperando las respuestas que proporcionan los servicios a las peticiones hechas desde la plataforma. Cuando este agente invoca a un servicio, se crea un hilo de comunicación constante

entre la plataforma y el servicio, el cual permanece abierto hasta que el servicio responda a la petición enviada. Todos los mensajes son analizados por el *Security Agent* para revisar su estructura y sintaxis. El *CommServ Agent* comprueba periódicamente el estado de los servicios para saber si están disponibles, ocupados, o incluso bloqueados.

- ***Directory Agent.*** Administra el directorio (lista) de servicios que pueden ser utilizados por el sistema. Es importante señalar que en los entornos de Inteligencia Ambiental se manejan gran cantidad de datos delicados, sobre todo de los propios usuarios. Por lo tanto, la seguridad en el manejo de la información es una característica muy importante que debe tenerse en cuenta al desarrollar cualquier sistema basado en la AmI (Snidaro, *et al*., 2007). En este sentido, FUSION@ no implementa mecanismos para el descubrimiento de servicios. El directorio de servicios sólo puede modificarse de forma manual, sin embargo, los servicios pueden ser agregados, eliminados o modificados de forma dinámica. Para agregar un nuevo servicio, es necesario incluir de forma manual la información del servicio dentro del directorio que administra el *Directory Agent*. Entonces, el *CommServ Agent* envía un mensaje *ping* al servicio. Si el servicio responde al mensaje *ping*, el servicio se da de alta en la plataforma. Aún cuando el directorio es en cierto modo estático, los agentes son capaces de elegir el mejor servicio disponible en caso de haber múltiples servicios que puedan procesar una determinada petición. El directorio contiene la información de todos los servicios disponibles. El nombre y descripción del servicio, los parámetros requeridos, y la ubicación del servicio es alguna de la información estática almacenada en el directorio. También existe información dinámica que se actualiza constantemente, por ejemplo: el rendimiento del servicio (tiempo promedio para responder a las peticiones); el número de ejecuciones anteriores; y finalmente la calidad del servicio (QoS: *Quality of Service*). El *QoS* es un valor entre 0.00 y 1.00 para todos los servicios. Todos los nuevos servicios tienen un valor de 1.00. Este

valor decrece cada vez que el servicio presenta un funcionamiento deficiente, por ejemplo: cuando se bloquea, cuando no se encuentra el servicio, cuando tarda demasiado en comparación a ejecuciones anteriores, etc. El valor del *QoS* aumenta cada vez que el servicio procesa de forma exitosa una nueva petición.

- ***Supervisor Agent.*** Supervisa el correcto funcionamiento de todos los agentes en la plataforma, enviando de forma periódica mensajes de consulta para conocer su estado. Si un agente no responde, el *Supervisor Agent* "mata" al agente y crea una nueva instancia de dicho agente.
- ***Security Agent.*** Analiza la estructura y sintaxis de todos los mensajes intercambiados entre la plataforma, los servicios y las aplicaciones. Si un mensaje no es correcto, informa al agente correspondiente (*CommApp Agent* o *CommServ Agent* según sea el caso) de que el mensaje no puede ser procesado. En caso de que un mensaje incorrecto provenga de un servicio, el *Security Agent* informa al *Directory Agent* para que modifique el valor *QoS* del respectivo servicio.
- ***Admin Agent.*** Decide los servicios que deben ser invocados para procesar las peticiones recibidas desde las aplicaciones. Los usuarios pueden indicar a la plataforma el servicio específico que debe ser invocado. Otra opción consiste en dejar la decisión al *Admin Agent* para que, basándose en parámetros como el tipo de petición y el *QoS* de cada servicio, seleccione el servicio más adecuado para responder a determinada petición. Los mecanismos para la toma de decisiones son descritos en profundidad en la sección 4.4.2 de este capítulo. El *Admin Agent* cuenta con una lista dinámica de direcciones para el envío de mensajes entre las aplicaciones y los servicios. Por otra parte, comprueba periódicamente el adecuado funcionamiento de los servicios, indicando al *CommServ Agent* que haga peticiones de prueba (mensajes *ping*) a todos los servicios. Si un servicio no responde, el *Admin Agent* busca un servicio alternativo para satisfacer la petición realizada e informa al *Directory Agent* para que modifique el *QoS* del servicio fallido.

- ***Interface Agent.*** Este tipo de agente está diseñado para ser embebido en las aplicaciones de los usuarios. Estos agentes no requieren la utilización de mensajes SOAP cuando son desarrollados bajo una misma plataforma, por ejemplo JADE. En este caso, las peticiones son directamente enviadas al *CommServ Agent* mediante mensajes ACL y a su vez al *Security Agent* para analizar el contenido de los mensajes. El resto del proceso coincide con el ya descrito, pasando las peticiones por el *Admin Agent* y después por el *CommServ Agent* para invocar el servicio adecuado. Los *Interface Agent* deben ser lo suficientemente simples para ser ejecutados incluso en dispositivos móviles, como PDAs o teléfonos móviles, debido a que toda la carga de procesamiento se delega a los servicios. Este tipo de agente es fundamental en los desarrollos basados en la Inteligencia Ambiental, ya que interactúa directamente con los usuarios. Puede haber tantos *Interface Agents* como sean necesarios en cada sistema. El diseño de estos agentes es muy importante para que los usuarios puedan comunicarse de forma sencilla, transparente y en cierto modo inteligente con el resto de los componentes del sistema.

FUSION@ es una arquitectura totalmente abierta, en la cual los agentes no están definidos de una manera estática. La flexibilidad de FUSION@ permite que los desarrolladores puedan agregar nuevos tipos de agentes o extender las funcionalidades de los existentes, dependiendo de las necesidades de cada proyecto. Sin embargo, para mantener la naturaleza distribuida de FUSION@, gran parte de las funcionalidades de los agentes deben ser modeladas como servicios, liberándolos de tareas que puedan disminuir su rendimiento y adecuado funcionamiento.

En FUSION@, los servicios solo pueden ofrecer sus funcionalidades a través de los agentes. Además, los servicios son específicos para cada sistema y deben estar siempre disponibles para el sistema que hace uso de ellos.

4.4.2. Mecanismos de Razonamiento y Planificación

Una de las principales características de FUSION@ es el uso de mecanismos de razonamiento y planificación. Estos mecanismos proveen a FUSION@ con mayores capacidades para solucionar problemas de forma dinámica. Los agentes deciden cuales son los servicios más adecuados para atender a las peticiones realizadas tanto por los usuarios como desde la propia arquitectura. Asimismo, facilitan a los desarrolladores el determinar el número óptimo de servicios que deben implementarse de cara a optimizar el rendimiento del sistema.

4.2.2.1. Optimización de Servicios

FUSION@ implementa un mecanismo para determinar el número óptimo de servicios que debe haber para satisfacer la demanda de peticiones. El número de servicios disponibles en el sistema se estima de manera dinámica. Se pretende que el número de servicios arrancados se adecúe a dicha demanda tratando de garantizar que el factor de utilización del sistema sea menor que 1. Esta estimación se lleva a cabo mediante el uso de la teoría de colas. *The queueing theory* tiene su origen en 1909 con el estudio de las comunicaciones telefónicas en Dinamarca (Erlang, 1909). Se llevó a cabo un estudio para dar servicio a una serie de clientes bajo una demanda incierta. La teoría de colas se puede definir como el estudio matemático del comportamiento de las líneas que se generan por los usuarios para el acceso a unos determinados servicios. Para realizar dicho estudio, se crean una serie de modelos matemáticos que tratan de describir el comportamiento de las colas generadas por los usuarios. En estos estudios, se tienen en cuenta diversas variables, intentando crear un equilibrio entre el coste

y el tiempo de espera (Tadj, 1995). Esta teoría es aplicable a diversos campos (Shanthikumar, *et al*, 2007). Una de estas aplicaciones es el balanceo de carga de servidores Web (Menasce, 2002), en donde se tiene que decidir el número adecuado de servidores para satisfacer la demanda de peticiones, teniendo en cuenta los costes.

En un sistema de colas se pueden distinguir diferentes parámetros que determinan el funcionamiento (Antoniol, *et al*., 2004):

- **Tasa de llegada.** Consiste en la distribución de llegada que sigue la entrada de usuarios al sistema. Normalmente la función de distribución seguida es una de *Poisson*.
- **Tiempo de servicio.** Es el tiempo empleado en llevar a cabo el servicio una vez que el cliente es atendido. La distribución seguida normalmente es una exponencial.
- **Servicios.** Uno o varios servicios con la posibilidad de ausencia.
- **Disciplina de la cola.** Representa el modelo de entrada/salida de la cola. Entre los modelos más conocidos se encuentran FIFO (*First In First Out*), LIFO (*Last In First Out*), RR (*Round Robin*), RSS (*Random Selection of Service*), etc.
- **Capacidad de la cola.** Indica si longitud de la cola es finita o infinita.
- **Tamaño de la fuente de entrada al sistema.** Corresponde al tamaño de las peticiones que se hacen a los servicios.

Para la especificación de los parámetros se suele seguir la notación de Kendall (*Kendall's notation*). Esta notación consiste en una séxtupla que almacena los datos indicados anteriormente. Se representa mediante la siguiente sintaxis 1/2/3/4/5/6 (Tadj, 1995). Los valores de cada uno de los términos de la expresión varían de la forma siguiente para cada uno de los términos:

- M, *Poisson*; D, tiempos determinísticos; $E_{k,}$ distribución de Erlang; G, distribución general.

- 1 para un sólo servicio; S, para varios servicios.
- En la disciplina se ponen las siglas del método seleccionado (FIFO, LIFO, etc.).
- En la capacidad de la cola se pone un número ó ∞.
- En el número máximo de consumidores en el sistema, un número ó ∞.

Existen muchas variantes en la teoría de colas, según sean las opciones elegidas. El modelo M/M/1 es el más sencillo de usar (Kühn, *et al*., 2008). Este modelo es aquel en el que la tasa de llegadas sigue una distribución de *Poisson* (1), la de servicio una exponencial (2), y el servidor es único.

$$P(x=k)=\frac{\lambda^k e^{-\lambda}}{k!} \tag{1}$$

$$F(x)=1-e^{-\lambda x} \tag{2}$$

Debido a la naturaleza distribuida de FUSION@, el número de servicios replicados y de servidores es variable. Por lo tanto, se presentan modelos M/M/s o bien M/G/s (Tadj, 1995) (Roth, 1994), suponiendo siempre que la llegada de las peticiones sigue una distribución de *Possion* (1), pues dicha distribución mide el número de llegadas en un determinado tiempo y es la más utilizada (Menasce, 2002). Además, el tiempo para completar las peticiones sigue una distribución exponencial, puesto que para el modelo M/M/s donde S=1,2,3, … es el número de servicios y dada una tasa de llegada $\lambda_n = \lambda$ =constante, la tasa de servicio cuando se encuentran *n* procesos viene definida por la siguiente ecuación (3).

$$\mu_n=\begin{cases} n\mu & n=1,2,\ldots,s-1 \\ s\mu & n\geq s \end{cases} \tag{3}$$

μ representa el tiempo medio para completar una petición para los *S* servicios arrancados. Este valor depende tanto de los servicios arrancados como de la máquina en la que se encuentran. En los cálculos no se hace distinción de los tiempos de servicio independientes ya que no se puede estimar los servicios ocupados y por tanto la tasa de servicio general cuando el número de clientes en el sistema es inferior al número de servicios. Suponiendo que en el sistema se encuentra en condiciones de estabilidad, es decir se cumple que el factor de utilización es menor que 1 $\rho = (\lambda / \mu s) < 1$, la probabilidad de que hayan *n* clientes (P_n) en el sistema viene dado por la siguiente ecuación (4):

$$P_n = c_n P_0 = \begin{cases} \dfrac{\lambda^n}{n!\mu^n} P_0 & n = 0,1, \ldots, s-1 \\ \dfrac{\lambda^s}{s!\mu^s}\left(\dfrac{\lambda}{s\mu}\right)^{n-s} P_0 & n \geq s \end{cases} \tag{4}$$

Donde,

$$P_0 = \frac{1}{\sum_{n=0}^{s-1}\frac{\lambda^n}{n!} + \sum_{n=s}^{\infty}\frac{\lambda^s}{s!\mu}\left(\frac{\lambda}{s\mu}\right)^{n-s}} = \frac{1}{\sum_{n=0}^{s-1}\frac{\lambda^n}{n!} + \frac{\lambda^s}{s!\mu}\frac{1}{1-\frac{\lambda}{s\mu}}} \tag{5}$$

$$c_n = \begin{cases} \dfrac{\lambda^n}{n!\mu^n} & n = 1,2, \ldots, s-1 \\ \dfrac{\lambda^s}{n!\mu^s}\left(\dfrac{\lambda}{s\mu}\right)^{n-s} & n \geq s \end{cases} \tag{6}$$

En caso de que no se cumpla la condición del factor de utilización del sistema, será necesario aumentar el número de servicios del sistema para lo que se realiza el cálculo de las ecuación (7). Una vez definida la probabilidad de que hayan *n* clientes en el sistema, se puede definir los parámetros que realmente

importan, en este caso se trata del número de elementos que se van a encontrar en la cola del sistema. Viene definido por la siguiente ecuación (7):

$$L_q = \sum_{n=s}^{\infty}(n-s)P_n = P_0 \frac{\lambda^s}{s!\mu^s}\frac{\lambda}{s\mu}\frac{1}{\left(1-\frac{\lambda}{s\mu}\right)^2} \quad (7)$$

A partir de la ecuación (7) se puede definir el tiempo medio de espera de los clientes en la cola del sistema.

$$W_q = \frac{L_q}{\lambda} \quad (8)$$

Para determinar el número de servicios óptimo, se realiza una estimación en la que se minimiza la función de coste que depende del número de servicios arrancados y del tiempo de espera en la cola. La función se define de forma particular para cada servicio dependiendo de los costes reales de cada uno de los servicios del sistema, aunque se proporciona la siguiente función de coste (9).

$$f(W_q, P_0, .., P_{s-1}, \mu) = \beta W_q + \sum_{j=0}^{s-1} P_j \mu(s-j) \quad (9)$$

Siendo β una constante comprendida entre 0 y 1. Por defecto, el valor seleccionado es 0.5. Una vez defina la función a optimizar se inicia el algoritmo que determina los servicios a tener activos:

1. Sean Para cada uno de los servicios *i* de la máquina que se encuentran sin arrancar estimar P_{in}^s suponiendo el servicio *i* iniciado
2. Calcular el valor de $f_i^s(W_q, P_0, .., P_{s-1}, P_i^s, \mu)$ para cada uno de los servicios anteriores.

3. Seleccionar el menor valor de los f_i^s
4. Si el valor es menor que el de $f(W_q, P_0, .., P_{s-1}, \mu)$ ir al paso 9.
5. Para cada uno de los servicios *i* que se encuentran arrancados estimar P_{in}^r suponiendo el servicio parado.
6. Calcular el valor de $f_i^r(W_q, P_0, .., P_{s-2}, \mu)$ para cada uno de los servicios arrancados.
7. Seleccionar el menor de los f_i^r
8. Si el valor seleccionado no es menor que el anterior finalizar el algoritmo
9. Iniciar o parar el servicio iniciado y volver al paso 1.

4.2.2.2. Gestión de Servicios

FUSION@, y en particular el *Admin Agent*, emplea un mecanismo compuesto por un conjunto de técnicas que le permite seleccionar el servicio más adecuado para atender a una petición en un momento determinado. Cuando una nueva petición es recibida, comienza el proceso para asignar el servicio más adecuado para responder a dicha petición, tomando en cuenta los siguientes parámetros: el valor *QoS*, las preferencias del usuario, y el tiempo de ejecución estimado. Los dos primeros parámetros son establecidos de antemano, por lo que no es necesario que los calcule el *Admin Agent*.

El tiempo de ejecución se estima mediante una red neuronal tipo RBF (*Radial Basis Function*). Existen otras redes neuronales que se pueden aplicar para la estimación del tiempo de ejecución, como es el caso de los perceptrones multicapa (MLP: *Multi-Layer Perceptron*), sin embargo, necesitan un tiempo de entrenamiento mayor que una red RBF. En la **Tabla 4.2** se muestra el resultado de una comparativa de las mediadas referente al *tiempo de ejecución* para la

estimación de tiempos. Esta tabla muestra el resultado de aplicar el test de Kruskal-Wallis (Kruskal, *et al*., 1952) sobre el *tiempo de ejecución* obtenido para 50 ejecuciones para cada uno de los algoritmos evaluados (RBF, MLP y Regresión Lineal Múltiple). En este caso, la mediana del *tiempo de ejecución* es menor para la regresión lineal múltiple, mientras que la RBF y la MLP no presentan variaciones significativas. El indicador "=" señala una igualdad en las medianas, "*" señala la desigualdad de medias, y finalmente "(-)" indica si el valor experimental de la fila es mejor que el de la columna.

Tabla 4.2. Comparativa de la mediana de los tiempos de ejecución para los algoritmos RBF, MLP y RLM

	RBF	**MLP**	**RLM**
RBF			
MLP	=		
RLM	*(-)	*(-)	

Posteriormente, se analizó el error de estimación por los diferentes métodos aplicando para ello Kruskal-Wallis sobre el error de estimación para las estimaciones del *tiempo de ejecución*. Los resultados obtenidos se muestran en la **Tabla 4.3**, en donde se aprecia que la red neuronal RBF proporciona resultados similares al MLP y mejora los obtenidos mediante RLM. Una vez más, el indicador "=" señala una igualdad de medianas, "*" una desigualdad, y "(+)" la mediana experimental de la fila mayor que la de la columna.

Tabla 4.3. Comparativa de la mediana del error de estimación para RBF, MLP y RLM

	RBF	**MLP**	**RLM**
RBF			
MLP	=		
RLM	*(+)	*(+)	

La red neuronal RBF está formada por tres capas, la capa de entrada, una intermedia/oculta y la de salida. El número de neuronas de la capa de entrada está definido por el número de entradas de cada tipo (parámetros) a cada uno de los servicios. En el caso de arreglos (*arrays*), cada elemento se cuenta como una entrada al servicio. El número de neuronas de la capa intermedia se determina dinámicamente, realizando un entrenamiento y validación cruzada. La validación cruzada se lleva a cabo mediante el método GCV (*Generalized Cross-Validation*) (Tiwari, *et al*., 2004).

La variación del número de neuronas de la capa intermedia se realiza siguiendo el siguiente algoritmo. Primeramente, se inicializan la lista de errores al vacío $e = \{\}$ y la lista de ángulos $\alpha = \{\}$. Se realiza el entrenamiento con una neurona en la capa intermedia y se almacena el valor del entrenamiento en e_1 en la lista de valores $e = e \cup e_1$. A continuación, se inicializa el número de neuronas de la capa intermedia. Si no se dispone de entrenamiento previo, se inicializa a $4n+1$, siendo n el número de neuronas de la capa de entrada. Por el contrario, si se dispone de entrenamiento previo, se inicializa a $2n+1$, donde n es el número de neuronas del entrenamiento previo. Sea r el número de neuronas actual de la capa intermedia. Con este valor se realiza el entrenamiento para r neuronas intermedias y se almacena el error en e_r $e = e \cup e_r$. Posteriormente, se realiza el entrenamiento para $r/2$ neuronas y se almacena el error en $e_r/2$ e = e $\cup$ $e_{r/2}$. Se calculan las rectas que pasan por los puntos $r_{1,r/2} = (e_r e_{r/2})$ $r_{1,r/2} = (e_1 e_{r/2})$ y se obtiene el ángulo $\alpha_{r/2}$ que forman ambas rectas. Se introduce $\alpha_{r/2}$ en la lista de ángulos calculados $\alpha = \alpha \cup_{r/2}$. Si sólo existe un ángulo en la lista de ángulos $\#\alpha = 1$ (# indica el número de elementos del conjunto), entonces se establece el nuevo valor de r a $r/4$ y se vuelve a realizar el entrenamiento. Si el $\#\alpha = 2$ siendo α_i el otro valor existente entonces: Si $i<r/2$ entonces $r=r/2$ y se vuelve a realizar el entrenamiento; Sino $r=r/2+r/4$ y se vuelve a realizar el

entrenamiento. Sino seleccionar los dos valores contiguos por la izquierda y derecha de $\alpha_{r/2}$ denotados por α_i y α_j de modo que:

- Sino si $\alpha_i > \alpha_j$
    i. Si $j=r/2+1$ se establece el valor de las neuronas a r
    ii. Sino establecer el nuevo valor de $r=r/2+(j-r/2)/2$
- Sino
    i. Si $i=r/2-1$ se establece el valor de las neuronas a r
    ii. Sino establecer el nuevo valor de $r=r/2-(r/2-i)/2$

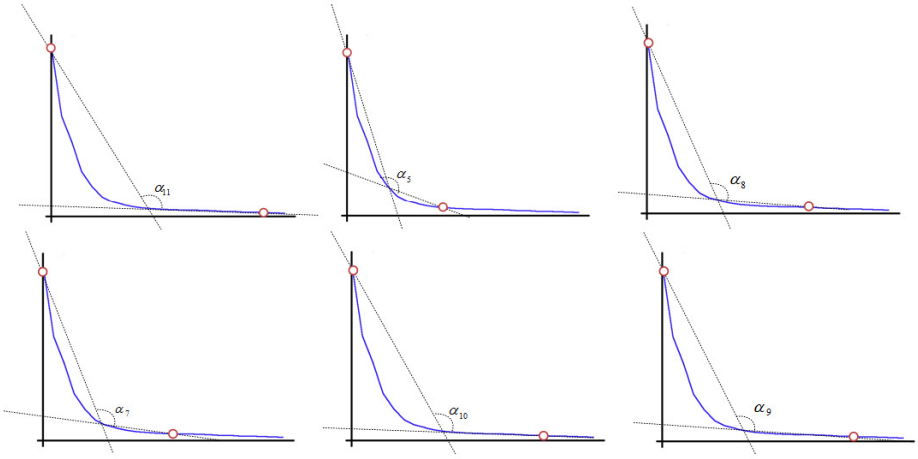

Figura 4.6. Progresión del número de neuronas de la capa intermedia

Finalmente, se vuelve a realizar el entrenamiento y se establece el valor de las neuronas de la capa intermedia a *r*. En la **Figura 4.6**, se muestra el progreso del algoritmo en la búsqueda del menor ángulo α. En el eje X se tiene el número de neuronas de la capa intermedia y en el eje Y el error en la validación cruzada. Se han representado todos los valores obtenidos en una simulación de dicho

cálculo, aunque para un caso real sólo es necesario calcular los valores que se obtienen en los conjuntos que se indican a continuación. Los conjuntos de valores estudiados en cada instante son los siguientes: (23, 11, 1) (11, 5, 1) (17, 8, 1) (14, 7, 1) (20, 10, 1) (18, 9, 1). El resultado final viene dado por (18, 9, 1), por lo tanto, el número de neuronas final de la capa intermedia es 9. La capa de salida consta de una única neurona, cuyo valor de salida es la predicción del tiempo.

El aprendizaje de la red consta de una fase de aprendizaje no supervisado para la capa intermedia y otra fase de aprendizaje supervisado para la capa de salida. En la etapa de entrenamiento se lleva a cabo una validación cruzada variando el número de neuronas de la capa intermedia, en el caso de que no se consigan nuevos resultados. En una primera etapa se realiza el entrenamiento de la capa intermedia para posteriormente entrenar la capa de salida. El reentrenamiento de la red se lleva a cabo cuando el tiempo promedio es *k* veces mayor o menor que el tiempo estimado para las *n* últimas ejecuciones. Estos valores son definidos a priori para cada sistema. A medida que se realiza el entrenamiento de la red, se realiza una validación cruzada, determinándose de esta manera el punto de finalización del entrenamiento. La validación cruzada se lleva a cabo mediante el método GCV (Tiwari, *et al*., 2004).

Una vez estimado el tiempo de ejecución de los procesos de la cola, se procede a realizar la asignación a los servicios disponibles. La asignación se realiza intentando maximizar el rendimiento en cuanto a los tiempos necesarios para responder a todas las peticiones. Debido a que la asignación tiene que ser en tiempo de ejecución, la asignación de los procesos debe de ser eficiente y adaptable dinámicamente. Por tal motivo, se ha optado por un modelo FIFO para cada uno de los servicios. Cada servicio tiene asociado una cola de espera en la que se van asignado los procesos que llegan a la cola común que gestiona el *Admin Agent*. La asignación a la cola de cada servicio se hace teniendo en cuenta los procesos asignados a cada una de las colas y la estimación del rendimiento

dado por (10). Se asigna el proceso (petición) a la cola, de modo que el rendimiento acumulado sumado al de la estimación del proceso sea mínimo.

$$r_i = (1 - QoS_i) \cdot t \cdot (1 - p_i) \tag{10}$$

Donde:

- r_i es la estimación del rendimiento del servicio *i*.
- QoS_i calidad del servicio *i*. Valor entre 0.00 y 1.00
- *T* tiempo estimado de duración
- p_i preferencias por el servicio *i*. Valor entre 0 o 1

Sea *n* el número de servicios, R_i el rendimiento acumulado para la cola *i*, un nuevo proceso *s* con estimaciones de rendimiento para el servicio *j* r_{ij} se asignará a la cola R_i que cumpla la siguiente condición siempre. Si el valor de la preferencia es igual a 1, sólo se tienen en cuenta las colas con dicho valor para la preferencia.

$$\min\{R_1 + r_{i1}, \ldots, R_n + r_{in}\} \tag{11}$$

Suponiendo que se asigna a la cola *k*, el nuevo rendimiento acumulado para la cola *k* será $R_k + r_{ik}$ y se continua con el siguiente proceso.

En la **Figura 4.7** se muestra una representación del funcionamiento del sistema de colas. Se tiene una cola principal asociada a un tipo de servicio, cada uno de los procesos de la cola tiene una estimación de rendimiento r_{ij}, y es asociado a una de las colas hasta que todas las colas estén ocupadas. De esta forma, es posible asignar las peticiones que entren al sistema a cada uno de los servicios disponibles, lo que se traduce en un mejor rendimiento y optimización de los recursos.

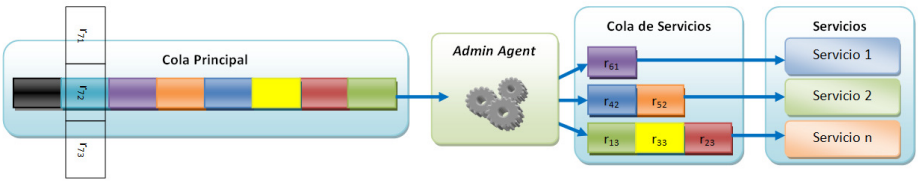

Figura 4.7. Asignación de servicios a través del sistema de colas

4.4.3. Agentes y Servicios Sensibles al Contexto

Aunque la optimización de las funcionalidades internas de FUSION@ es muy importante, también lo es una adecuada interacción con los usuarios y el entorno. Una incorrecta interacción conlleva a desarrollar complicados sistemas que no cumplen los requerimientos de la Inteligencia Ambiental desde el punto de vista de los usuarios. Por tal motivo, es necesario dotar a los agentes con la capacidad de obtener información física del contexto, así como de adaptarse a las características de los usuarios. Esto facilita a los usuarios el uso de dicha tecnología y de las propias funcionalidades del sistema.

Para que FUSION@ sea capaz de afrontar los retos que plantean los desarrollos basados en la Inteligencia Ambiental, es necesario que los agentes sean sensibles al contexto, obteniendo información tanto de los usuarios como del entorno. Una de las innovaciones propuestas por FUSION@ consiste en la utilización de un conjunto de tecnologías que permiten a los agentes obtener dicha información, centrándose en características definidas por la Inteligencia Ambiental como la ubicuidad, inteligencia y movilidad. De este modo, los agentes son capaces de cambiar su comportamiento ante determinadas situaciones. Por ejemplo, si un usuario se encuentra en una situación de riesgo, los agentes deberán reaccionar de forma automática, eligiendo una de las posibles soluciones con las que dispone el sistema para esa situación.

Por la propia naturaleza de FUSION@, siendo una arquitectura abierta y flexible, no es conveniente definir el uso de una tecnología en particular. Sin embargo, existen tecnologías de referencia que son tenidas en cuenta por FUSION@ que por su propio diseño permiten ser aplicadas en gran cantidad de entornos, en este caso de Inteligencia Ambiental.

La identificación de los usuarios es una pieza clave para una adecuada personalización de servicios e interacción con el entorno. Una de las tecnologías con mayor potencial es la identificación por radiofrecuencia (RFID). Es importante señalar que RFID es simplemente un método de identificación automática que almacena información en dispositivos llamados etiquetas (*tags*). El uso de dispositivos RFID permite a los agentes identificar a los usuarios y personalizar las acciones que deben tomar ante determinadas situaciones. Identificando a los usuarios, es posible crear comportamientos individualizados dependiendo de las características de cada uno de ellos.

Por otra parte, la percepción del entorno es fundamental para situar a los usuarios dentro del contexto y así personalizar la interacción del sistema ante determinadas situaciones. Para ello, las tecnologías domóticas proporcionan a la Inteligencia Ambiental la infraestructura necesaria para lograr una adecuada interacción entre los usuarios y el entorno. Los dispositivos para el control y la automatización permiten obtener información sobre el entorno y reaccionar físicamente sobre éste, expandiendo las capacidades de los usuarios y automatizando acciones cotidianas. Una de las tecnologías con un gran potencial de implementación en escenarios de Inteligencia Ambiental es ZigBee. Existe una gran cantidad de dispositivos ZigBee y su utilización en escenarios domóticos y de Inteligencia Ambiental es cada vez mayor. La versatilidad de esta tecnología permite dotar a los agentes prácticamente cualquier capacidad sensorial, por ejemplo, iluminación, humedad, humo, etc. El conjunto de tecnologías RFID, ZigBee y Wi-Fi representan un gran avance hacia el desarrollo de sistemas que cumplan con la visión de la Inteligencia Ambiental.

Es importante señalar que la simple utilización uso de nuevas tecnologías no es suficiente para que un sistema cumpla con los requisitos de la Inteligencia Ambiental. Por lo tanto, es necesario diseñar mecanismos e interfaces que hagan uso de dichas tecnologías y faciliten a los usuarios su utilización. En FUSION@, esto se logra a través de los *Interface Agents* o mediante servicios independientes. Los servicios obtienen información de forma constante y la envían a la plataforma de agentes para que sea procesada e interpretada.

Por su parte, los *Interface Agents* son una de las piezas más importantes para lograr una adecuada interacción con los usuarios. Este tipo de agentes interpretan la información enviada por la plataforma de agentes y la presentan a los usuarios a través de interfaces multimodales (vista, oído, tacto, etc.). Gracias a que las principales tareas de procesamiento de los *Interface Agents* son modeladas como servicios y procesadas por FUSION@, se amplía el espectro para diseñar interfaces que incluso puedan ser ejecutadas en dispositivos móviles y que se ajusten a las necesidades de los usuarios. Para entender mejor este concepto, se describe el siguiente ejemplo: un servicio de localización recibe constantemente información de dispositivos RFID. Esta información se envía periódicamente a la plataforma de agentes. La información se interpreta y envía al dispositivo móvil del usuario para saber su posición dentro una determinada área. De esta forma, el dispositivo del usuario no requiere hacer cálculos complejos para determinar la posición del usuario, ya que simplemente debe interpretar los datos que le envía la plataforma y mostrarlos al usuario de forma sencilla.

4.6. Conclusiones

La arquitectura FUSION@ integra la tecnología de agentes con una filosofía SOA, obteniendo un nuevo marco que presenta importantes avances en el desarrollo de sistemas basados en la Inteligencia Ambiental. FUSION@ facilita el desarrollo de sistemas multiagente flexibles, aprovechando las características de los agentes para modelar las funcionalidades de los sistemas y de los propios agentes como aplicaciones y servicios distribuidos e independientes. De esta forma, se obtienen agentes con menor carga computacional y se potencia la escalabilidad y reutilización de recursos. Por otra parte, el enfoque distribuido de FUSION@ aporta una mayor capacidad de recuperación ante errores. Además, los servicios como las aplicaciones son capaces de comunicarse de forma distribuida, incluso desde dispositivos móviles, independientemente del lenguaje de programación o el sistema operativo utilizado. FUSION@ es lo suficientemente flexible para incorporar de forma automática nuevas funcionalidades en entornos altamente dinámicos. FUSION@ también propone la utilización de las tecnologías inalámbricas ZigBee, RFID y Wi-Fi, un conjunto capaz de proporcionar a los agentes valiosa información del contexto de forma automática, tomando en cuenta conceptos definidos por la Inteligencia Ambiental, como la ubicuidad, la inteligencia, la movilidad y la disponibilidad de los recursos.

En el siguiente capítulo de esta memoria, se presenta un caso de estudio que consiste en el desarrollo de un sistema multiagente que hace uso de FUSION@ para mejorar diversos aspectos relacionados con los cuidados médicos y la asistencia de pacientes con la enfermedad de Alzheimer dentro de residencias geriátricas.

Capítulo

5

Caso de Estudio

En este capítulo se describe el caso de estudio que permite evaluar la arquitectura propuesta en el marco de la investigación presentada en este trabajo de tesis doctoral. El caso de estudio consiste en el desarrollo de ALZ-MAS (*ALZheimer Multi-Agent System*), un sistema multiagente enfocado a mejorar los servicios de asistencia y cuidados médicos para personas dependientes, especialmente con la enfermedad de Alzheimer. ALZ-MAS se ha desarrollado a partir de la base proporcionada por la arquitectura FUSION@, presentada en el capítulo 4 de esta memoria. En este sentido, ALZ-MAS está compuesto por un conjunto de agentes inteligentes con capacidades razonamiento, cooperación, planificación y toma de decisiones de forma dinámica. Las principales funcionalidades de ALZ-MAS y de los propios agentes que componen el sistema han sido modeladas como aplicaciones y servicios distribuidos, los cuales son gestionados por FUSION@.

La infraestructura tecnológica de ALZ-MAS hace uso de las tecnologías Wi-Fi, RFID y ZigBee. Estas tecnologías permiten a los agentes obtener información del contexto, es decir, tanto de los usuarios (constantes vitales, ubicación, actividades, medicación, etc.) como de su propio entorno (temperatura del ambiente, situaciones de riesgo, estado de la iluminación, etc.). Asimismo, los agentes son capaces de reaccionar de acuerdo a las características de una situación determinada, creando nuevos comportamientos y fijándose nuevas metas.

ALZ-MAS tiene como su pilar más importante a la Inteligencia Ambiental, por lo que requiere de un proceso de desarrollo compuesto por diversas etapas secuenciales. Este proceso se inicia modelando el contexto, etapa fundamental en todo desarrollo de Inteligencia Ambiental, abordando el problema de forma global para concluir con una definición más detallada del problema. Posteriormente, se define el modelo del sistema utilizando herramientas para el análisis y diseño de sistemas orientadas a agentes Gaia y SysML, de forma que se pueda dar mayor validez, utilidad y reutilización al modelo.

Este capítulo se divide en varias secciones. En la sección 5.1 se describen las principales características del sistema ALZ-MAS. La sección 5.2 muestra el proceso que ha permitido modelar el contexto del sistema, sentando las bases para modelar las características de ALZ-MAS, a través del proceso de análisis y diseño que se presenta en la sección 5.3. La sección 5.4 describe el proceso de integración de FUSION@ y ALZ-MAS. La sección 5.5 describe, de forma general, la fase de implementación del sistema ALZ-MAS en un entorno real. En la sección 5.6 se muestran algunos de los resultados más importantes obtenidos, y finalmente, se presentan las conclusiones de este capítulo.

5.1. Descripción del Sistema ALZ-MAS

Los agentes y los sistemas multiagente se están convirtiendo en una realidad en los entornos de dependencia, especialmente en los cuidados médicos. Muchos de los desarrollos basados en agentes se enfocan en la monitorización de pacientes, supervisión de tratamientos y minería de datos clínicos. Lanzola, *et al*. (1999) presentan una metodología que facilita el desarrollo de agentes interoperables para entornos médicos y proponen un modelo genérico para implementarlos. Este modelo puede ser especializado para soportar los requerimientos de los sistemas hospitalarios. Por su parte, Meunier (1999) propone utilizar maquinas virtuales para crear agentes móviles utilizando un paradigma de programación funcional. Estas maquinas virtuales proporcionan a los desarrolladores una plataforma sobre la cual desarrollar aplicaciones y agentes móviles, orientadas sobre todo a aplicaciones médicas y procesamiento de imágenes. Aunque esta propuesta es interesante, resulta inviable debido a los problemas de seguridad que afectan a los agentes móviles. Además, no define una alternativa para localizar a los pacientes o para crear estrategias de planificación. Miksch (1999) ha desarrollado un sistema basado en agentes para ayudar a los pacientes a obtener mejores tratamientos. El sistema gestiona la información relacionada con los posibles problemas crónicos de los pacientes y proporciona consejos sanitarios a través de una red de información en línea. Por su parte, Decker, *et al*., (1998), proponen una solución multiagente para incrementar la eficiencia hospitalaria a través de diversas técnicas de planificación. Esta solución intenta optimizar el tiempo de atención a los pacientes en el hospital, utilizando mecanismos con restricciones de tiempo y recursos tanto humanos como técnicos. Existen además proyectos como RoboCare (Pecora, *et al*., 2007) y TeleCARE (Camarinha-Matos, *et al*., 2007) que hacen uso de agentes y sistemas

multiagente para proporcionar asistencia y cuidados médicos a personas dependientes.

Estos desarrollos y muchos otros amplían las posibilidades y estimulan los esfuerzos en la investigación para mejorar la asistencia y los cuidados médicos a personas dependientes. Sin embargo, todos ellos se encuentran en fases tempranas de desarrollo o simplemente no han llegado a ser del todo implementados. Además, ninguno cubre las necesidades de los sistemas basados en la Inteligencia Ambiental, desde el punto de vista de la interacción con los usuarios.

Por tal motivo, se ha desarrollado ALZ-MAS (*ALZheimer Multi-Agent System*), un sistema multiagente que hace uso de la arquitectura FUSION@ y que está diseñado en torno al paradigma de la Inteligencia Ambiental. Este sistema se centra en mejorar los cuidados médicos y servicios de asistencia de ancianos y personas que presenten algún grado de dependencia, especialmente pacientes que padezcan la enfermedad de Alzheimer.

El sistema ALZ-MAS está compuesto por un conjunto de agentes, los cuales hacen uso de complejos mecanismos de razonamiento y planificación. Estos mecanismos generan planes e informes basándose en el historial médico de los pacientes y las indicaciones del personal médico, distribuyendo los tiempos y asignando tareas entre el personal médico de forma automática y dinámica sin afectar sus jornadas laborales. Los agentes en ALZ-MAS son capaces de actuar de forma autónoma y tienen la capacidad de ser ejecutados en dispositivos inalámbricos, como asistentes personales (PDAs) o teléfonos móviles, pero es necesario que se comuniquen entre ellos para la toma de decisiones y la generación de planes.

Como se aprecia en la **Figura 5.1**, ALZ-MAS hace uso de una infraestructura distribuida que se compone de un conjunto de tecnologías inalámbricas propuestas por FUSION@, entre ellas la identificación por radio frecuencia (RFID), ZigBee, Wi-Fi, y dispositivos móviles. Estas tecnologías proporcionan a

los agentes información valiosa sobre el contexto, permitiendo al sistema ALZ-MAS adaptarse a las características tanto de los usuarios como del propio entorno de forma automática y en tiempo de ejecución.

El personal médico, en especial los enfermeros, cuenta con un sistema capaz de gestionar de forma distribuida la información personal y expedientes médicos de los pacientes, llevar un seguimiento de las indicaciones y tratamientos prescritos por los doctores, planificar la jornada laboral del personal encargado de los cuidados de los pacientes, realizar tareas de monitorización y localización de los pacientes y del personal de forma automática, etc. Además, se supervisa la evolución de los pacientes ante los tratamientos prescritos por los doctores, informando a los enfermeros sobre cualquier anomalía y generando dinámicamente citas con los doctores.

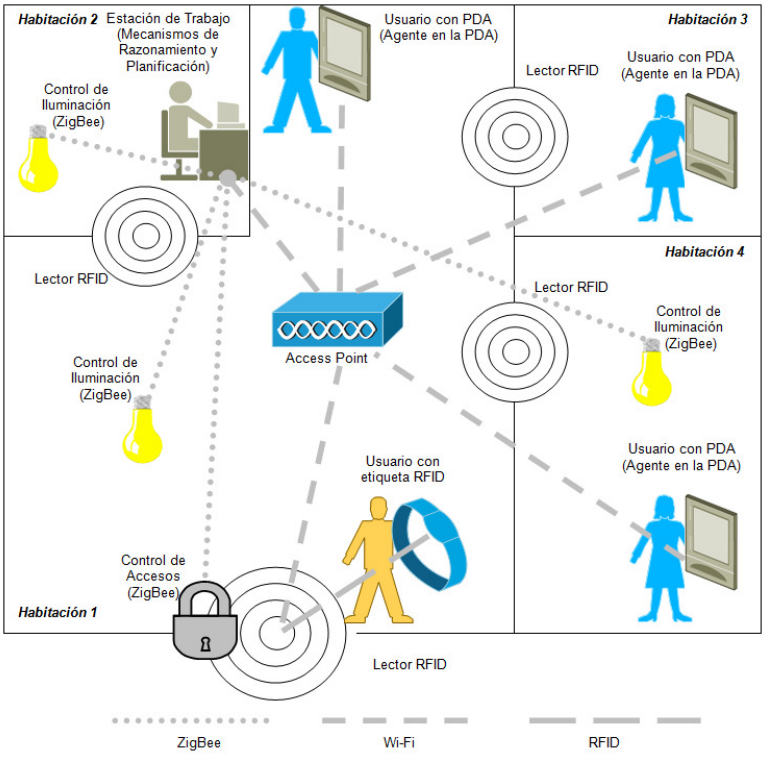

Figura 5.1. Esquema general de las tecnologías utilizadas en ALZ-MAS

Una de las principales características de cara a la interacción con los pacientes es que, salvo por el uso de pulseras RFID, la infraestructura del sistema es prácticamente imperceptible para ellos, pero en todo momento se proporciona un amplio soporte para su atención y detección de posibles irregularidades en su comportamiento médico y hábitos, sin interferir en sus actividades cotidianas. ALZ-MAS monitoriza automáticamente la ubicación de los usuarios a través de tecnología RFID y se notifica cualquier acceso a zonas protegidas o restringidas. Esto permite tener un mayor control y supervisión de los usuarios, ya que en todo momento es posible saber la localización exacta de cada uno de ellos.

A continuación, se define el modelo del contexto de ALZ-MAS, el cual será utilizado en las posteriores fases del análisis y diseño del sistema.

5.2. Modelado del Contexto

Uno de los requisitos más importantes para que un sistema pueda ser clasificado dentro del paradigma de la Inteligencia Ambiental, es que los sistemas deben ser sensibles al contexto, con la capacidad de percibir los estímulos externos procedentes del entorno. Por lo tanto, las acciones de los usuarios deben ser previstas, analizando la situación en la que se producen, de forma similar a como ocurre en el comportamiento humano. Sin embargo, estas situaciones pueden presentar un problema de ambigüedad cuando deben ser modeladas, y es que, ante una determinada situación, existen diferentes respuestas en función del contexto que la envuelve, por lo que es necesario definir adecuadamente dichas funciones.

El "contexto" es cualquier información utilizada para caracterizar la situación de una entidad, sea una persona, un lugar o un objeto (Dey, et al., 2000). Esta información es importante para definir la interacción entre los usuarios y la tecnología que los rodea. El entorno es todo aquello que rodea a los usuarios, mientras que el contexto incluye la información de ambos, que a su vez será lo que dé forma al sistema. Para desarrollar un modelo del contexto es posible emplear técnicas que representen al sistema en la realidad, presentando al contexto como fuente de información.

Debido a la naturaleza de la presente investigación, en especial de la arquitectura FUSION@ y el sistema ALZ-MAS, con la Inteligencia Ambiental como su pilar más importante, es preciso ubicar a los usuarios en el centro del diseño, con el objetivo de lograr un modelo del contexto de acuerdo a las expectativas definidas en los objetivos, repercutiendo en una interacción humano-sistema ubicua, sencilla y transparente para los usuarios.

El modelado del contexto se realiza de forma evolutiva, comenzando por un modelo de bajo nivel al que progresivamente se agrega un mayor nivel de detalle y funcionalidades, obteniendo un modelo más preciso y robusto conforme se avanza en el proceso de análisis y diseño.

El primer paso para realizar un adecuado modelado del contexto, es definir el escenario de aplicación, ya que resultaría una tarea muy compleja diseñar un modelo de un contexto global para cualquier escenario posible. El sistema ALZ-MAS está enfocado a mejorar diversos servicios médicos y de asistencia social para personas con algún grado de dependencia, especialmente a personas con la enfermedad de Alzheimer. Su campo de acción abarca un amplio espectro de escenarios, desde hogares hasta residencias geriátricas, hospitales y centros de día. Por tal motivo es necesario definir un modelo del contexto que sea flexible y escalable, que tenga en cuenta las características de cada escenario y describa claramente los distintos elementos que lo componen y las relaciones entre ellos.

Existen básicamente dos ámbitos de aplicación para este sistema: el ámbito doméstico y el ámbito institucional, englobando este último a los centros de día, residencias geriátricas, hospitales, o cualquier institución proveedora de cuidados individualizados a personas dependientes. A continuación, se describen de forma básica dos escenarios generalizados para cada uno de estos ámbitos:

- **Institucional.** *Una persona dependiente se encuentra en una residencia, hospital o centro de día en el que trabajan médicos, enfermeros, personal no sanitario y administrativos, de forma que la atención esté enfocada a proporcionarle los mejores cuidados. Estos trabajadores monitorizan una serie de datos relacionados con la persona dependiente (constantes vitales, ubicación, actividades, medicación, etc.) y datos del entorno en el que se encuentra (temperatura de la habitación, alarmas, posición de la cama, etc.) para mejorar su calidad de vida, a través de una serie de servicios (planificación de tareas, medicación, modificación del entorno, etc.), los cuales son ejecutados por diversos actores (médicos, enfermeros, máquinas, la propia persona dependiente, etc.).*
- **Doméstico.** *Una persona dependiente se encuentra en su hogar y recibe atención de un cuidador, el cual puede ser personal especializado o una persona cercana a su entorno (familiares, amigos, etc.) sin formación profesional para ejercer labores de asistencia a personas dependientes. Los cuidadores monitorizan constantemente a las personas dependientes (constantes vitales, ubicación, medicación, actividades, etc.), así como el entorno (temperatura del hogar, estado de los electrodomésticos, etc.), con el objetivo de mejorar la calidad de vida de la persona que recibe los cuidados. Para esto, los cuidadores pueden recibir ayuda de especialistas (médicos, enfermeros, auxiliares). Los servicios ofrecidos por el sistema son ejecutados por diversos actores (cuidadores, médicos, auxiliares, e incluso las mismas personas dependientes).*

Los dos escenarios descritos anteriormente tienen elementos en común, siendo el más importante la atención individualizada a personas dependientes,

por lo que todos los elementos del modelo serán diseñados en torno a estas personas. Otros elementos compartidos son algunos de los actores y servicios dentro del sistema, por lo que pueden modelarse de forma conjunta, teniendo en cuenta que puedan ser especializados según las necesidades específicas de cada escenario de aplicación. Ambos escenarios serán el punto de partida para definir el modelo del contexto de ALZ-MAS. El proceso seguido para obtener este modelo se detalla a continuación.

El diseño de un modelo genérico del contexto, lo suficientemente abstracto para abarcar los escenarios descritos en la sección anterior, permitirá añadir gradualmente nuevos componentes de cara a lograr un modelo especializado con mayor flexibilidad para adaptarse a las necesidades y características de cada uno de dichos escenarios. Para ello, es necesario determinar quiénes y qué componen el contexto, información que dará origen al desarrollo del sistema.

Siguiendo esta perspectiva, la información del contexto debe incluir, entre otras cosas, localización específica, capacidades de los servicios ofrecidos, actividades, tareas, roles, situaciones, deseos, creencias, intenciones, etc. De igual forma, debe conocerse en profundidad el medio físico (por ejemplo, la luminosidad o nivel de ruido de un recinto) ya que será la capa de contacto del entorno con el usuario.

La utilización de teorías como la de las cinco W (*Who, Where, When, What, Why*) (Brooks, 2003), la cual recoge las preguntas más importantes que deben responderse para conseguir un modelado eficaz, resultan de gran ayuda para conocer mejor el contexto y desarrollar un modelo más detallado. A continuación, se detallan las cinco preguntas que conforman esta teoría:

- **¿Quién es? (*Who*).** Debe conocerse quiénes están presentes en el entorno que se desea modelar, ya que en todo momento los elementos identificados podrán influir los unos en los otros de forma prevista o imprevista. Implica la identificación de los usuarios y de aquellos elementos (humanos o no humanos) que lo rodean. Identificando a los

usuarios, es posible conocer sus perfiles y todo aquello que pueda influirles en un momento determinado.

- **¿Dónde está? (*Where*).** La ubicación de los elementos que forman parte del modelado es muy importante en un entorno de computación ubicua, involucrando la localización tanto de los usuarios como de los recursos. Según el lugar en el que se encuentren las necesidades del usuario pueden cambiar, por lo que el entorno deberá adaptarse a sus necesidades y características, para poder ofrecer adecuadamente los recursos disponibles.
- **¿Qué está haciendo? (*What*).** Deben conocerse las actividades de quienes están en el entorno. Un control de las actividades de los usuarios ayuda a definir y personalizar los servicios que interactúan con ellos.
- **¿Por qué? (*Why*).** El sistema debe estar preparado para soportar actividades que sigan un patrón específico, así como para los imprevistos que son generados por un fallo del sistema o por el carácter imprevisible del comportamiento humano.
- **¿Cuándo? (*When*).** Además de la ubicación física de los elementos, es importante conocer su ubicación temporal. El sistema debe estar preparado para realizar cualquier actividad en cualquier momento. Sin embargo, las actividades podrán depender de quienes estén en una ubicación cada momento.

Además de responder a las preguntas planteadas por la teoría de las cinco W, otra pregunta que se ha considerado relevante de cara a modelar el contexto es:

- **¿Cómo está?** La medición de parámetros tanto del usuario como del entorno facilita conocer su estado para actuar en consecuencia ante una situación determinada.

Subiendo al máximo el nivel de abstracción, con la intención de definir el funcionamiento básico del sistema, se han definido cinco entidades (sistema, entorno, usuario, personas, y mecanismos) y sus interacciones fundamentales (estimulación y percepción) comunes a los escenarios estudiados. El usuario está

rodeado por el entorno y éste a su vez por el sistema. El estado del usuario y del entorno es percibido por personas o mecanismos, quienes a su vez reaccionan y los estimulan conforme a una situación determinada.

La descripción del modelo del contexto requiere la identificación de las entidades participantes. Es importante señalar que el enfoque del sistema ALZ-MAS sitúa al usuario como el centro del diseño, por lo que el resto de elementos actúan en relación a él. Las principales entidades en este modelo son las siguientes:

- **Usuario.** Son todas las personas que hacen uso de los servicios proporcionados por el sistema o por alguno de los actores. Será la entidad principal, y en base a ésta actuarán el resto de entidades. Dentro del ámbito del sistema ALZ-MAS, los principales perfiles de usuario son:
    - **Persona dependiente.** Usuario con algún grado de dependencia, y que requiere atención personalizada, ya sea dentro de un centro de día, hospital, residencia, o su hogar. Serán los mayores beneficiarios de los servicios, pues éstos están enfocados a mejorar su calidad de vida.
    - **Enfermero.** Usuarios dedicados al cuidado de personas dependientes, y que cuentan con formación profesional para ejercer dichos cuidados. Podrán hacer uso de una serie de servicios y dispositivos para llevar a cabo sus tareas y actividades para mejorar la calidad de vida de las personas dependientes.
    - **Cuidador.** Usuarios cercanos al entorno de la persona dependiente y que además cuidan de ella, pero que sin embargo no tienen formación profesional para ejercer labores de asistencia a personas dependientes. Este tipo de usuario es uno de los más comunes dentro de la problemática de la dependencia en España, por lo que serán una pieza clave para el desarrollo de los servicios en el sistema.

  - **Médico.** Profesional de la medicina encargado de monitorizar el estado de salud de las personas dependientes, con el nivel más alto de conocimientos en la materia.
  - **Auxiliar.** Usuarios que no están totalmente vinculados al cuidado de personas dependientes, pero que apoyan puntualmente en tareas y actividades específicas.
  - **Visitante.** Usuarios que entran en el área de influencia del sistema de forma temporal y que no están relacionados con los cuidados a las personas dependientes.
  - **Administrador.** Usuarios con la capacidad para supervisar el adecuado funcionamiento del sistema.

- **Actor.** Son aquellos elementos, ya sean personas o mecanismos, que son capaces de ejecutar servicios, y a su vez modificar el contexto. Todos los usuarios descritos anteriormente pueden en un momento determinado cambiar de rol y ser actores, dependiendo de la situación y su implicación en la interacción con el contexto. La diferencia más significativa entre actor y usuario es que los actores son capaces de modificar el contexto, mientras que los usuarios solo hacen uso de los servicios. Otra característica importante es que los usuarios solo pueden ser humanos, mientras que los actores pueden ser también mecanismos. Por lo tanto, los principales actores en el sistema son los siguientes:
  - **Persona dependiente.** Estos actores hacen uso de gran cantidad de servicios para interactuar con el sistema de la forma más sencilla, natural e intuitiva posible, sin que se afecte su calidad de vida.
  - **Enfermero.** Podrán hacer uso de una serie de servicios y dispositivos para proporcionar información al sistema, con el fin de almacenarla, gestionarla y analizarla, y así modificar el comportamiento del sistema.

  - **Cuidador.** Será uno de los actores con mayor influencia en el comportamiento del sistema, debido a su constante participación en el cuidado de las personas dependientes.
  - **Médico.** Son el usuario con la mayor jerarquía en el sistema para influir en la atención a las personas dependientes y en las actividades del resto de actores y usuarios. Podrán diagnosticar situaciones y administrar tratamientos a través del sistema.
  - **Auxiliar.** Tendrán una interacción esporádica en el sistema, debido a que solamente se involucran en tareas y actividades puntuales.
  - **Visitante.** Estos actores tendrán una interacción temporal con el sistema, limitándose a realizar acciones simples, con escasa capacidad para modificar el contexto de forma directa, sino a través de otros servicios o actores.
  - **Administrador.** Actores con la capacidad gestionar y afectar los diversos servicios y modificar el funcionamiento global del sistema.
  - **Dispositivos.** Todos aquellos mecanismos que sean capaces de modificar el contexto, así como de proporcionar o recabar información sobre el contexto, hacia y desde los usuarios o actores, a través de la ejecución de servicios.
- **Servicio.** Toda actividad que el sistema es capaz de ofrecer a los usuarios o a sí mismo para satisfacer las necesidades de dichos usuarios o de otros servicios. Los principales servicios identificados en el sistema son:
  - **Monitorización.** Para lograr una adecuada percepción del contexto, es necesario distinguir claramente la procedencia de la información, por lo que es preciso realizar una monitorización tanto del entorno como del usuario. Este servicio está íntimamente ligado a la entidad Sensor, encargada

de obtener información del contexto, aunque la monitorización también puede ser realizada por personas.

- **Monitorización del usuario.** Obtiene información sobre el estado de los usuarios.
  - La identificación de los usuarios es una pieza clave para una personalización de servicios y contenidos.
  - La localización de los usuarios es otro aspecto importante de cara a proporcionar una mejor interacción con el entorno.
- **Monitorización del entorno.** Realiza el seguimiento del estado del entorno.
  - La identificación y localización de los recursos son necesarias para la interacción de los usuarios y su entorno.

- **Gestión de Información.** Homogeniza la información del contexto, la procesa y la almacena.
- **Toma de decisiones.** Procesos automáticos para asignar servicios y actores de acuerdo a una situación específica y para un momento determinado.

Una vez definidas las principales entidades, es posible describir de forma más detallada el ciclo de vida del sistema. Como se muestra en la **Figura 5.2**, el ciclo comienza por obtener información del usuario y su entorno a través del servicio de monitorización, obteniendo dicha información a través de sensores o personas. La información es analizada y procesada por un gestor de información que le dará consistencia a los datos y los almacenará. Una vez procesada la información, se definen y personalizan las actividades a efectuar y quiénes las deberán ejecutar. Las decisiones se envían a los actores, ya sean personas o dispositivos, para que se apliquen las acciones correspondientes y así estimular ya sea al usuario o a su entorno. El usuario puede cambiar de rol durante el

proceso y así desencadenar eventos que influyan en el contexto. Este ciclo se repite con cada nueva interacción con el contexto.

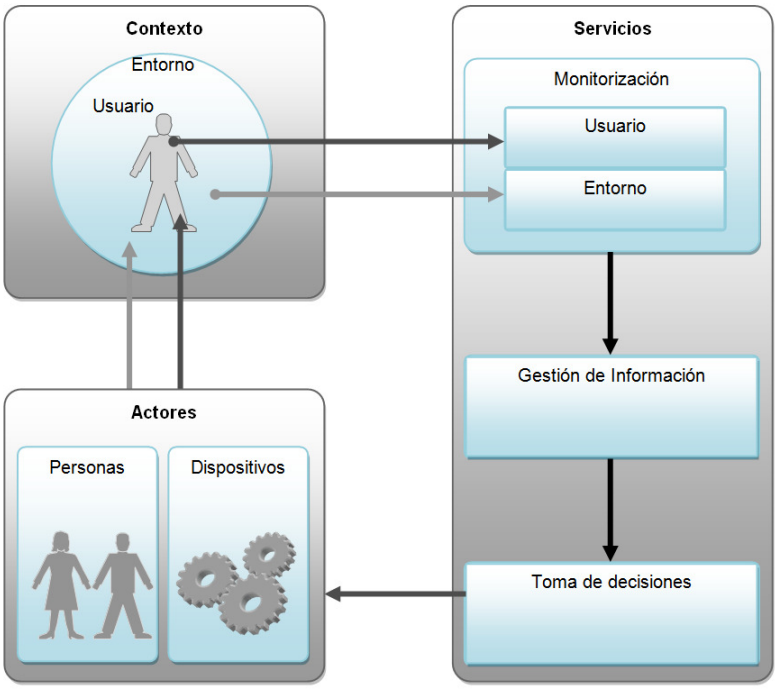

Figura 5.2. Ciclo de vida del sistema

Una vez definido el contexto y el esquema básico del sistema, se procede a realizar el análisis y diseño haciendo uso de la metodología Gaia y el lenguaje de modelado SysML. Durante este proceso se deberá tener en cuenta que la utilización del sistema deberá ser sencilla y lo más ubicua posible, previendo que los usuarios finales no tendrán altos niveles de especialización en el uso de tecnologías informáticas.

5.3. Análisis y Diseño

Una vez modelado el contexto y definidas las principales funcionalidades de ALZ-MAS, es necesario aplicar herramientas para el análisis y diseño de sistemas orientadas a agentes (AOSE). Primeramente se hace uso de la metodología Gaia para así obtener un modelo sencillo del sistema y de sus principales componentes. Los resultados obtenidos con Gaia son posteriormente adaptados y detallados mediante el lenguaje de modelado SysML, obteniendo un modelo del sistema mucho más cercano a la implementación. Finalmente, se define el modelo arquitectónico del sistema, permitiendo pasar directamente a la fase de desarrollo. A continuación, se describen las distintas fases del análisis y diseño para la obtención del modelo del sistema ALZ-MAS.

5.3.1. Análisis Gaia

En la fase de análisis de Gaia se definen el modelo de roles, a partir de las características y requerimientos del sistema, y el modelo de interacción, mostrando las distintas colaboraciones en el sistema y los protocolos que se han considerado en el modelo de roles. Además, se especifican los protocolos y actividades que debe realizar cada uno de los roles, así como los permisos que tienen sobre la información procesada por el sistema. En el proceso de análisis no se describen de forma detallada los datos que se utilizan como entradas o salidas, por lo que solo se representan mediante diagramas en los que se describen los roles participantes, los datos que se manejan y el resultado de cada

interacción. En esta fase tampoco se describen los servicios, ya que estos se desprenden directamente de los roles y se modelan en fases posteriores del análisis y diseño. A continuación, se describen los modelos obtenidos en la fase de análisis Gaia.

5.3.1.1. Modelo de roles

A partir de los requisitos iniciales descritos anteriormente gracias al modelado del contexto, se identifican los roles que participan en el sistema, utilizando criterios sociales de organización. Para este caso de estudio se han obtenido los siguientes roles (Apéndices D.1 –5):

1) **Rol Paciente.** Gestiona la información médico/sanitaria de los pacientes así como su comportamiento (monitorización, localización, tareas diarias, anomalías, etc.). Desempeña además las tareas de: controlar los datos personales del paciente; llevar un seguimiento de los tratamientos asignados por los médicos; informar periódicamente la evolución del paciente ante los tratamientos administrados; informar a los médicos ante una petición de informes; obtener la ubicación del paciente en un momento determinado; llevar un control y planificación de las tareas que el paciente debe seguir diariamente de acuerdo a su tratamiento; mantener una continua interacción con el rol enfermero; y finalmente, manejar situaciones de riesgo en las que los datos recogidos para el paciente sean anómalos.
2) **Rol Enfermero.** Gestiona la planificación diaria de los enfermeros y actualiza los atributos médicos de los pacientes. Además, es el encargado de aplicar a los pacientes los tratamientos asignados por los médicos. Dado que los enfermeros deben atender a varios pacientes y los tratamientos y tareas asignados a cada paciente pueden variar a lo largo

del día, la lista de tareas que el enfermero debe realizar es dinámica. Esta planificación debe optimizarse para que cada enfermero sea capaz de realizar todas las tareas asignadas. Esto se realiza ajustando las horas a las que se debe realizar cada tarea e intercambiando tareas entre los distintos enfermeros conectados al sistema.

3) **Rol Médico.** Gestiona los tratamientos de los pacientes y organiza la agenda de los médicos. Además, se encarga de asignar tratamientos a los pacientes, así como las tareas que los enfermeros deben realizar para completar dichos tratamientos. Las citas con los pacientes podrán ser asignadas por el propio médico o de forma automática por el sistema. Al igual que con las tareas, la asignación de citas se realiza de forma dinámica.
4) **Rol Seguridad.** Monitoriza la localización de enfermeros y pacientes, y gestiona las situaciones de riesgo detectadas. Este rol obtiene en todo momento la ubicación de los usuarios a través de tecnología RFID. Por otra parte, hace uso de dispositivos ZigBee para activar o desactivar automáticamente diversos servicios físicos, por ejemplo pestillos electrónicos, alarmas, sensores de humo, etc.
5) **Rol Manager.** Administra la información y realiza el reparto de tareas entre los enfermeros. El resto de roles se comunican con este rol para informar los cambios en tratamientos, tareas, citas e información médica para que éste actualice la información. El Rol Manager administra las altas, bajas y modificaciones de usuarios. Controla la conexión y desconexión de los usuarios en el sistema. Además, se encarga de optimizar la planificación de tareas ante cualquier evento que requiera una nueva planificación, como es el caso del número de usuarios en el sistema, o cuando cambian los tratamientos asignados.

5.3.1.2. Modelo de interacción

A continuación, se describen y se muestran en esquemas, las interacciones que se presentan en el sistema. Para cada interacción en la que intervienen dos o más roles, es necesario describir los protocolos utilizados. Los protocolos que se han definido son los siguientes (Apéndices D.6-26):

- **Conexión al sistema de un médico o enfermero.** Los usuarios solicitan conectarse al sistema haciendo uso de su respectivo nombre de usuario y contraseña. Se comprueban los datos y, en caso de ser correctos, el usuario es autorizado y conectado al sistema, ya sea como médico o como enfermero. Si el tipo de usuario es enfermero, se realiza una nueva planificación de tareas y se actualizan todas las tareas de los enfermeros conectados al sistema.
- **Desconexión del sistema de un médico o enfermero.** Cuando un usuario se desconecta del sistema, se actualiza el estado del usuario y se notifica el éxito de la operación. Si el usuario desconectado corresponde a un enfermero, se realiza una nueva planificación de tareas y se actualizan todas las tareas de los enfermeros conectados al sistema.
- **Nuevo tratamiento.** Cuando un médico asigna un nuevo tratamiento a un paciente, el tratamiento se guarda y se informa el estado del proceso. Cada vez que se asigna un tratamiento se realiza una nueva planificación de tareas y se actualizan todas las tareas de los enfermeros conectados al sistema.
- **Modificar tratamiento.** Cuando un médico modifica el tratamiento asignado a un paciente, se guardan los cambios y se informa el estado del proceso. Al modificar un tratamiento, se realiza una nueva planificación de tareas y se actualizan todas las tareas de los enfermeros conectados al sistema.

- **Eliminar tratamiento.** Si un médico elimina un tratamiento asignado a un paciente se almacenan los cambios, se informa el estado del proceso y se realiza una nueva planificación de tareas, actualizando todas las tareas de los enfermeros conectados al sistema.
- **Nueva cita.** Al crear una nueva cita, ya sea por el propio médico o por el sistema, la cita se almacena y se crea un nuevo tratamiento asociado con la cita. Se realiza una nueva planificación de tareas y se actualizan todas las tareas de los enfermeros conectados al sistema.
- **Modificar cita.** Al modificar una cita, los cambios se almacenan y se modifica automáticamente el tratamiento asociado a dicha cita. Se realiza una nueva planificación de tareas y se actualizan todas las tareas de los enfermeros conectados al sistema.
- **Eliminar cita.** Al eliminar una cita, se actualizan las citas y el tratamiento asociado. Se realiza una nueva planificación de tareas y se actualizan todas las tareas de los enfermeros conectados al sistema.
- **Iniciar tarea.** Cada enfermero debe informar el inicio de una tarea y se guarda la información.
- **Finalizar tarea.** Cada enfermero debe informar que ha terminado de realizar la tarea asignada que ha iniciado. El sistema registra el tiempo que ha tardado en realizar dicha tarea y se almacena la información.
- **Rechazar tarea.** Si un enfermero no es capaz de realizar una tarea asignada, se indica al sistema para que se realice una nueva planificación. Se realiza una nueva planificación de tareas y se actualizan todas las tareas de los enfermeros conectados al sistema.
- **Consultar atributos de paciente.** Cuando un enfermero solicita los atributos de un paciente, el sistema atiende la petición y le envía la información requerida.
- **Modificar atributos de paciente.** Al modificar los atributos del paciente, el enfermero envía los nuevos valores y se envía un aviso de confirmación al guardar la información.

- **Solicitar cita.** Si los atributos del paciente no están dentro de los rangos normales, el sistema asigna una nueva cita a uno de los médicos. Se realiza una nueva planificación de tareas y se actualizan todas las tareas de los enfermeros conectados al sistema.
- **Consultar localización de paciente.** Al solicitar la ubicación de un determinado usuario, el sistema genera un mapa con la localización exacta de dicho usuario y lo comunica al usuario que ha realizado la petición.
- **Enviar alertas.** Cuando un usuario accede a una zona restringida, se envía una alerta, es decir, una tarea con prioridad máxima, al enfermero disponible más cercano y con menor carga de trabajo.
- **Obtener lista de pacientes.** Cuando un usuario solicita el listado de los pacientes, el sistema envía automáticamente dicha información.
- **Enviar lista de atributos de paciente.** Se actualizan los atributos de un paciente al añadir, eliminar o modificar dichos atributos.
- **Enviar lista de pacientes.** Se actualiza la información de los pacientes registrados en el sistema al añadir, eliminar o modificar un registro.
- **Enviar lista de citas.** Al generarse una nueva cita, el sistema envía la lista de citas a los respectivos médicos conectados en el sistema. Si se da de baja a un médico sus citas se reparten entre el resto de médicos. Si se elimina a un paciente, sus citas también se eliminan. Al haber un cambio en la lista de citas, también lo hay en los tratamientos asociados a dichas citas y en el resto de tratamientos del paciente, por lo que también se eliminan. Con cada cambio en las citas, se realiza una nueva planificación de tareas y se actualizan todas las tareas de los enfermeros y médicos conectados al sistema.

5.3.2. Diseño de Alto Nivel Gaia

Una vez concluida la fase de análisis, se procede a desarrollar la fase de diseño de alto nivel, haciendo uso nuevamente de los conceptos definidos en Gaia, en donde se obtiene: el modelo de agentes, identificando tipos de agentes y las instancias que aparecerán en tiempo de ejecución; el modelo de servicios, definiendo los principales servicios que puede realizar cada agente a partir de las actividades, protocolos y propiedades (*liveness* y *safety*) de los roles; y el modelo de conocidos, definiendo la comunicación entre los agentes a partir del modelo de agentes y del modelo de interacción. Una vez más, los servicios no son definidos hasta en posteriores etapas del análisis y diseño.

5.3.2.1. Modelo de agentes

Como se menciona en el capítulo 4 de esta memoria, FUSION@ es una arquitectura totalmente abierta, en la cual los agentes no están definidos de una manera estática. Por lo tanto, es posible agregar agentes a la plataforma para así lograr un modelo que se ajuste a las necesidades de cada proyecto, en este caso del sistema ALZ-MAS. De esta forma, en el modelo de roles se han obtenido cinco roles distintos: Paciente, Enfermero, Médico, Seguridad y Manager. Analizando estos roles y sus interacciones, así como la capacidad que tienen los agentes para jugar varios roles, se han definido cuatro tipos de agentes tal y como se puede apreciar en la **Figura 5.3**:

- ***Patient Agent.*** Ejecuta el Rol Paciente, gestionando los datos médicos de los pacientes. Asimismo, realiza las tareas de supervisión, localización,

tareas diarias, y situaciones de riesgo. Periódicamente valida los datos de los pacientes y los informa al Agente Manager para que los almacene. Se utilizan creencias y metas personalizadas para cada paciente, dependiendo del tratamiento que los médicos prescriban. Este agente supervisa el estado paciente a través de metas, por lo que para saber si una meta se ha alcanzado o no, es necesario mantener una comunicación continua con el resto de los agentes. El Agente Paciente debe asegurarse de que todas las acciones indicadas en los tratamientos son realizadas. Existe una instancia de *Patient Agent* por cada paciente registrado en el sistema.

- ***Mobile Agent.*** Este agente ejecuta los roles Médico y Enfermero. Al arrancar el agente, su funcionalidad será la de conexión en el sistema. Cuando el usuario introduce sus datos de conexión (nombre de usuario y contraseña), el agente se comunica con el Agente Manager para comprobar si los datos son correctos y el tipo de usuario que se ha conectado. Dependiendo del tipo de usuario conectado, asume el Rol Médico o parte de las funcionalidades del Rol Enfermero hasta que el usuario se desconecte del sistema. Asumiendo el Rol Médico, asigna tratamientos a los pacientes y gestiona las citas de los médicos. Cuando asume el Rol Enfermero, gestiona la planificación de los enfermeros. Este agente está diseñado para ser ejecutado en dispositivos móviles, como es el caso de los asistentes personales (PDAs). Este tipo de agente corresponde a un *Interface Agent* de FUSION@, ya que se ejecuta en dispositivos móviles y gran parte de las tareas de procesamiento se delegan a la arquitectura de agentes y a los servicios.
- ***Nurse Agent.*** Este agente ejecuta las funcionalidades complementarias del Rol Enfermero y que no son asumidas por el *Mobile Agent*. El *Nurse Agent* planifica de forma dinámica las tareas de los enfermeros. Además, gestiona los perfiles, las tareas, el tiempo disponible y los recursos de los enfermeros. Los planes generados deben garantizar que todos los pacientes asignados a cada enfermero están siendo atendidos. Existe una

instancia de *Nurse Agent* asociada a cada instancia de *Mobile Agent*. Cada vez que un *Mobile Agent* asuma el Rol Enfermero se creará el correspondiente *Nurse Agent*, destruyéndose cuando el *Mobile Agent* asociado se desconecte o cambie de rol. Este hecho se debe a que la poca capacidad de procesamiento y autonomía de los dispositivos móviles hacen difícil la ejecución de los mecanismos necesarios para realizar la planificación de tareas para los enfermeros. Este agente recoge los resultados de la planificación global realizada por el *Manager Agent* y los procesa para asignar cada tarea de forma eficiente. Los resultados obtenidos en cada nueva planificación son enviados al respectivo *Mobile Agent*.

- ***Manager Agent***. Ejecuta los Roles Manager y Seguridad. Haciendo uso de la infraestructura del sistema (RFID, ZigBee, Wi-Fi, etc.), obtiene periódicamente información que le permite reaccionar físicamente sobre el entorno (por ejemplo, activando alarmas). Además, calcula en todo momento la ubicación de los usuarios, informando en dónde se encuentra cada uno de ellos. Asimismo, gestiona la información de los usuarios. Existe solo un *Manager Agent* en el sistema, manteniendo una comunicación constante con todo el resto de los agentes. Actúa en paralelo con el *Nurse Agent* para realizar la planificación de tareas, asignando de forma global todas las tareas pendientes a los enfermeros disponibles.

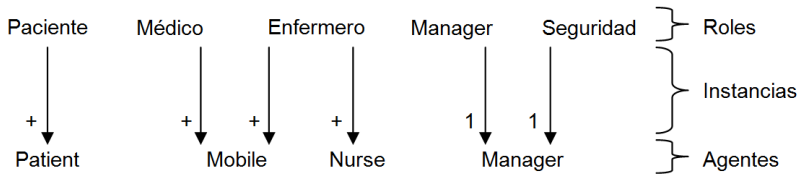

Figura 5.3. Modelo de Agentes ALZ-MAS

5.3.2.2. Modelo de Conocidos

El modelo de conocidos define los enlaces de comunicación que existen entre los agentes del sistema. Tiene como principal objetivo buscar posibles problemas de comunicación en tiempo de ejecución. Solo se muestran los enlaces, sin entrar en detalle sobre las características de la comunicación (mensajes, formatos, etc.).

El modelo general definido para ALZ-MAS se muestra en la **Figura 5.4**. Los agentes de FUSION@ como los propios de ALZ-MAS se comunican entre sí para aportar todas las funcionalidades del sistema.

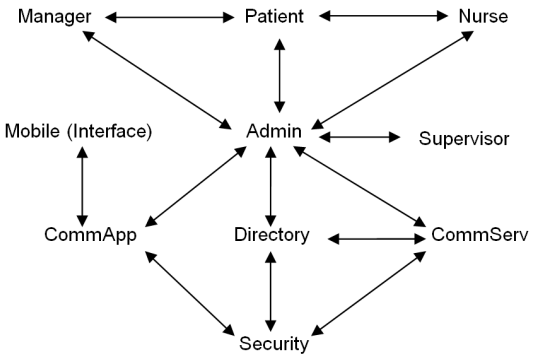

Figura 5.4. Modelo de conocidos ALZ-MAS

5.3.3. Diseño de Bajo Nivel SysML

Una vez finalizado el análisis y diseño de alto nivel mediante le metodología Gaia, se procede a analizar los resultados y realizar un diseño de bajo nivel haciendo uso del lenguaje de modelado SysML. En esta etapa, se obtienen los diagramas de

Definición de Bloques, Secuencia, Interacción y Estados, los cuales se describen a continuación.

5.3.3.1. Diagramas de Definición de Bloques

Los diagramas de definición de bloques representan los agentes del sistema y cada uno de sus roles, definiendo las capacidades y servicios que deben realizar. A diferencia de los modelos Gaia, estos diagramas permiten definir la estructura interna de cada uno de los agentes. Debido a que las características de los agentes de FUSION@ ya han sido descritas en el capítulo 4 de esta memoria, tan solo se definen los agentes propios de ALZ-MAS.

- El *Patient Agent* (Apéndice E.1) tiene las capacidades para solicitar nuevas citas con un médico y de guardar los valores de sus atributos. Además, ofrece una serie de servicios, entre los que se encuentran el mostrar de forma dinámica la localización en un mapa determinado, actualizar e informar la lista de atributos, así como actualizar sus valores cuando un *Mobile Agent* lo solicita.
- El *Mobile Agent* (Apéndice E.2) tiene una serie de capacidades que dependen del tipo de rol que adquiera el agente (médico o enfermero) en un momento determinado. Para ambos roles cuenta con las capacidades de conectarse y desconectarse del sistema, así como la localización de pacientes. Cuando asume el Rol Enfermero, gestiona el servicio de alertas, comunicándose con el *Manager Agent* para enviar alertas personalizadas a los enfermeros. Además, muestra la jornada laboral de los enfermeros y actualiza de forma periódica los atributos de los pacientes. Asumiendo el Rol Médico, gestiona las citas y los tratamientos asignados a los pacientes.

- El *Nurse Agent* (Apéndice E.3) realiza una organización automática y dinámica de tareas. Cuando recibe del *Manager Agent* o un *Mobile Agent* una nueva lista de tareas, la organiza para un cumplimiento óptimo de las mismas y se las envía al correspondiente *Mobile Agent*.
- El *Manager Agent* (Apéndice E.4) gestiona de forma global a todos los usuarios del sistema. Sus funcionalidades incluyen todas aquellas que el resto de los agentes necesitan para poder desempeñar sus capacidades, es decir, servicios de conexión y desconexión, gestión de citas, tratamientos y planes de tareas, localización, alarmas, etc.

5.3.3.2. Diagramas de Secuencia SysML

Los diagramas de secuencia (Apéndices E.5-14) representan las interacciones entre los agentes del sistema, concretamente el intercambio de mensajes. En particular, se detalla la secuencia y contenido de la transmisión de mensajes, mostrando las interacciones entre los agentes y los roles que éstos toman durante dichas interacciones.

- **Conexión.** El usuario introduce sus datos de registro (nombre de usuario y contraseña) a través del *Mobile Agent*. Éste envía los datos al *Manager Agent* para comprobar si son correctos, informando en todo caso al *Mobile Agent*. Dependiendo de los datos introducidos, el *Mobile Agent* asume el Rol Médico o el Rol Enfermero, activando las funcionalidades propias de cada rol. Si el *Mobile Agent* asume el Rol Enfermero, el *Manager Agent* realiza una planificación de tareas y envía el resultado a todos los *Nurse Agent*.
- **Desconexión.** Cuando el médico o enfermero se desconecta del sistema, el *Mobile Agent* informa al *Manager Agent,* quien a su vez confirma la operación. El *Mobile Agent* cambia de rol y se pone en espera para que

un nuevo usuario se conecte. Si el usuario desconectado es un enfermero, se realiza una nueva planificación de tareas.

- **Consulta de Tratamientos.** El médico selecciona a uno de los pacientes a través de su *Mobile Agent*, el cual a su vez solicita al *Manager Agent* la lista de los tratamientos asignados a dicho paciente. La información de los tratamientos es enviada al *Mobile Agent* del médico.
- **Nuevo Tratamiento.** El médico asigna un nuevo tratamiento a un paciente rellenando el formulario de alta de tratamiento a través de su *Mobile Agent*. Los datos se envían al *Manager Agent*, quien los almacena e informa al *Mobile Agent* del médico del resultado de la operación. A continuación, el *Manager Agent* realiza una nueva planificación de tareas y la envía a los enfermeros conectados al sistema.
- **Actualización de Tratamiento.** El médico selecciona un tratamiento de la lista de tratamientos disponible y modifica los datos del tratamiento a través del *Mobile Agent*. Éste informa de los cambios al *Manager Agent*, quien a su vez los almacena e informa el resultado de la operación. A continuación, el *Manager Agent* realiza una nueva planificación de tareas y la envía a los enfermeros conectados al sistema.
- **Borrado de Tratamiento.** El médico selecciona un tratamiento de la lista de tratamientos disponibles y lo elimina a través del *Mobile Agent*. Éste informa al *Manager Agent* que a su vez elimina el tratamiento e informa el resultado de la operación. A continuación, el *Manager Agent* realiza una nueva planificación de tareas y la envía a los enfermeros conectados al sistema.
- **Selección de Citas.** El médico introduce una fecha a través de su *Mobile Agent*, quien la envía al *Manager Agent* para que éste le devuelva la lista de citas del médico en la fecha correspondiente.
- **Nueva Cita.** El médico selecciona un paciente e introduce los datos de una nueva cita a través de su *Mobile Agent*. Éste envía los datos al *Manager Agent*, quien la almacena e informa el resultado de la operación. A su vez, el *Manager Agent* envía la lista de citas actualizada y

crea un tratamiento. Se realiza una nueva planificación de tareas y se envía a los enfermeros conectados al sistema.

- **Actualización de Cita.** El médico selecciona una cita de su lista de citas diarias o de una búsqueda por fecha de citas y modifica sus datos a través de su *Mobile Agent*. Éste envía los nuevos datos al *Manager Agent* quien almacena los cambios e informa el resultado de la operación. Se actualiza la cita y el respectivo tratamiento. A continuación, el *Manager Agent* realiza una nueva planificación de tareas y la envía a los enfermeros conectados al sistema.
- **Borrado de Cita.** El médico selecciona una cita de su lista de citas diarias o de una búsqueda por fecha de citas y la elimina a través de su *Mobile Agent*. Éste envía la petición al *Manager Agent* quien elimina la cita de e informa el resultado de la operación. Se actualiza la lista de citas diarias y el respectivo tratamiento. A continuación, el *Manager Agent* realiza una nueva planificación de tareas y la envía a todos los enfermeros conectados al sistema.

5.3.3.3. Diagramas de Interacción

Los Diagramas de Interacción (Apéndices E.15-26), al igual que los diagramas de secuencia, representan interacciones entre agentes, pero hacen énfasis en las asociaciones entre ellos. Es decir, representan la relación que existe entre cada agente y el resto de los agentes del sistema.

- **Planificación.** El *Manager Agent* realiza una asignación global de tareas y la envía a cada uno de los *Nurse Agent* sus tareas asignadas. Cada *Nurse Agent* recoge dichas tareas, las organiza y finalmente las envía al *Mobile Agent* correspondiente. Para realizar una planificación de tareas óptima, el *Manager Agent* recopila todas las tareas que no estén iniciadas aún de

las ocho horas siguientes a la actual y de las dos horas anteriores. Se estima el tiempo necesario para realizar cada tarea, tomando en cuenta el tiempo promedio empleado en realizar la misma tarea en ocasiones anteriores. Se asignan las tareas a los enfermeros teniendo en cuenta el perfil requerido por dichas tareas y el perfil de cada uno de los enfermeros, la prioridad de las tareas, la hora a la que deben realizarse y la duración de las mismas. Además, se verifica si las tareas han sido rechazadas previamente por algún enfermero. Una vez realizada la asignación global de tareas, el *Nurse Agent* organiza las tareas tomando en cuenta la prioridad de cada una de las tareas, la hora a la que deben realizarse y la duración de las mismas. En esta segunda organización, el *Nurse Agent* adelanta o retrasa la hora de inicio de las tareas basándose en su prioridad y duración.

- **Localización de Usuario.** El *Manager Agent* actualiza la ubicación de los usuarios cuando éstos cambian de habitación (al ser identificados por otro lector RFID en una zona distinta). Para visualizar la ubicación de un determinado usuario, el *Mobile Agent* solicita al *Manager Agent* dicha información, quien a su vez pide al *Patient Agent* que dibuje un mapa con la ubicación exacta del usuario. A continuación, se envía el mapa al agente *Mobile Agent* correspondiente.
- **Información de Paciente.** Un enfermero selecciona un paciente a través de su *Mobile Agent*. Éste realiza una petición al *Patient Agent* para obtener la información médica del paciente seleccionado. La información es enviada al *Mobile Agent*. Si el enfermero actualiza la información médica del paciente, la petición se envía al *Manager Agent*, quien actualiza los datos e informa el resultado de la operación. Si los valores introducidos presentan algún conflicto, se genera automáticamente una cita con un médico. A continuación, el *Manager Agent* realiza una nueva planificación de tareas y la envía a los enfermeros conectados al sistema.

- **Inicio y Finalización de Tareas.** Un enfermero selecciona la tarea que va a comenzar. El *Mobile Agent* marca la tarea como iniciada, guarda la fecha y hora de comienzo de la tarea e informa al *Manager Agent* del inicio de la tarea. Si una tarea está iniciada, el enfermero puede seleccionarla para finalizarla. El *Mobile Agent* calcula el tiempo que ha tardado en realizarse la tarea, la marca como finalizada e informa al *Manager Agent* para que la información sea almacenada. Además, el *Mobile Agent* informa al *Nurse Agent* para que realice una nueva planificación de tareas, pero sólo del enfermero en particular.
- **Rechazo de Tareas.** El enfermero selecciona una tarea que no haya iniciado ni finalizado y la rechaza a través de su *Mobile Agent*. Éste envía la petición de rechazo al *Manager Agent*, que realiza una nueva planificación de tareas y la envía a todos los enfermeros conectados al sistema.
- **Borrado de Médico.** El administrador del sistema es capaz de eliminar usuarios, en este caso, un médico. Si existe al menos otro médico disponible, el *Manager Agent* lo elimina. Si en ese momento estaba conectado, le envía una orden de desconexión al correspondiente *Mobile Agent* y se reparten sus citas con el resto de médicos del sistema. Se actualizan los tratamientos asociados a dichas citas, se realiza una nueva planificación de tareas y se envía a todos los enfermeros conectados al sistema**.**
- **Borrado de Enfermero.** Al eliminar un enfermero, se verifica si existe al menos otro enfermero y el *Manager Agent* lo elimina. Si en ese momento estaba conectado, le envía una orden de desconexión al correspondiente *Mobile Agent*. Se realiza una nueva planificación de tareas y se envía a todos los enfermeros conectados al sistema.
- **Alta de Paciente.** El administrador del sistema introduce los datos de un nuevo paciente. El *Manager Agent* comprueba que no exista, almacena la información e informa al administrador el resultado de la operación. Se crea automáticamente un nuevo *Patient Agent* que

gestionará la información médica del paciente. Se envía la lista actualizada de pacientes a todos los médicos y enfermeros conectados al sistema.

- **Borrado de Paciente.** El *Manager Agent* elimina el registro y el correspondiente *Patient Agent*. Se envía la lista actualizada de pacientes a todos los médicos y enfermeros. Además, se eliminan todas las citas y tratamientos no finalizados del paciente. Se realiza una nueva planificación de tareas y se envía a todos los enfermeros conectados al sistema.
- **Alta de Atributo de Paciente.** El administrador selecciona un paciente e introduce los datos de un nuevo atributo médico para dicho paciente. El *Manager Agent* almacena la información y envía la lista actualizada de atributos al correspondiente *Patient Agent*.
- **Actualización de Atributo de Paciente.** Tras seleccionar un atributo de un paciente, el administrador modifica sus datos. El *Manager Agent* almacena los cambios y envía la lista actualizada de atributos al correspondiente *Patient Agent*.
- **Borrado de Atributo de Paciente.** Tras seleccionar un atributo de un paciente, el administrador lo elimina a través del *Manager Agent* y se actualiza la información. Se envía la lista actualizada de atributos al correspondiente *Patient Agent*.

5.3.3.4. Diagramas de Estados

Los diagramas de estados (B.27 – B.30) muestran el comportamiento interno de cada agente, definiendo los estados por los que pasan los agentes y los eventos que generan transiciones de un estado a otro en tiempo de ejecución.

- ***Patient Agent.*** Cuando se crea un nuevo paciente, se crea su agente asociado que estará en el estado de *espera*. Ante los distintos eventos que se produzcan, pasará a uno de los otros estados y cuando los termine, volverá al estado de *espera*. Se pasa del estado de *espera* al estado de *finalización* del agente cuando el paciente es eliminado y el agente recibe por parte del *Manager* la orden de terminar. El resto de los estados del agente son los siguientes:
    - *Actualizando Localización.* Cuando el *Patient Agent* recibe su nueva localización por parte del *Manager Agent* (asumiendo el Rol Seguridad), pasa del estado de *espera* a este estado, actualiza su localización y vuelve al estado anterior.
    - *Dibujando Localización.* cuando el agente recibe una solicitud de localización, pasa a este estado, dibuja el mapa con la ubicación exacta de los usuarios. El mapa es enviado al *Mobile Agent* que lo solicitó y vuelve al estado de *espera*.
    - *Actualizando Atributos.* Cuando el *Patient Agent* recibe una nueva lista de atributos por parte del *Manager Agent* o nuevos valores para sus atributos desde el *Mobile Agent*, pasa a este estado, actualiza sus atributos y vuelve al estado de *espera*.
    - *Entregando Atributos.* Cuando el *Patient Agent* recibe una solicitud de atributos por parte de un *Mobile Agent*, pasa a este estado, le envía sus atributos médicos al *Mobile Agent* que lo solicitó y vuelve al estado de *espera*.
- ***Mobile Agent.*** Este agente tiene dos estados principales, que son el de *registro* y *espera*, y otros dos para procesar las solicitudes que lleguen de otros agentes o del usuario.
    - *Registro.* Cuando se inicia el agente en el dispositivo móvil o en el PC, se pasa al estado de registro. Cuando un usuario intenta conectarse en el sistema, si sus datos son correctos, se pasa al estado de *espera*. Desde el estado de *espera* se pasa al estado de finalización cuando el usuario intenta cerrar el agente.

- *Esperando.* Es el estado de *espera* del agente. Cuando el usuario intenta desconectarse del sistema, se pasa al estado anterior de *registro*. Desde el estado de *espera* se pasa al estado de *finalización* cuando el usuario intenta cerrar el agente. Antes de pasar al estado de *finalización*, se envía una notificación de desconexión al *Manager Agent*.
- *Procesando Solicitud.* Se pasa del estado de *espera* a este estado cuando el usuario intenta gestionar tratamientos o citas si es un médico, o cuando el usuario trata de gestionar su plan de tareas o los atributos médicos de un paciente, si es un enfermero. Ocurre lo mismo cuando un usuario (sea del tipo que sea) pide la localización de un paciente. Tras enviar la solicitud al agente que corresponda, se vuelve al estado de *espera*.
- *Actualizando Interfaz.* Cuando se recibe un mensaje de otro agente, se pasa del estado de *espera* a este estado, se actualiza la interfaz con los nuevos datos recibidos (listas de citas, de tratamientos, de pacientes, mapas de localización, planes de trabajo, atributos médicos, tipos de tareas, tipos de perfiles, etc.), se informa al usuario del evento y se vuelve al estado anterior.

- ***Nurse Agent.*** Cuando se conecta un *Mobile Agent* asumiendo el Rol Enfermero, se crea un agente asociado que estará en estado de *espera*. Cada vez que se recibe una lista de tareas por parte del *Manager Agent* o del *Mobile Agent*, se pasa al estado de *planificación*, en el cual realiza una organización de la lista de tareas para formar un plan de trabajo para la jornada laboral de los enfermeros. El resultado se envía a los *Mobile Agent* y vuelve al estado de *espera*. El agente pasará al estado de *finalización* cuando se desconecte el *Mobile Agent* y el *Manager Agent* se lo comunique al *Nurse Agent*.
- ***Manager Agent.*** Cuando se inicia este agente se pasa al estado *Iniciando*, en el que se comprueba la información, se inicializan los lectores de

radiofrecuencia y la interfaz gráfica, se comprueba cuántos pacientes existen y se arranca un *Patient Agent* por cada uno de los pacientes. A continuación, se pasa al estado compuesto *Manager*, el cual tiene dos partes bien diferenciadas dependiendo del rol que asuma el agente en un momento determinado:

- Asumiendo el Rol Manager, gestiona la información y las peticiones del resto de los agentes en el sistema. En esta situación, el agente siempre está esperando solicitudes de otros agentes o de algún usuario. Cuando recibe algún mensaje de otro agente se pasa al estado *procesando solicitud*, se procesa dicha solicitud y se vuelve al estado anterior. Cuando un usuario intenta gestionar a los médicos, enfermeros o pacientes, se pasa al estado *Administrando*, volviendo al estado de *espera* al terminar la gestión. Mientras el agente está en el estado *Administrando*, sigue procesando las solicitudes de los otros agentes (este estado se puede ejecutar de manera concurrente con el estado *Procesando Solicitud*). Al producirse algún cambio en los tratamientos de los pacientes, se pasa al estado *Planificando*, se reparten las tareas entre los enfermeros conectados y se vuelve al estado de *espera*.
- Asumiendo el Rol Seguridad, monitoriza constantemente los lectores de radiofrecuencia y ejecuta la interfaz gráfica. *Comprobando Sensores* es el estado principal. En este estado, el agente obtiene continuamente información de lectores de radiofrecuencia. Cuando un paciente cambia de zona se pasa al estado *Actualizando Localización*, se actualiza la posición del paciente y se vuelve al estado anterior. Cuando un paciente accede a una zona restringida, (por ejemplo, al salir del centro geriátrico), se pasa al estado de *Alerta*, enviando una notificación a uno de los enfermeros conectados al sistema y se vuelve al estado de *comprobación de sensores*.

5.4. Integración de ALZ-MAS y FUSION@

Tras el análisis y diseño del sistema ALZ-MAS se hace necesario adaptar los resultados para que cumplan con los requerimientos de FUSION@. Con el objetivo de obtener un marco adecuado para comprobar la eficiencia y robustez de esta arquitectura, se han desarrollado en paralelo dos versiones del sistema ALZ-MAS, ambas con las mismas funcionalidades pero una de ellas haciendo uso de FUSION@.

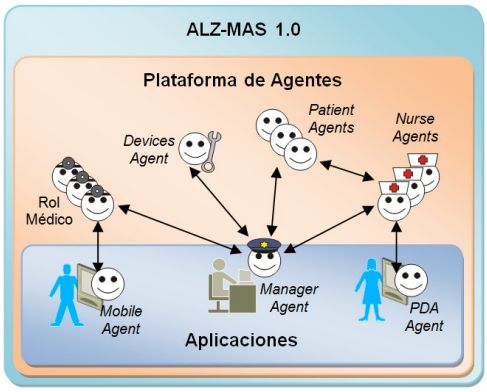

Figura 5.5. Estructura básica de ALZ-MAS 1.0

La **Figura 5.5** muestra la estructura básica de la versión 1.0 de ALZ-MAS, la cual no implementa FUSION@. En esta versión, cada agente integra sus propias funcionalidades dentro de su estructura. De esta forma, cuando un agente necesita ejecutar una tarea que deba procesar otro agente, este se debe comunicar directamente con dicho agente para hacer una petición. Por lo tanto, si un agente se desconecta del sistema, sus funcionalidades no estarán disponibles para el resto de los agentes. Este hecho representa uno de los mayores problemas en ALZ-MAS 1.0, ya que por la propia naturaleza distribuida

de los *Mobile Agent*, estos se desconectan continuamente del sistema, dando como resultado constantes bloqueos y haciendo necesario reiniciar nuevas instancias. Otro problema en esta versión es que es que el mecanismo para la planificación de tareas también está integrado con los agentes. Este mecanismo planificador conlleva una constante entrada de peticiones, lo que sobrecarga en gran medida a los agentes *Manager* y *Nurse*. El mecanismo planificador debe estar siempre disponible para cualquier petición entrante, ya que el sistema depende de este mecanismo para generar las decisiones respecto a la jornada laboral de los enfermeros. Por tal motivo, es recomendable que se disponga de los recursos computacionales necesarios para que se desempeñe de forma adecuada sin afectar al resto de componentes.

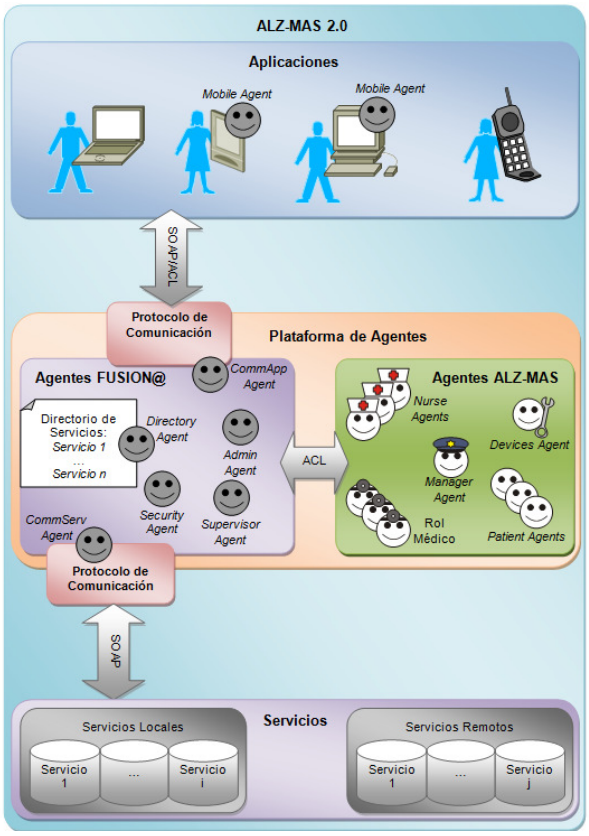

Figura 5.6. Estructura básica de ALZ-MAS 2.0

Como se muestra en la **Figura 5.6**, la estructura de ALZ-MAS 2.0 ha sido totalmente modificada para integrarse con FUSION@. En esta versión, las principales funcionalidades de ALZ-MAS han sido modelando como aplicaciones y servicios independientes, liberando a los agentes de procesos con alta demanda computacional.

Una de las principales funcionalidades de ALZ-MAS es el uso de mecanismos de razonamiento y planificación basados en casos (mecanismos CBR y CBP), los cuales permiten a los agentes hacer uso de experiencias pasadas para crear mejores planes y alcanzar sus metas. Estos mecanismos proveen a los agentes con una mayor capacidad de aprendizaje y adaptación. En ALZ-MAS 1.0, este mecanismo se encuentra fuertemente integrado en la estructura de los agentes. Sin embargo, en ALZ-MAS 2.0, el mecanismo planificador ha sido modelado como un servicio independiente, es decir, como un programa al que se le envían una serie de parámetros a través de la plataforma de agentes y devuelve el resultado una vez procesada la petición. El mecanismo planificador está diseñado para organizar de forma automática y dinámica algunas de las principales tareas diarias del personal médico dentro de la residencia. Este mecanismo optimiza la asignación de tareas para facilitar su cumplimiento por el personal médico conectado al sistema. Para realizar la planificación, se toman en cuenta diversos parámetros, entre ellos el perfil del enfermero, la prioridad de la tarea, el tiempo mínimo y máximo en que se debe completar la tarea, los objetivos de la tarea, etc.

La **Figura 5.7** muestra los principales pasos para realizar una planificación de tareas. Un plan consiste en una secuencia ordenada de tareas que los enfermeros deben realizar en los tiempos marcados. El ciclo para generar un nuevo plan comienza cuando el *Mobile Agent* envía una petición a la plataforma de agentes. El mensaje es recibido por el *CommApp Agent* y lo reenvía al *Admin Agent*, que decide cual debe ser el servicio a invocar, en este caso el servicio planificador. La plataforma invoca el servicio enviando un mensaje SOAP al servicio planificador para que genere un nuevo plan. El servicio consulta la memoria de casos y recupera las soluciones más similares para resolver el

problema actual. En este caso, se ha permitido una tolerancia de 20% en la similitud de los planes. Para determinar la similitud, se hace uso de algoritmos tales como coseno, *clustering*, etc. Una vez seleccionados los planes similares, el planificador genera una nueva solución (plan). El algoritmo para generar la solución se realiza a través de una red neuronal con restricciones de tiempo y distancias que deben recorrer los enfermeros para ejecutar las tareas. Debido a que no es el objetivo de esta investigación describir en detalle este algoritmo, no se realiza una descripción detallada. Además, este algoritmo está siendo modificado para aceptar un mayor número de restricciones y optimizar su rendimiento. El servicio planificador envía una solución global a la plataforma de agentes. Esta solución está compuesta por todas las tareas de todos los enfermeros. La solución global es dividida por el *Manager Agent* y se envían las tareas propias de cada enfermero a los correspondientes *Nurse Agent* y *Mobile Agent* conectados al sistema.

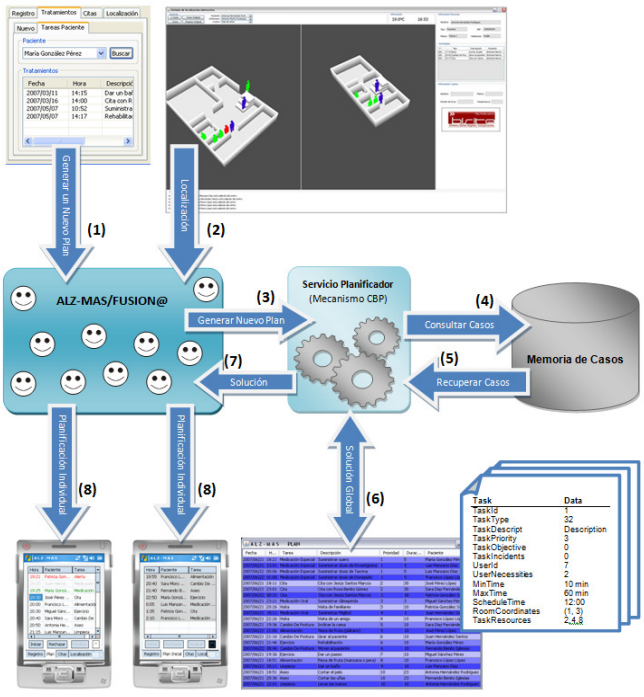

Figura 5.7. Principales pasos para generar un plan

Como se aprecia en la **Figura 5.8**, el resultado de cada planificación se visualiza en diversas interfaces gráficas, mostrando la hora de inicio, la descripción, la prioridad, el paciente al que va dirigida determinada tarea, etc.

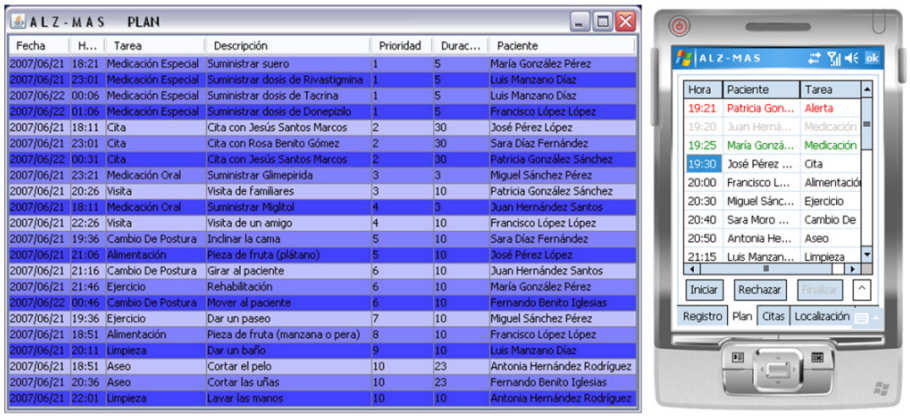

Figura 5.8. Resultado de la planificación. Izquierda: Planificación global; Derecha: Planificación específica para un enfermero mostrada en un emulador de PDA

A continuación se describe la fase de implementación del sistema ALZ-MAS en un escenario real.

5.5. Implementación

Tras el análisis y diseño del sistema ALZ-MAS mediante las herramientas Gaia y SysML, se cuenta con un modelo suficientemente detallado para continuar con la fase de desarrollo del sistema. Esta fase ha sido un proceso intenso de más de dos años de desarrollo constante, en el cual han participado numerosos

desarrolladores e investigadores, principalmente pertenecientes al grupo de investigación BISITE de la Universidad de Salamanca.

Para el desarrollo de FUSION@ y ALZ-MAS se han empleado diversas herramientas para la programación de sistemas multiagente, tales como JADE (Bellifemine, *et al*., 1999) y Jadex (Pokahr, *et al*., 2003). Debido a que la fase de desarrollo es extensa y esta memoria no tiene por objetivo describirla, se pasa directamente a la fase de implementación, la cual se describe a continuación.

Los prototipos de ALZ-MAS, y por consiguiente FUSION@, han sido implementados en una residencia geriátrica[1] de la ciudad de Salamanca, España, aportando una serie de herramientas para optimizar los cuidados y la atención médica a pacientes con la enfermedad de Alzheimer. La residencia cuenta con instalaciones para albergar hasta 60 personas mayores de 65 años. Su personal médico está compuesto por 1 trabajadora social, 2 doctores y un promedio de 6 enfermeros por cada turno de 8 horas diarias. La residencia cuenta con dos plantas: en la primer planta se ubican los dormitorios de todos los residentes, mientras que en la planta baja se encuentran el comedor, la sala de usos múltiples, la enfermería, la peluquería, los aseos, el cuarto de terapia, la oficina y el cuarto de visitas.

La interacción entre los usuarios, el entorno y el propio sistema se produce principalmente a través de un conjunto de agentes inteligentes, y tecnologías inalámbricas. Todos ellos trabajan de forma conjunta y en tiempo de ejecución para proporcionar a los usuarios con informes, respuestas, planes, etc. de forma sencilla. La infraestructura tecnológica consiste en la utilización de dispositivos de identificación por radiofrecuencia para identificar y localizar a los usuarios, dispositivos ZigBee para automatizar servicios (luces, cerraduras, etc.) y dispositivos Wi-Fi. Todos estos dispositivos han sido distribuidos estratégicamente por la residencia geriátrica. Los dispositivos Wi-Fi permiten al sistema comunicarse de forma distribuida con dispositivos móviles, entre ellos

[1] Por motivos de confidencialidad, el nombre de la residencia se omite en esta memoria.

los dispositivos móviles (por ejemplo, PDAs) que el personal médico utiliza para interactuar con el sistema. La **Figura 5.9** muestra la pantalla de acceso al sistema a través de un emulador de PDA.

Figura 5.9. Izquierda: Pantalla de acceso a ALZ-MAS a través de una PDA; Derecha: Visualización de los datos médicos de un paciente

Por otra parte, los dispositivos RFID realizan las tareas de identificación y localización tanto de los pacientes como de los enfermeros dentro de la residencia. La configuración utilizada en esta residencia se compone de una serie de antenas colocadas en cada una de las puertas del edificio. Las antenas son controladas por un conjunto de lectores RFID (4 antenas por lector) que operan en la frecuencia de los 868MHz. La potencia de los lectores ha sido limitada para obtener un alcance de lectura de hasta aproximadamente 3 metros. Los pacientes y enfermeros hacen uso de etiquetas RFID para ser identificados y localizados (los pacientes las llevan ocultas en su ropa, mientras que los enfermeros las llevan en sus respectivas tarjetas acreditativas). Cuando una persona que porte una etiqueta RFID se acerca a una de las zonas en donde se encuentra una antena, el lector obtiene el número ID y lo envía a un ordenador central para que

sea interpretado y gestionado. La información es entonces enviada a distintas interfaces del sistema. La **Figura 5.10** muestra los principales componentes utilizados para la identificación y localización de usuarios.

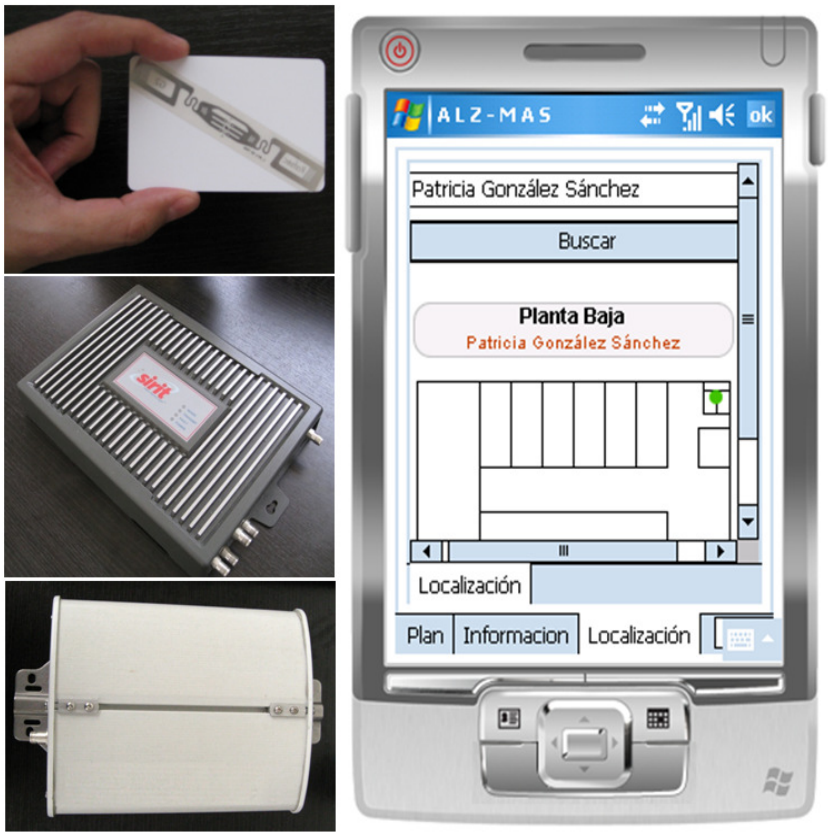

Figura 5.10. De arriba abajo y de izquierda a derecha: Etiqueta utilizada por los pacientes y enfermeros; Lector UHF; Antena direccional de largo alcance; Interfaz de localización ejecutándose en un emulador de PDA

Asimismo, una red de dispositivos ZigBee han sido instalados en la residencia, los cuales permiten obtener una serie de datos sobre el entorno, como la temperatura ambiente, la iluminación, pestillos electrónicos, alarmas, etc. Como se muestra en la **Figura 5.11**, estos dispositivos están basados en las placas de desarrollo C8051F121 de la empresa *Silicon Laboratories* (SiLabs) y

operan en la frecuencia de los 2.4GHz. Entre otras cosas, ZigBee permite un despliegue rápido y sencillo sin afectar estructuralmente la edificación.

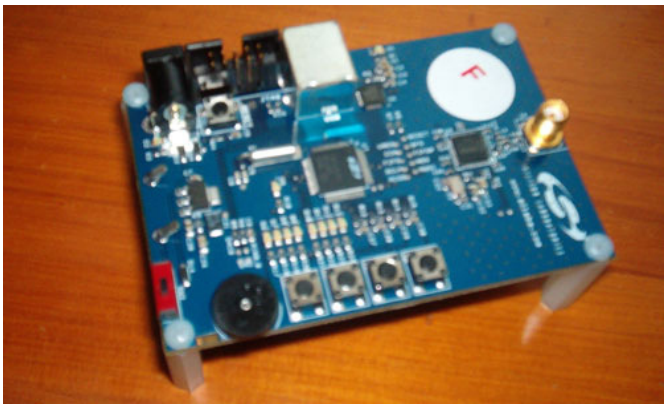

Figura 5.11. Placa de desarrollo C8051F121 de SiLabs

Todos los datos obtenidos por estas tecnologías son procesados por los *Interface Agents* o por los servicios de FUSION@, los cuales envían la información al resto de agentes para que sea gestionada. Los agentes y servicios encargados de recopilar datos del entorno, actualizan automáticamente y de forma periódica la información de los dispositivos y también actúan sobre estos modificando su estado, dependiendo de cada situación en particular.

La información es constantemente actualizada y enviada a las distintas interfaces de usuario. La **Figura 5.12** muestra la interfaz principal de ALZ-MAS, la cual ha evolucionado a lo largo de la fase de desarrollo hasta convertirse en una compleja interfaz que visualiza en tres dimensiones (3D) las distintas plantas de la residencia y además muestra información de los pacientes, el personal médico y el estado de los distintos dispositivos conectados al sistema, especialmente RFID y ZigBee.

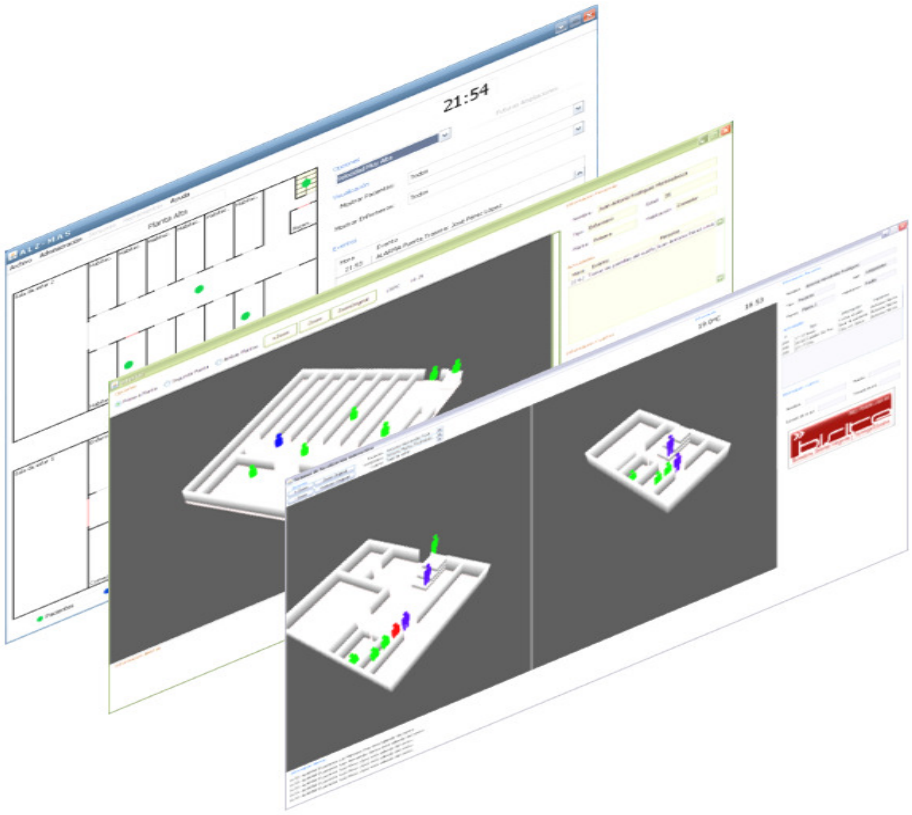

Figura 5.12. Evolución de la interfaz principal de ALZ-MAS

A continuación, se presentan los resultados y conclusiones correspondientes al caso de estudio tras haber finalizado con el desarrollo del sistema ALZ-MAS y su posterior implementación.

5.6. Análisis del Caso de Estudio y Resultados

La arquitectura FUSION@ ha sido evaluada a través del sistema ALZ-MAS, realizado diversos experimentos para demostrar su eficiencia en entornos reales. Se han recopilado una serie de datos, muchos de ellos relacionados con las actividades del personal médico antes y después de la implementación de los diferentes prototipos desarrollados a lo largo de esta investigación. Algunos de los experimentos se han centrado en determinar el tiempo empleado en actividades indirectas al cuidado de los pacientes y en el número de enfermeros que trabajan de forma simultánea en la residencia geriátrica donde se ha puesto en marcha el sistema.

Los resultados obtenidos han sido satisfactorios, demostrando que la arquitectura FUSION@ permite modelar e implementar sistemas basados en el paradigma de la Inteligencia Ambiental de forma eficiente. Asimismo, se ha analizado el impacto de un sistema de este tipo en un entorno real, comprobando como puede optimizar la oferta de servicios tradicionales y aumentar su calidad.

Las tareas de los enfermeros han sido divididas en dos categorías: tareas indirectas y tareas indirectas. Las tareas directas son aquellas que requieren que los enfermeros actúan directamente con los pacientes, por ejemplo dar la medicación, alimentarlos, ducharlos, etc. Por otra parte, las tareas indirectas son aquellas en donde no es necesaria la atención de los enfermeros sobre los pacientes, por ejemplo la elaboración de informes, rondas de vigilancia y el seguimiento de visitantes. ALZ-MAS es capaz de reducir significativamente el tiempo empleado en tareas indirectas. Como se aprecia en la **Figura 5.13**, los tiempos promedio han disminuido tras la implementación del sistema.

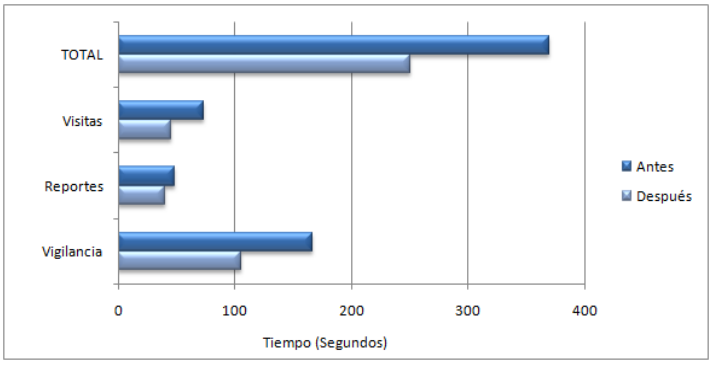

Figura 5.13. Tiempo promedio empleado por los enfermeros en tareas indirectas antes y después de la implementación de ALZ-MAS

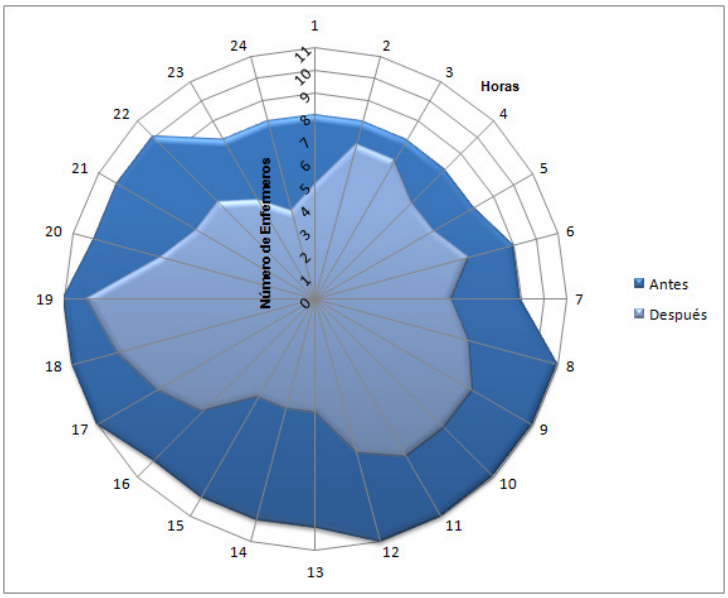

Figura 5.14. Número promedio de enfermeros antes y después de la implementación de ALZ-MAS

La **Figura 5.14** muestra el tiempo promedio de enfermeros trabajando simultáneamente en la residencia en cada hora del día, así como el número promedio de enfermeros calculado por ALZ-MAS para completar todas las tareas planificadas. Al iniciar la jornada laboral se cargan todas las tareas que deben

desempeñar los enfermeros. El sistema realiza una planificación de tareas distribuyéndolas entre cada uno de los enfermeros disponibles teniendo en cuenta una jornada laboral de ocho horas. ALZ-MAS facilita asignación de tareas, por lo que la jornada laboral de los enfermeros es optimizada. Es importante señalar que gran parte de las tareas diarias del personal médico se introdujeron en el sistema, sin embargo, dichas tareas son muy cambiantes, incluso en una misma jornada laboral, por lo que las estimaciones obtenidas por el sistema son solamente determinantes en el sentido de llevar una mejor organización de tareas, sin influir en el número de trabajadores.

La seguridad de la residencia también se ha visto mejorada en cierta forma, ya que el sistema de localización en tiempo real garantiza que los pacientes no ingresen a zonas restringidas o que salgan de la residencia. La **Figura 5.15** muestra el número promedio de accesos a zonas restringidas, apreciando un elevado aumento en la detección de estas incidencias tras la implementación de ALZ-MAS. Este dato en particular es muy revelador, ya que se puede asumir que varias situaciones de riesgo podrían no haber sido detectadas en el pasado. Asimismo, gracias a la tecnología RFID se proporciona una individualización a través de un único número ID, disminuyendo posibles incidencias surgidas por la incorrecta identificación de los pacientes.

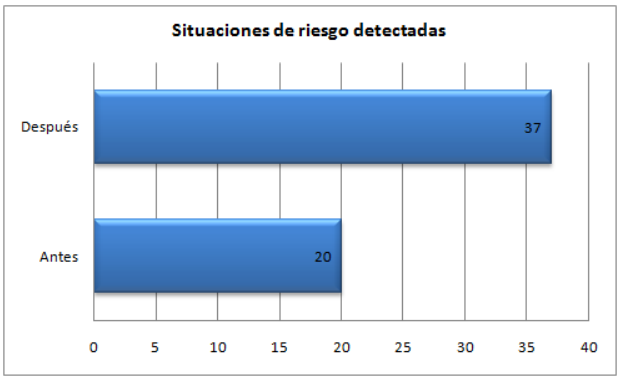

Figura 5.15. Número de accesos a zonas restringidas antes y después de la implementación del sistema ALZ-MAS

Se ha hecho un esfuerzo considerable por desarrollar un sistema útil para las residencias geriátricas, sin embargo, ALZ-MAS sigue siendo un sistema no apto para ser implementado en un escenario totalmente realista sin llevar a cabo una exhaustiva monitorización. Esto se debe principalmente a que la información manejada por el sistema es muy delicada en términos médicos y legales.

Por otra parte, aunque los enfermeros llevan una mejor organización de sus tareas diarias, se generan muchas otras tareas indirectas al cuidado de los pacientes, las cuales necesitan ser gestionadas por personal informático especializado. Tal ha sido el caso de las PDAs utilizadas por el personal médico, ya que estos generalmente no contaban con los conocimientos informáticos adecuados para solucionar diversos problemas, o incluso manejar de forma correcta el sistema. Por tal motivo, fue necesario formar al personal médico en el uso básico de las PDAs proporcionadas, actividad que resultó mucho más compleja de lo esperado.

En lo que respecta al rendimiento del sistema, es inaceptable el hecho de que ciertos componentes del sistema fallen, sobre todo en escenarios médicos y de dependencia. Los fallos detectados en el sistema han permitido corregir gran parte de las funcionalidades, sin embargo, es necesario continuar depurando y evaluando tanto el sistema ALZ-MAS como la arquitectura FUSION@ de cara a obtener una solución mucho más robusta y eficiente.

Sin embargo, aun con estos inconvenientes, ALZ-MAS ha tenido una buena aceptación entre el personal médico. El uso de tecnología inalámbrica ha sido un factor clave en este hecho, debido a que la infraestructura implementada ha pasado prácticamente desapercibida por los usuarios (personal médico y pacientes). En este sentido, la implementación de ALZ-MAS ha sido un éxito, a tal grado de que a partir de la infraestructura propuesta se han generado dos patentes a nivel europeo[2], las cuales consisten en un sistema de localización en

[2] Explotación comercial por la empresa Tulecom Group S.L.

tiempo real con tecnología ZigBee y un sistema de control de accesos con tecnología RFID.

5.7. Conclusiones

ALZ-MAS es un sistema multiagente que hace uso de la arquitectura FUSION@ y se centra en mejorar los servicios de asistencia y cuidados médicos para personas dependientes, especialmente con la enfermedad de Alzheimer.

El sistema ALZ-MAS aprovecha las características que proporciona la arquitectura FUSION@, definiendo un conjunto de agentes y haciendo uso de tecnologías inalámbricas como RFID, Wi-Fi y ZigBee. Esto proporciona al sistema la capacidad de obtener información del contexto y una comunicación distribuida entre sus componentes, logrando elevados niveles de interacción entre los usuarios y el entorno de forma transparente, sencilla e intuitiva.

ALZ-MAS es una herramienta útil, capaz de optimizar la jornada laboral del personal médico de la residencia al realizar una asignación dinámica de tareas. Asimismo, el sistema facilita el cumplimiento de tareas indirectas al cuidado de los pacientes, entre ellas la monitorización de tratamientos y el saber en todo momento en dónde se encuentran cada uno de los pacientes de forma automática.

Por otra parte, se han logrado reutilizar algunas de las funcionalidades de ALZ-MAS en otros desarrollos del grupo de investigación BISITE, entre las que destacan el mecanismo planificador en un sistema para gestionar y monitorizar rutas de vigilancia en entornos industriales (Tapia, *et al*., 2009).

Capítulo

6

Conclusiones

El crecimiento constante de las tecnologías de la comunicación y de la sociedad de la información en los últimos años supone la necesidad de ofrecer nuevas soluciones que faciliten la interacción de los usuarios con la tecnología. En esta memoria se ha presentado FUSION@, una arquitectura que integra la tecnología de agentes con una filosofía orientada a servicios para facilitar y optimizar el desarrollo de sistemas multiagente distribuidos basados en el paradigma de la Inteligencia Ambiental.

Este capítulo describe como se han alcanzado los distintos objetivos definidos en esta investigación para validar y evaluar la hipótesis de partida: "*La arquitectura FUSION@ facilita el desarrollo de sistemas multiagente de forma eficiente, permitiendo que estos tengan una mayor escalabilidad y capacidad para la reutilización de recursos. Además, FUSION@ permite extraer y modelar las funcionalidades de los agentes como servicios independientes, obteniendo agentes con baja carga computacional, lo cual hace que esta arquitectura sea única en su*

tipo. Asimismo, un enfoque distribuido dota a la arquitectura de una elevada capacidad de recuperación ante errores, así como una mayor flexibilidad para adaptar su comportamiento en tiempo de ejecución.". Por otra parte, se presentan las principales contribuciones de esta investigación y se describen algunas de las posibles líneas de trabajo a seguir.

Consideramos que los requerimientos más importantes que propone la Inteligencia Ambiental han sido cubiertos por FUSION@:

- ✓ **Computación ubicua.** Gracias a las características de los agentes se proporciona un marco robusto, pero lo suficientemente flexible, para cubrir los requerimientos de los sistemas basados en la Inteligencia Ambiental en escenarios dinámicos y distribuidos con fuerte interacción con los usuarios. Se trata de elementos de computación que trabajan de forma distribuida. En este sentido, y dada la capacidad de colaboración existente en los sistemas multiagente, es posible hablar de una computación ubicua.

- ✓ **Comunicación ubicua.** El enfoque distribuido, la flexibilidad para que los usuarios accedan a los servicios, la capacidad de elección de los agentes, así como la utilización de hardware integrado en el entorno, permiten que las distintas funcionalidades se comuniquen de manera transparente para los usuarios. Los usuarios pueden hacer uso de las funcionalidades que ofrece el sistema desde distintas aplicaciones y a través de diversos medios, como es el caso de los dispositivos móviles.

- ✓ **Inteligencia.** El núcleo de FUSION@ es un conjunto de agentes deliberativos basados en el modelo BDI. Estos agentes proporcionan a la arquitectura una gran capacidad de adaptación y facilitan el proceso de toma de decisiones. Asimismo, los agentes son capaces de actuar de forma autónoma y generar conocimiento a través de mecanismos de razonamiento, como es el caso de los sistemas CBR.

- ✓ **Sensibilidad al contexto.** La integración de tecnologías como RFID, ZigBee y Wi-Fi, aportan a los agentes la capacidad para percibir estímulos del entorno de forma automática y en tiempo de ejecución. De esta forma, es posible personalizar el comportamiento del sistema para ajustarse a las características y necesidades del contexto en cada situación determinada.

Por otra parte, la arquitectura propuesta presenta una serie de características que facilitan y optimizan el desarrollo de sistemas multiagente distribuidos basados en la Inteligencia Ambiental:

- ✓ **Distribución de recursos.** Las funcionalidades de los sistemas y de los propios agentes se modelan como aplicaciones y servicios independientes. De esta forma, se obtienen agentes más ligeros en términos de carga computacional, lo que permite expandir las posibilidades de desarrollo sobre dispositivos con reducidas capacidades de procesamiento (PDAs, teléfonos móviles, microcontroladores, etc.).

- ✓ **Reutilización de funcionalidades.** Las funcionalidades se modelan como aplicaciones y servicios independientes. De esta forma, es posible utilizarlas en distintos desarrollos, realizando modificaciones menores para ajustarse a las necesidades de cada escenario. Asimismo, las funcionalidades pueden ser replicadas para así obtener una mayor disponibilidad de recursos.

- ✓ **Robustez.** El enfoque distribuido de FUSION@ permite inicializar y detener aplicaciones, servicios o agentes, de forma independiente y sin afectar el resto de componentes del sistema.

La **Figura 6.1** intenta reflejar de forma general la diferencia que existe entre la arquitectura FUSION@ y otros modelos de arquitecturas distribuidas. Para el desarrollo de FUSION@ se ha intentado equilibrar la descentralización con la

inteligencia. La descentralización la hemos definido como el resultado de distribuir las funcionalidades, la reutilización que se puede lograr con ellas, y la independencia que existe con los lenguajes de programación. Por otra parte, la inteligencia, que se puede definir como el resultado de las capacidades de razonamiento, de adaptación del comportamiento de forma autónoma, y de percibir estímulos del contexto y reaccionar ante éstos de forma personalizada. Mientras que FUSION@ intenta lograr un equilibrio, alternativas SOA o *Web Services* presentan importantes limitaciones en cuanto al nivel de negociación entre servicios y en la sensibilidad al contexto. Por otra parte, CORBA no es lo suficientemente independiente de lenguajes de programación y las aplicaciones desarrolladas no siempre son compatibles entre sí. Finalmente, las plataformas de agentes, aunque proporcionan herramientas muy útiles para lograr sistemas inteligentes, no facilitan la compatibilidad entre plataformas ni ofrecen las herramientas necesarias para lograr una descentralización de funcionalidades más eficiente.

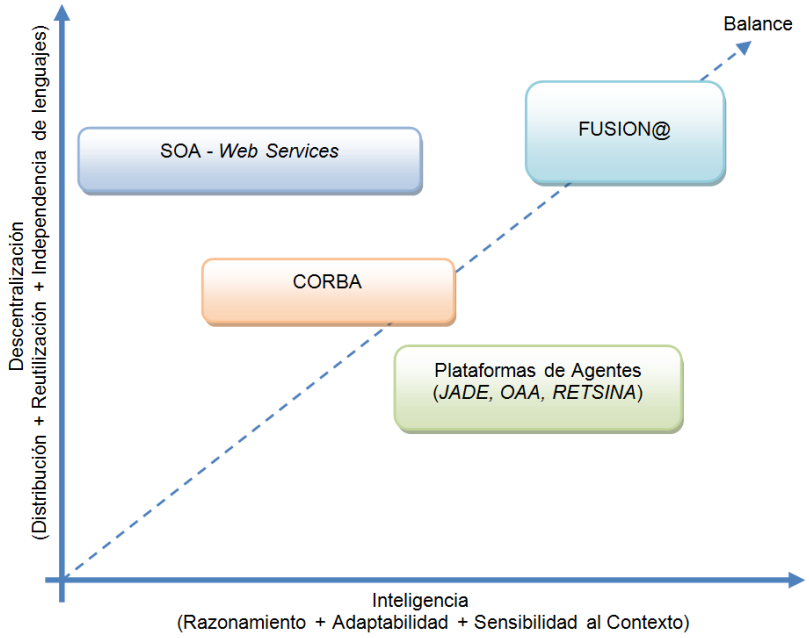

Figura 6.1. Comparativa gráfica de FUSION@ con otros modelos de arquitectura

Los ancianos y las personas dependientes representan un gran reto para el desarrollo de nuevas tecnologías que mejoren la interacción con su entorno. Esta ha sido la principal razón para elegir una residencia de enfermos de Alzheimer como un escenario adecuado para comprobar la validez de la arquitectura FUSION@ a través del sistema ALZ-MAS, un sistema multiagente diseñado para mejorar los cuidados médicos y servicios de asistencia ofrecidos a este tipo de pacientes. Como se muestra en la **Tabla 6.1**, se han desarrollado dos versiones distintas del sistema ALZ-MAS, una de ellas (versión 1.0) ha sido diseñada y desarrollada siguiendo el enfoque tradicional de los sistemas creados dentro del grupo de investigación BISITE[3] de la Universidad de Salamanca, mientras que la versión 2.0 ha sido desarrollada siguiendo la pauta de la arquitectura FUSION@.

Tabla 6.1. Comparativa entre ALZ-MAS 1.0 y ALZ-MAS 2.0

	ALZ-MAS 1.0	**ALZ-MAS 2.0**
Tipo de arquitectura	✓ Agentes	✓ Agentes + SOA
Plataforma de agentes	✓ JADE	✓ JADE + FUSION@
Protocolos de comunicación soportados	✓ ACL	✓ ACL ✓ SOAP
Tipos de agentes definidos	✓ 5	✓ 12
Sistemas Operativos soportados	✓ Linux ✓ Windows 98/2000/XP/2003 Server/Vista	✓ Linux ✓ Windows 98/2000/XP/2003 Server/Vista
Lenguaje de programación utilizado	✓ Java	✓ Java ✓ C/C++ ✓ .NET
Modelo de distribución de recursos	✓ Replicación de Agentes	✓ Replicación de Agentes ✓ Replicación de Servicios ✓ Replicación de Aplicaciones
Modelo de reutilización de recursos	✓ Funcionalidades profundamente integradas en la estructura de los agentes	✓ Funcionalidades modeladas como servicios y aplicaciones independientes, fuera de la estructura de los agentes ✓ Los servicios pueden utilizarse en otros desarrollos

[3] http://bisite.usal.es

El rendimiento interno de FUSION@ también ha sido evaluado a través de ALZ-MAS. Para ello, se han desarrollado dos versiones distintas del sistema ALZ-MAS, una de ellas haciendo uso de FUSION@ y otra en donde todas las funcionalidades están integradas en los propios agentes de ALZ-MAS. Las principales pruebas han consistido en una serie de simulaciones ejecutando el mecanismo planificador de tareas. La memoria de casos ha sido depurada en cada simulación para eliminar la capacidad de aprendizaje del mecanismo, lo que fuerza al planificador a realizar todo el proceso de asignación de tareas. Se han definido 30 agendas (listas de tareas) distintas, cada una con 50 tareas que el planificador debe organizar en un plan global (todas las tareas de todos los enfermeros). Las simulaciones se han dividido en siete grupos, con 1, 5, 10, 15, 20, 25 y 30 agendas simultáneas. Estas simulaciones se han realizado 50 veces para cada versión del sistema en un mismo ordenador. Para la versión ALZ-MAS 2.0 se han replicado 5 servicios planificadores, todos ellos ejecutándose en el mismo ordenador con las siguientes especificaciones: Procesador Intel Core2 Duo T7500 (2.20GHz); 2MB de RAM; y Sistema Operativo Windows XP con Service Pack 2.

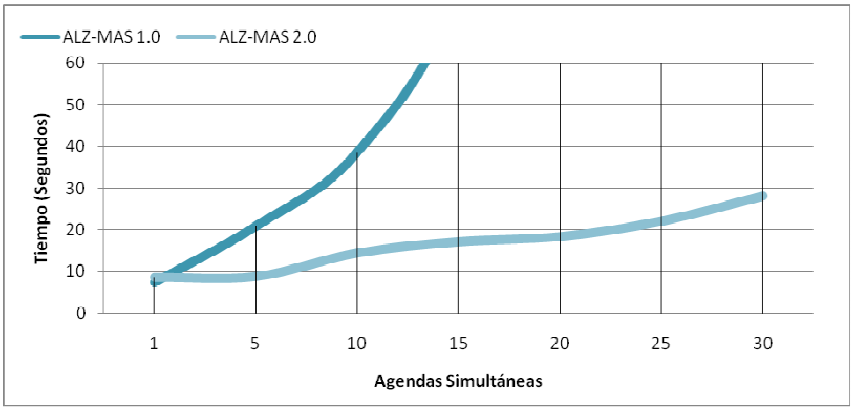

Figura 6.2. Tiempo promedio para planificar las agendas por ambos sistemas

Como se aprecia en la **Figura 6.2**, la cual muestra el tiempo promedio que ambos sistemas necesitan para planificar un determinado número de agendas

simultáneas, la versión ALZ-MAS 1.0 ha sido incapaz de realizar la planificación a partir de 15 agendas simultáneas. Sin embargo, ALZ-MAS 2.0 ha sido capaz de completar la planificación incluso para las 30 agendas simultáneas gracias a los cinco servicios replicados. Un dato importante es que el sistema que no implementa FUSION@ es un poco más rápido al planificar una sola agenda, aunque entre más agendas simultáneas haya, el rendimiento decrece exponencialmente.

Uno de los principales aportes de FUSION@ a ALZ-MAS se encuentra en las mejoras obtenidas en cuanto al rendimiento y robustez del sistema. La **Figura 6.3** muestra el número de agentes bloqueados durante las simulaciones, lo que supone que FUSION@ proporciona una mayor capacidad para la recuperación ante errores.

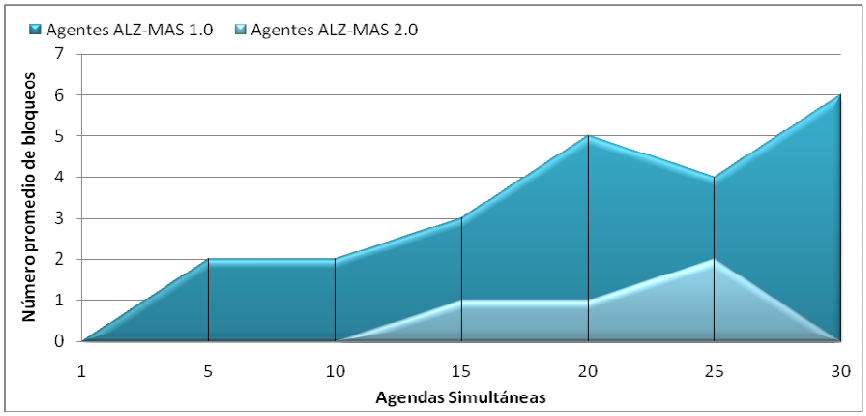

Figura 6.3. Número de agentes y servicios bloqueados

Es importante señalar que se ha realizado un gran esfuerzo para reducir al máximo, incluso evitar por completo, el número de bloqueos, sin embargo, estos resultados demuestran que aún es necesario seguir trabajando para depurar cada uno de los componentes de esta propuesta, desde la arquitectura FUSION@ hasta el propio sistema ALZ-MAS y sus funcionalidades básicas. De esta forma,

será posible lograr un desarrollo más fiable y robusto, aunque ciertamente, los resultados obtenidos hasta ahora son alentadores.

FUSION@ intenta facilitar la comunicación ubicua, incorporando agentes con características especiales de comunicación que actúan como mediadores entre distintas plataformas de agentes. Además, como se muestra en la **Tabla 6.2**, los desarrolladores no se encuentran restringidos al lenguaje de programación propio de la plataforma de agentes utilizada, lo que facilita la integración y la reutilización de funcionalidades. De hecho, durante el desarrollo del sistema ALZ-MAS, se hizo evidente la facilidad que presentan plataformas como .NET para crear interfaces sobre dispositivos móviles, sobre todo en dispositivos que hagan uso del sistema operativo Windows Mobile.

Tabla 6.2. Lenguajes de desarrollo soportados por distintas plataformas de agentes y FUSION@

JADE	OAA	RETSINA	FUSION@ *
✓ Java ✓ .NET **	✓ Java ✓ C/C++ ✓ Prolog	✓ Java ✓ C/C++	✓ Java ✓ C/C++ ✓ .NET ✓ JavaScript
** Funcionalidades desarrolladas hasta el momento.* *** Utilizando extensiones (add-ons) de terceros.*			

En todo desarrollo experimental se cuenta con una serie de ventajas y desventajas que son necesarias analizar de cara a valorar la propuesta realizada y así poder mejorar los aspectos más débiles. La **Tabla 6.3** presenta un resumen de las principales ventajas y desventajas de la arquitectura FUSION@. La optimización en la distribución de recursos es un aporte importante, ya que las funcionalidades son modeladas como aplicaciones y servicios independientes que pueden ser ejecutados de forma local como remota. La independencia de los lenguajes de programación es otro logro significativo, ya que los desarrolladores cuentan con un abanico mayor de posibilidades para desarrollar las distintas

funcionalidades que deseen añadir a la arquitectura o a los sistemas basados en FUSION@. Uno de los logros más substanciales radica en la mejora del rendimiento de los sistemas multiagente, en particular el sistema ALZ-MAS, liberando a los agentes de gran parte de su carga computacional y trasladándola como aplicaciones y servicios externos, facilitando además la reutilización de funcionalidades. Los resultados preliminares demuestran que FUSION@ es aplicable al desarrollo de sistemas y funcionalidades complejos, como es el caso del sistema ALZ-MAS y los servicios planificadores de tareas. Por otra parte, una de las principales desventajas es que la arquitectura FUSION@ continúa en fase de desarrollo y no ha sido completamente depurada. Asimismo, es necesario formalizar la arquitectura en su totalidad para facilitar su utilización por otros desarrolladores e investigadores. Un aspecto negativo que aún permanece en FUSION@ es que se tiene dependencia de la plataforma de agentes utilizada, es decir, los agentes de FUSION@ deberán desarrollarse utilizando los lenguajes de programación y las herramientas soportadas por la propia plataforma.

Tabla 6.3. Principales ventajas y desventajas de la arquitectura FUSION@

Ventajas	Desventajas
✓ Optimiza la distribución de recursos. ✓ Independencia de lenguajes de programación. ✓ Libera a los agentes de carga computacional, trasladándola a los servicios y aplicaciones. ✓ Facilita la reutilización de funcionalidades. ✓ Exitosa implementación en un escenario real (ALZ-MAS). ✓ Define un conjunto de tecnologías para ser utilizadas en desarrollos similares.	⊗ Aún se encuentra en fase de desarrollo y no está completamente depurada ni formalizada. ⊗ Es dependiente de las plataformas de agentes. ⊗ Sólo ha sido completamente implementada en un sistema (ALZ-MAS). Es necesario extender su implementación a otros desarrollos. ⊗ Aún no se han aplicado técnicas de evaluación estandarizadas.

Cabe destacar que, tanto el sistema ALZ-MAS como las tecnologías presentadas en esta investigación, son tan solo un ejemplo del potencial de

FUSION@ para desarrollar sistemas basados en el paradigma de la Inteligencia Ambiental.

La investigación realizada en este trabajo de tesis doctoral ha contribuido en gran medida a sentar las bases para futuros proyectos, no solo dentro del grupo de investigación BISITE, sino también en colaboración con entidades públicas y privadas. Las principales contribuciones de esta investigación se describen a continuación.

6.1. Contribuciones de la Investigación

El trabajo de investigación presentado en esta memoria aporta algunas nuevas contribuciones, principalmente en los ámbitos de la Teoría de Agentes y de la Inteligencia Ambiental. A continuación, se describen las principales contribuciones de esta investigación.

- **Marco para el análisis y diseño de sistemas multiagente.** Desde el punto de vista de la ingeniería del software se ha realizado una revisión de las herramientas para el análisis y diseño de sistemas multiagente Gaia y SysML, tratando de utilizar las ventajas de cada una de ellas y de eliminar los inconvenientes en cuanto a notación y contenidos que separan a ambas herramientas. El resultado es una combinación de herramientas que permite modelar entornos complejos y dinámicos basados en la Inteligencia Ambiental de forma eficiente. El enfoque hibrido resultante permite la incorporación de nuevos elementos en un sistema desarrollado y no está restringido a dominios de desarrollo específicos

- **Marco para el desarrollo de sistemas basados en la Inteligencia Ambiental.** Se ha realizado una revisión del estado del arte de la Inteligencia Ambiental y algunas áreas relacionadas, entre las que destaca la domótica. Asimismo, se han analizado un conjunto de tecnologías que permiten a los sistemas acercarse a la visión propuesta por la Inteligencia Ambiental. Estas tecnologías proporcionan una infraestructura robusta que permite a los agentes comunicarse de forma distribuida y obtener información del contexto de forma automática y ubicua, requerimientos fundamentales en este tipo de desarrollos.

- **Estudio de tecnologías inalámbricas y de automatización.** Se han analizado en profundidad las tecnologías RFID, Wi-Fi, ZigBee, X10, KNX y LonWorks. Este análisis es un compendio completo para que nuevos investigadores dispongan de literatura básica para iniciar futuros proyectos en áreas relacionadas. Por otra parte, algunas de estas tecnologías han sido aplicadas en un entorno real, descrito en detalle en el caso de estudio.

- **Propuesta de una arquitectura para entornos de Inteligencia Ambiental.** Se ha propuesto una arquitectura que optimiza el desarrollo de sistemas multiagente distribuidos para entornos dinámicos y con fuerte interacción con los usuarios. La arquitectura proporciona elementos basados en la inteligencia, la sensibilidad al contexto así como la computación, la comunicación y la interacción ubicuas. La arquitectura ha sido utilizada para crear un entorno inteligente en un escenario real y se han evaluado positivamente sus capacidades para modelar problemas de Inteligencia Ambiental.

- **Modelado del contexto.** Una parte fundamental en todo desarrollo basado en la Inteligencia Ambiental es el modelado del contexto, en este caso en un escenario de dependencia. Se ha realizado un modelado del sistema ALZ-MAS, presentado en detalle en el caso de estudio,

describiendo cada una de las fases que llevan a lograr un modelo especialmente diseñado en torno a las necesidades de los usuarios. Este modelo servirá como base para el desarrollo de posteriores proyectos y líneas de investigación afines.

- **Aplicación de la arquitectura a un problema real.** Se ha definido un caso de estudio para comprobar la validez de la arquitectura propuesta, el cual consiste en el desarrollo de ALZ-MAS, un sistema multiagente enfocado a mejorar los cuidados médicos y servicios de asistencia de pacientes con la enfermedad de Alzheimer en residencias geriátricas. De esta forma, se ha podido demostrar que la arquitectura propuesta es una alternativa viable para desarrollar sistemas multiagente distribuidos sobre entornos altamente dinámicos. Por otra parte, el sistema ALZ-MAS es capaz de mejorar diversos aspectos relacionados con administración de los cuidados médicos en residencias geriátricas.

- **Intercambio de conocimiento.** Durante el proceso de desarrollo de esta investigación se ha realizado un gran esfuerzo por obtener una retroalimentación de distintos investigadores y grupos de investigación en áreas relacionadas con el objetivo de robustecer esta investigación mediante el intercambio mutuo de ideas y conocimiento. Se ha puesto especial interés en difundir nuestras experiencias y los avances de esta investigación, desde sus etapas iniciales hasta su forma final, a través de diversas publicaciones[4] y la asistencia a conferencias, congresos, *workshops*, etc.

La investigación presentada en esta memoria es relevante sobre todo desde el punto de vista científico, sentando las bases para el desarrollo de importantes proyectos de ámbito nacional:

[4] El listado de publicaciones realizadas está disponible a través de la página web http://bisite.usal.es del Grupo de investigación BISITE de la Universidad de Salamanca.

- Proyecto ALIADO[5] (*ALzheimer Intelligent Ambient DOmotic system*) FIT-350300-2007-84, otorgado por el Ministerio de Industria Turismo y Comercio.

- Proyecto GERMAP[6] (Plataforma Inteligente para la Gestión Integral de Residencias Geriátricas) 137/2007, otorgado por el Ministerio de Trabajo y Asuntos Sociales, IMSERSO.

- Proyecto FUSION@ (*Flexible User and ServIces Oriented multi-ageNt Architecture*), presentado a la Convocatoria Proyectos de Investigación Fundamental Orientada a la Transmisión de Conocimiento a la Empresa (TRACE).

Estos proyectos permitirán continuar con el desarrollo de esta investigación y aplicarla en escenarios todavía más complejos.

6.2. Trabajo a Futuro

La investigación presentada en este trabajo de tesis doctoral deja abiertas una gran cantidad de puertas para un trabajo futuro. Asimismo, importantes entidades, tanto públicas como privadas, han mostrado su interés en participar en el avance y crecimiento de dicha investigación, aportando sus conocimientos en áreas médicas, científicas y tecnológicas, por lo que el futuro de esta investigación resulta prometedor a medio plazo. Entre estas entidades destacan:

- El Grupo de Investigación BISITE de la Universidad de Salamanca.
- La empresa Tulecom[7] (Tulecom Group S. L.).

[5] http://proyecto-aliado.org/home/index.shtml
[6] http://www.boe.es/boe/dias/2008/03/31/pdfs/A18085-18086.pdf

- La Asociación de familiares de Enfermos de Alzheimer de Salamanca (AFA Salamanca)[8].
- El Centro de Referencia Estatal de Atención a Personas con Enfermedad de Alzheimer y otras Demencias de Salamanca (CRE Salamanca)[9].
- El Grupo de Investigación MAmI[10] de la Universidad de Castilla-La Mancha.
- El Grupo de Investigación GHIA[11] de la Universidad de la Universidad Autónoma de Madrid.
- El Centro Tecnológico para el Desarrollo de las Telecomunicaciones de Castilla y León (CEDETEL)[12].

A continuación, se describen algunas de las posibles líneas de trabajo a seguir basándose en esta investigación:

- **Seguridad y protección de datos.** Es importante dotar de una mayor seguridad las comunicaciones realizadas dentro de la arquitectura FUSION@, ya que los datos intercambiados pueden considerarse importantes, lo que hace necesario garantizar la privacidad y la protección de los datos. Actualmente, se está trabajando en la integración de un nuevo motor[13] para el filtrado de mensajes XML, así como de cadenas SQL embebidas en estos mensajes.

- **Sistemas de localización más efectivos y precisos.** Como parte del proyecto ALIADO, se está desarrollando un sistema de localización en tiempo real en interiores[14] capaz de lograr una precisión menor de 30cm. Esto supondrá un gran avance en la personalización de

[7] http://www.tulecom.com
[8] http://www.afasalamanca.org
[9] http://www.crealzheimer.es
[10] http://mami.uclm.es
[11] http://astreo.ii.uam.es/~ghia
[12] http://www.cedetel.es
[13] Proyecto de tesis doctoral desarrollado por Cristián Pinzón. Grupo de Investigación BISITE.
[14] Módulo desarrollado en conjunto por el grupo de investigación BISITE y la empresa Tulecom.

contenidos y el diseño de interfaces multimodales que permitan avanzar aún más hacia la visión propuesta por la Inteligencia Ambiental.

- **Integración Gaia-SysML.** Parece adecuado realizar un estudio detallado de la integración de las herramientas Gaia y SysML. Además, sería conveniente desarrollar una herramienta que facilite a los diseñadores y desarrolladores el proceso de análisis y diseño de los sistemas.

- **Ontología FUSION@ y ALZ-MAS.** El desarrollo de ontologías hace posible que los modelos obtenidos sean más abstractos y extrapolables a otros desarrollos. Por tal motivo, es deseable desarrollar ontologías tanto de la arquitectura FUSION@ como del sistema ALZ-MAS de cara a complementar el marco propuesto para facilitar la construcción de sistemas basados en estos desarrollos.

- **Resolución de nuevos problemas prácticos.** Para realizar una comprobación más exhaustiva de la validez de la arquitectura propuesta, es necesario aplicarla a nuevos problemas prácticos. De esta forma, sería posible comprobar si se adapta de forma adecuada a la resolución de problemas de características distintas, o bien si se ha construido una arquitectura muy restringida a los problemas concretos que se han estudiado durante la realización de este trabajo. En este sentido, se está trabajando actualmente en el desarrollo de diversos sistemas[15] basados en FUSION@, obteniendo resultados preliminares alentadores.

- **Infraestructura domótica.** Haciendo uso de una arquitectura domótica más robusta y flexible es posible gestionar de forma más óptima una mayor cantidad de servicios físicos (iluminación, temperatura, alarmas, etc.). En este punto, se está trabajando en el desarrollo de una arquitectura[16] basada en SOA y compatible con FUSION@ para redes de

[15] Proyectos de tesis doctoral desarrollados por Aitor Mata y Rosa Cano. Grupo de Investigación BISITE.
[16] Proyecto desarrollado en conjunto por CEDETEL, el Grupo BISITE y la empresa Tulecom.

sensores inalámbricos (WSN: *Wireless Sensors Networks*) que permita un despliegue rápido de la infraestructura y una optimización de las funcionalidades en términos de rendimiento, distribución de recursos, mantenimiento, etc.

- **Pruebas y validación.** Es necesario realizar pruebas mucho mas exhaustivas con el objetivo de evaluar en detalle los modelos propuestos en términos de tiempo, facilidad y calidad de análisis y diseño, tiempo de cálculo de respuestas, calidad de las respuestas, etc. Los resultados obtenidos permitirán desarrollar modelos y sistemas más depurados y robustos.

Chapter 7

Research Overview

Ambient Intelligence (AmI) offers a great potential to improve quality of life and simplify the use of technology by offering a wider range of personalized services and providing users with easier and more efficient ways to communicate and interact with other people and systems. However, the development of systems that clearly fulfil the needs of AmI is difficult and not always satisfactory. It requires a joint development of models, techniques and technologies based on dynamic services. An AmI-based system consists on a set of human actors and adaptive mechanisms which work together in a distributed way. Those mechanisms provide on demand personalized services and stimulate users through their environment according specific situation characteristics.

This chapter is a summary of the research presented in this thesis work. In the next section, the specific problem description that essentially motivated the development of a new architecture (FUSION@) is presented. Section 2 briefly

describes the Ambient Intelligence paradigm and some related technologies. Section 3 presents the agents technology and analyses the feasibility of distributed architectures in AmI environments. Section 4 describes the main characteristics of FUSION@ and briefly explains some of its components. Section 5 presents a case study consisting of a distributed multi-agent system for health care scenarios developed using this architecture. Finally section 6 presents the results and conclusions obtained.

7.1. Objectives and Motivation

The development of AmI-based software requires creating increasingly complex and flexible applications, so there is a trend toward reusing resources and share compatible platforms or architectures. In some cases, applications require similar functionalities already implemented into other systems which are not always compatible. At this point, developers can face this problem through two options: reuse functionalities already implemented into other systems; or re-deploy the capabilities required, which means more time for development, although this is the easiest and safest option in most cases. While the first option is more adequate in the long run, the second one is most chosen by developers, which leads to have replicated functionalities as well as greater difficulty in migrating systems and applications. Moreover, the absence of a strategy for integrating applications generates multiple points of failure that can affect the systems' performance. This is a poorly scalable and flexible model with reduced response to change, in which applications are designed from the outset as independent software islands.

For these reasons, it is necessary to develop new functional architectures capable of providing adaptable and compatible frameworks, allowing access to services and applications regardless of time and location restrictions. A functional architecture defines the physical and logical structure of the components that make up a system, as well as the interactions between those components (Franklin, *et al*., 1996). There are Service Oriented Architectures (SOA) and agent frameworks and platforms which provide tools for developing distributed systems and multi-agent systems (Martin, *et al*., 1999) (Sycara, *et al*., 2003) (Bellifemine, *et al*., 1999). However, these tools do not solve by themselves the AmI-based systems needs. Therefore, it is necessary to develop innovative solutions that integrate different approaches in order to create flexible and adaptable systems, especially for achieving higher levels of interaction with people in a ubiquitous and intelligent way.

The main objective of this research is to design an architecture that is applicable to the development of intelligent environments based on the Ambient Intelligence paradigm. The architecture should propose several features capable of being executed in dynamic and distributed environments and facilitate the interaction between users and the environment. These features should be implemented in devices with limited storage and processing capabilities. In addition, the architecture must provide mechanisms that allow modelling Ambient Intelligence problems in terms of agents and multi-agent systems. In this sense, we have defined a *Flexible User and ServIces Oriented multi-ageNt Architecture* (FUSION@). FUSION@ integrates intelligent agents and a service-oriented philosophy. It also defines the use of a set of technologies that allow the agents to obtain information about the context and automatically react over the environment.

The initial hypothesis consisted on demonstrate that FUSION@ would allow the development of multi-agent systems with increased scalability and reuse of resources. In addition, FUSION@ would extract and model the functionalities of

the agents as independent services and applications, giving as a result lighter agents in terms of computational processing. A distributed approach would provide the architecture with a greater capacity for recovery from errors, and such as greater flexibility to adjust its behaviour in execution time.

To achieve the proposed objectives, it was necessary to analyze the state the art of Ambient Intelligence, the agents and multi-agent technology, reasoning models associated with the agents, distributed architectures, methodologies for the analysis and design of multi-agent systems, and several technologies compatible with those objectives.

7.2. Ambient Intelligence

People are currently surrounded by technology which tries to increase our quality of life and facilitate our daily activities. However, there are situations where technology is difficult to handle or people are lack of knowledge to use it.

Ambient Intelligence (AmI) is an emerging multidisciplinary area (ISTAG, 2003) (Ducatel, *et al*., 2001b) (Aarts, *et al*., 2002) (Carretero, *et al*., 2005) (Tapia, *et al*., 2007a) (Tapia, *et al*., 2007c) based on ubiquitous computing (Weiser, 1993) (Weiser, 1995) which influences the design of protocols, communications, systems, devices, etc. (Mukherjee, *et al*., 2006) (Reynolds, 2006). Ambient Intelligence proposes new ways of interaction between people and technology, making it suited to the needs of individuals and the environment that surrounds those (Aarts, *et al*., 2003a) (Carretero, *et al*., 2005). Ambient Intelligence tries to adapt the technology to the people's needs by proposing three basic concepts: ubiquitous computing, ubiquitous communication and intelligent user interfaces.

The vision of AmI assumes seamless, unobtrusive, and often invisible but controllable interactions between humans and technology. AmI provides new possibilities for solving a wide scope of problems. It also proposes a new way to interact between people and technology, where this last one is adapted to individuals and their context. Ambient Intelligence shows a vision where people are surrounded by intelligent interfaces merged in daily life objects (Emiliani, *et al*., 2005) (Haya, *et al*., 2005) (Aarts, *et al*., 2003a), creating computing-capable environments with intelligent communication and processing, serving people by means of a simple, natural and effortless human-system interaction (Richter, *et al*., 2004) (Ducatel, *et al*., 2001a).

There are several related areas with Ambient Intelligence, such as Ubiquitous Computing, Pervasive Computing, Sentient Computing, Context-Aware Computing, Computational Creativity and Domotics (Home Automation). This last is especially important as it provides the technological infrastructure needed to exploit the Ambient Intelligence potential.

An essential aspect in this research is the use of technologies that provide the agents with information of the environment and allow them to react upon it. One of these technologies is the Radio Frequency IDentification (RFID), a wireless communication technology used to identify and receive information about humans, animals and objects on the move. An RFID system contains basically four components: tags, readers, antennas and software. Mainly used in industrial/manufacturing, transportation and distribution, there are other growing sectors, including health care (ITAA, 2004) (SDSU, 2005). Another important technology is Wireless LANs (Local Area Networks) which increase the mobility, flexibility and efficiency of the users, allowing programs, data and resources to be available no matter the physical location (Bénédet, *et al*., 2004) (Sun Microsystems, 2000). New handheld devices facilitate the use of new interaction techniques, for instance, some systems focus on facilitating users with guidance or location systems by means of their wireless devices (Corchado,

et al., 2005). FUSION@ incorporates "lightweight" agents that can reside in mobile devices, such as cellular phones, PDAs, etc., and therefore support wireless communication. This facilitates the portability to a wide range of devices. We have also explored automation technologies such as ZigBee, a low cost, low power consumption, two-way, wireless communication standard, developed by the ZigBee Alliance (ZigBee Standards Organization, 2006). ZigBee is designed to be embedded in consumer electronics, home and building automation, industrial controls, PC peripherals, medical sensor applications, toys and games, and is intended for home, building and industrial automation purposes, addressing the needs of monitoring, control and sensory network applications (ZigBee Standards Organization, 2006). The information collected through these technologies is processed by intelligent agents in FUSION@, providing the flexibility to develop Ambient Intelligence based systems with self-adaptive capabilities to changes in the environment and user necessities.

7.3. Agents and Multi-Agent Technology and Distributed Architectures

The emergence of Ambient Intelligence involves substantial changes in the design of functional architectures, since it is necessary to provide features which enable a ubiquitous computing and communication and also an intelligent interaction with users. This section discusses some of the most important problems of existent functional architectures, including their suitability for constructing intelligent environments according the AmI paradigm. This section also presents the strengths and weaknesses of existent platforms and analyzes the feasibility of a new alternative: FUSION@.

Agent and multi-agent systems have been successfully applied to several Ambient Intelligence scenarios, such as education, culture, entertainment, medicine, robotics, etc. (Tapia, *et al*., 2006a) (Tapia, *et al*., 2007d) (Sadeh, *et al*., 2005) (Bajo, *et al*., 2006c) (Bajo, *et al*., 2006b) (Schön, *et al*., 2005) (Angulo, *et al*., 2004). An *agent* can be defined as a computational system situated in an environment and is able to act autonomously in this environment to achieve its design goals (Wooldridge, 2002). Expanding this definition, we have that an *agent* is *anything* with the ability to perceive its environment through sensors and respond in the same environment through actuators, assuming that each agent may perceive its own actions and learn from the experience (Russell, *et al*., 1995). A multi-agent system (MAS) is defined as any system composed of multiple autonomous agents with incomplete capabilities to solve a global problem, where there is no global control system, the data is decentralized and the computing is asynchronous (Wooldridge, 2002) (Jennings, *et al*., 1998) (ANA-MAS, 2005).

The agents have characteristics such as autonomy, reasoning, reactivity, social abilities, pro-activity, etc. which make them appropriate for developing dynamic and distributed systems based on Ambient Intelligence, as they possess the capability of adapting themselves to the users and environmental characteristics (Jayaputera, *et al*., 2007). In addition, the continuous advancement in mobile computing makes it possible to obtain information about the context and also to react physically to it in more innovative ways (Jayaputera, *et al*., 2007). To facilitate ubiquitous computation capabilities, the agents in FUSION@ are based on the deliberative Belief, Desire, Intention (BDI) model (Jennings, *et al*., 1995) (Bratman, *et al*., 1988) (Pokahr, *et al*., 2003), where the agents' internal structure and capabilities are based on mental aptitudes, using beliefs, desires and intentions (Bratman, 1987) (Georgeff, *et al*., 1998). Nevertheless, Ambient Intelligence developments need higher adaptation, learning and autonomy levels than pure BDI model (Bratman, *et al*., 1988), since it is necessary to incorporate memory management and reasoning mechanisms.

This is achieved in FUSION@ by modelling the agents' characteristics to provide them with innovative mechanisms that allow solving complex problems and autonomous learning. Deliberative agents use these mechanisms to learn from past experiences and to adapt their behaviour according the context. One of the advantages of these intelligent reasoning mechanisms is the effective distribution of computational tasks into services and applications.

Excessive centralization of services negatively affects the systems' functionalities, overcharging or limiting their capabilities. Classical functional architectures are characterized by trying to find modularity and a structure oriented to the system itself. On the other hand, modern functional architectures like Service-Oriented Architecture (SOA) or CORBA consider integration and performance aspects that must be taken into account when functionalities are created outside the system. Distributed architectures are aimed at the interoperability between different systems, distribution of resources, and the lack of dependency of programming languages (Cerami, 2002). Functionalities are linked by means of standard communication protocols that must be used by applications in order to share resources in the network (Ardissono, *et al*., 2004). The compatibility and management of messages between functionalities is an important and complex element in any of these approaches.

One of the most prevalent alternatives in distributed architectures is agent and multi-agent systems which can help to distribute resources and reduce the central unit tasks (Ardissono, *et al*., 2004) (Voos, 2006). There are several agent frameworks and platforms, such as Open Agent Architecture (OAA) (Martin, *et al*., 1999), RETSINA (Sycara, *et al*., 2003) and JADE (Bellifemine, *et al*., 1999) which provide a wide range of tools for developing distributed multi-agent systems. FUSION@ sets on top of these architectures by adding new layers for integrating a SOA approach and facilitating the distribution and management of functionalities (i.e. services). A distributed agents-based architecture provides more flexible ways to move functions to where actions are needed, thus obtaining better responses at execution time, autonomy, services continuity, and

superior levels of flexibility and scalability than centralized architectures (Camarinha-Matos, *et al*., 2007). Additionally, the programming effort is reduced because it is only necessary to specify global objectives so that agents cooperate in solving problems and reaching specific goals, thus giving the systems the ability to generate knowledge and experience. Unfortunately, the difficulty in developing a multi-agent architecture is higher and because there are no specialized programming tools to develop agents, the programmer needs to type a lot of code to create services and clients (Rigole, *et al*., 2002). It is also necessary to have a more complex system analysis and design, which implies more time to reach the implementation stage. Moreover, the system control is reduced because the agents need more autonomy to solve complex problems. The development of agents is an essential piece in the analysis of data from distributed sensors and gives those sensors the ability to work together and analyze complex situations, thus achieving high levels of interaction with humans (Pecora, *et al*., 2007).

Agent and multi-agent systems combine classical and modern functional architecture aspects. Multi-agent systems are structured by taking into account the modularity in the system, and by reuse, integration and performance. Nevertheless, integration is not always achieved because of the incompatibility among the agent platforms. The integration and interoperability of agents and multi-agent systems with SOA and Web Services approaches has been recently explored (Ardissono, *et al*., 2004). Some developments are centred on communication between these models, while others are centred on the integration of distributed services, especially Web Services, into the structure of the agents. Bonino da Silva *et al*. (2007) propose merging multi-agent techniques with semantic web services to enable dynamic, context-aware service composition. They focus on SOA in designing a multi-agent service composition as an intelligent control layer, where agents discover services and adapt their behaviour and capabilities according to semantic service descriptions. Ricci *et al*. (2007) have developed a java-based framework to create SOA and Web Services

compliant applications, which are modelled as agents. Communication between agents and services is performed by using what they call "artifacts" and WSDL (Web Service Definition Language). Shafiq *et al*. (2006) propose a gateway that allows interoperability between Web Services and multi-agent systems. This gateway is an agent that integrates Foundation for Intelligent Physical Agents (FIPA) and The World Wide Web Consortium (W3C) specifications, translating Agent Communication Language (ACL), SOAP and WSDL messages, and combines both directories from agent platforms and web services. Li *et al*. (2004) propose a similar approach, but focus on the representation of services. They use SOAP and WSDL messages to interact with agents. Liu (2007) proposes a multi-agent architecture to develop inter-enterprise cooperation systems using SOA and Web Services components and communication protocols. Walton (2006) presents a technique to build multi-agent systems using Web Services, defining a language to represent the dialogs among agents. There are also frameworks, such as Sun's Jini and IBM's WebSphere, which provide several tools to develop SOA-based systems. Jini uses Java technology to develop distributed and adaptive systems over dynamic environments. Rigole *et al*. (2002) have used Jini to create agents on demand into a home automation system, where each agent is defined as a service in the network. WebSphere provides tools for several operating systems and programming languages. However, the systems developed using these frameworks are not open at all because the framework is closed and services and applications must be programmed using a specific programming language that supports their respective proprietary APIs.

Although these developments provide an adequate background for developing distributed multi-agent systems integrating a service oriented approach, most of them are in early stages of development, so it is not possible to actually know their potential in real scenarios. In addition, FUSION@ not only provides communication and integration between distributed agents, services and applications; it also proposes a new method to facilitate the development of distributed multi-agent systems by means of modelling the functionalities of the

agents and the systems as services and applications. Another feature in this architecture is security, which is managed by the agents. All communications must take place via the agents, so services cannot share their resources unless the agents allow it. Besides, services defined for each system must always be available, so they are not shared with other systems unless it is specified. The FUSION@ approach is presented in detail in the following section.

7.4. FUSION@: *A Flexible User and ServIces Oriented multi-ageNt Architecture*

FUSION@ is a novel architecture which integrates a services-oriented approach with intelligent agents for building systems based on the Ambient Intelligence paradigm. These systems must be dynamic, flexible, robust, adaptable to changes in context, scalable and easy to use and maintain. However, the architecture can be used to develop any kind of complex systems. The architecture proposes a new and easier method to develop distributed intelligent ubiquitous systems, where functionalities can communicate in a distributed way with intelligent agents, even from mobile devices, independent of time and location restrictions. The architecture focuses on distributing the majority of the systems' functionalities into remote and local services and applications. The functionalities of the systems are not integrated into the structure of the agents; rather they are modelled as distributed services and applications which are invoked by the agents acting as controllers and coordinators. Because the architecture acts as an interpreter, the users can run applications and services programmed in virtually any language, but have to follow a communication protocol that all applications and services must incorporate. Another important

functionality is that, thanks to the agents' capabilities, the systems developed can make use of reasoning mechanisms or learning techniques to handle services and applications according to context characteristics, which can change dynamically over time. Agents, applications and services can communicate in a distributed way, even from mobile devices. This makes it possible to use resources no matter its location. It also allows the starting or stopping of agents, applications, services or devices separately, without affecting the rest of resources, so the system has an elevated adaptability and capacity for error recovery.

FUSION@ is based on agents due to their characteristics, such as autonomy, reasoning, reactivity, social abilities, pro-activity, mobility, organization, etc., which allow them to cover several needs for Ambient Intelligence environments, especially ubiquitous communication and computing and adaptable interfaces. FUSION@ combines a services-oriented approach and intelligent agents to obtain an innovative architecture that presents important improvements in the area of Ambient Intelligence, facilitating ubiquitous computation and communication and high levels of human-system-environment interaction. It also provides an advanced flexibility and customization to easily add, modify or remove applications or services on demand, independently of the programming language. FUSION@ also facilitates the inclusion of context-aware technologies, as radiofrequency location and identification, that allow systems to automatically obtain information from users and the environment in an evenly distributed way, focusing on the characteristics of ubiquity, awareness, intelligence, mobility, etc., all of which are concepts defined by Ambient Intelligence. Wireless networks are one of the top technologies that have been used to test this architecture in several systems. These networks provide an infrastructure capable of supporting the distributed communication needed for agents, applications and services, and of increasing mobility, flexibility and efficiency since resources can be accessed no matter their physical location. The goal in FUSION@ is not only to distribute services and applications, but to also promote a new way of developing AmI-

based systems focusing on ubiquity and simplicity. FUSION@ provides a flexible distribution of resources and facilitates the inclusion of new functionalities in highly dynamic environments. It also provides the systems with a higher ability to recover from errors and a better flexibility to change their behaviour at execution time.

Multi-agent architectures such as Open Agent Architecture (OAA) (Martin, *et al*., 1999), RETSINA (Sycara, *et al*., 2003) and JADE (Bellifemine, *et al*., 1999) define agent-based structures to resolve distributed computational problems and facilitate user interactions. However, the communication capabilities are limited, not allowing easy integration with services and applications. On the other hand, SOA-based architectures usually do not provide intelligent computational and interactive mechanisms. FUSION@ combines both paradigms, trying to take advantage of their strengths and avoid their weakness.

FUSION@ sets on top of existing agent frameworks by adding new layers to integrate a services-oriented approach and facilitate the distribution and management of resources. FUSION@ defines four basic blocks:

1) **Applications.** These represent all the programs that can be used to exploit the system functionalities. Applications are dynamic and adaptable to context, reacting differently according to the particular situations and the services invoked. They can be executed locally or remotely, even on mobile devices with limited processing capabilities, because computing tasks are largely delegated to the agents and services.
2) **Agent Platform.** This is the core of FUSION@, integrating a set of agents, each one with special characteristics and behaviour. An important feature in this architecture is that the agents act as controllers and administrators for all applications and services, managing the adequate functioning of the system, from services, applications, communication and performance to reasoning and decision-making. In FUSION@,

services are managed and coordinated by deliberative BDI agents with distributed computation and coordination abilities. The agents modify their behaviour according to the users' preferences, the knowledge acquired from previous interactions, as well as the choices available to respond to a given situation.

3) **Services.** These represent the activities that the architecture offers. They are the bulk of the functionalities of the system at the processing, delivery and information acquisition levels in a ubiquitous way. Services are designed to be invoked locally or remotely. Services can be organized as local services, web services, GRID services, or even as individual stand alone services. Services can make use of other services to provide the functionalities that users require. FUSION@ has a flexible and scalable directory of services, so they can be invoked, modified, added, or eliminated dynamically and on demand. It is absolutely necessary that all services follow the communication protocol to interact with the rest of the architecture components.
4) **Communication Protocol.** This allows applications and services to communicate directly with the agent platform. The protocol is completely open and independent of any programming language, facilitating ubiquitous communication capabilities. This protocol is based on SOAP specification to capture all messages between the platform and the services and applications (Cerami, 2002). Services and applications communicate with the agent platform via SOAP messages. A response is sent back to the specific service or application that made the request. All external communications follow the same protocol, while the communication among agents in the platform follows the FIPA Agent Communication Language (ACL) specification. This is especially useful when applications run on limited processing capable devices (e.g. cell phones or PDAs). Applications can make use of agents platforms to communicate directly (using FIPA ACL specification) with the agents in

FUSION@, so while the communication protocol is not needed in all instances, it is absolutely required for all services.

This framework has been modelled following the SOA model, but adding the applications block which represents a fundamental part in Ambient Intelligence: the interaction with users. These blocks provide all the functionalities of the architecture. FUSION@ adds new features to common agent frameworks, such as JADE, OAA and RETSINA and improves the services provided by these previous architectures. JADE implements a "container" philosophy. Each instance of the JADE run-time is called container. Each agent has a container (Container Agent) and there is a Main Container which encloses all containers. Communication between agents is done through FIPA ACL (Bellifemine, *et al*., 1999). In OAA, the communication and cooperation between agents are negotiated by "facilitators" using ICL (Interagent Communication Language) (Martin, *et al*., 1999). Facilitators manage all requests from users and agents. RETSINA implements a Hierarchical Task Network (HTN) which is a planning methodology that creates plans (i.e. sequence of actions) by task decomposition. RETSINA includes a Communicator for managing and communicating with the agents using KQML (Knowledge Query and Manipulation Language) (Sycara, *et al*., 2003). These previous architectures have limited communication abilities and are not compatible with SOA architectures.

One of the advantages of FUSION@ is that the users can access the system through distributed applications, which run on different types of devices and interfaces (e.g. computers, cell phones, PDAs). All requests and responses are handled by the agents in the platform. The agents analyze all requests and invoke the specified services either locally or remotely. Services process the requests and execute the specified tasks. Then, services send back a response with the result of the specific task.

A simple case can better demonstrate the basic functioning of FUSION@ when it is requesting a service. *A user needs to calculate the sum of two numbers*

and wants to do it through a mobile device (e.g. PDA) remotely connected to the system. The user executes a mathematical toolkit which provides him with a large set of formulas from which he selects the sum function, introduces a set of values, and clicks a button to get the result. When the user clicks the button, the application sends a request to the platform to find a service that can process that request. The agents invoke the appropriate service and send the request. The service processes the request and sends the result back to the agents which in turn send it to the application. It is obvious that invoking a remote service to execute a sum is not the best choice. But imagine a large scale process that uses complex AI techniques, such as genetic algorithms, data mining, neural networks, etc. where the limited processing capacity of the mobile device makes it impossible to calculate. In this case, the service may be in a powerful computer and could be remotely invoked by the mobile device.

The Web Services Architecture model uses an external directory (Universal Description, Discovery and Integration) to list all available services. Each service must send a WSDL file to the UDDI to be added to the directory. Applications consult the UDDI to find a specific service. Once the service is located, the application can start communication directly with the selected service. However, FUSION@ does not include a service discovery mechanism, so applications must use only the services listed in the platform. In addition, all communication is handled by the platform, so there is no way to interact directly between applications and services. Moreover, the platform makes use of deliberative agents to select the optimal option to perform a task, so users do not need to find and specify the service to be invoked by the application. These features have been introduced in FUSION@ to create a secure communication between applications and services. They also facilitate the inclusion of new services that users can make use regarding their location and application.

FUSION@ is a modular multi-agent architecture, where services and applications are managed and controlled by deliberative BDI agents. There are different kinds of agents in the architecture, each one with specific roles,

capabilities and characteristics. This fact facilitates the flexibility of the architecture in incorporating new agents. However, there are pre-defined agents which provide the basic functionalities of the architecture:

- **CommApp Agent.** This agent is responsible for all communications between applications and the platform. It manages the incoming requests from the applications to be processed by services. It also manages responses from services (via the platform) to applications. CommApp Agent is always on "listening mode". Applications send XML messages to the agent requesting a service and then the agent creates a new thread to start communication by using sockets. The agent sends all requests to the Admin Agent which processes the request. The socket remains open until a response to the specific request is sent back to the application using another XML message. All messages are sent to Security Agent for their structure and syntax to be analyzed.
- **CommServ Agent.** It is responsible for all communications between services and the platform. The functionalities are similar to CommApp Agent but backwards. This agent is always on "listening mode" waiting for responses of services. Admin Agent signals to CommServ Agent which service must be invoked. Then, CommServ Agent creates a new thread with its respective socket and sends an XML message to the service. The socket remains open until the service sends back a response. All messages are sent to Security Agent for their structure and syntax to be analyzed. This agent also periodically checks the status of all services to know if they are idle, busy, or crashed.
- **Directory Agent.** It manages the list of services that can be used by the system. For security reasons (Snidaro, *et al*., 2007), the list of services is static and can only be modified manually; however, services can be added, erased or modified dynamically. The list contains the information of all trusted available services. The name and description of the service, parameters required, and the IP address of the computer where the

service is running are some of the information stored in the list of services. However, there is dynamic information that is constantly being modified: the service performance (average time to respond to requests), the number of executions, and the quality of the service. This last data is very important, as it assigns a value between 0 and 1 to all services. All new services have a quality of service (QoS) value set to 1. This value decreases when the service fails (e.g. service crashes, no service found, etc.) or has a subpar performance compared to similar past executions. QoS is increased each time the service efficiently processes the tasks assigned. Information management is especially important on Ambient Intelligence environments because the data processed is very sensitive and personal. Thus, security must be a major concern when developing AmI-based systems. For this reason FUSION@ does not implement a service discovery mechanism, requiring systems to employ only the specified services from a trusted list of services. However, agents can select the most appropriate service (or group of services) to accomplish a specific a task.

- **Supervisor Agent.** This agent supervises the correct functioning of the other agents in the system. Supervisor Agent periodically verifies the status of all agents registered in the architecture by sending ping messages. If there is no response, the Supervisor agent kills the agent and creates another instance of that agent.
- **Security Agent.** This agent analyzes the structure and syntax of all incoming and outgoing XML messages. If a message is not correct, the Security Agent informs the corresponding agent (CommApp or CommServ) that the message cannot be delivered. This agent also directs the problem to the Directory Agent, which modifies the QoS of the service where the message was sent.
- **Admin Agent.** Decides which agent must be called by taking into account the QoS and users preferences. Users can explicitly invoke a service, or can let the Admin Agent decide which service is best to

accomplish the requested task. If there are several services that can resolve the task requested by an application, the agent selects the optimal choice. An optimal choice has higher QoS and better performance. Admin Agent has a routing list to manage messages from all applications and services. This agent also checks if services are working properly. It requests the CommServ Agent to send ping messages to each service on a regular basis. If a service does not respond, CommServ informs Admin Agent, which tries to find an alternate service, and informs the Directory Agent to modify the respective QoS.

- **Interface Agent.** This kind of agent was designed to be embedded in users' applications. Interface agents communicate directly with the agents in FUSION@ so there is no need to employ the communication protocol, rather the FIPA ACL specification. The requests are sent directly to the Security Agent, which analyzes the requests and sends them to the Admin Agent. The rest of the process follows the same guidelines for calling any service. These agents must be simple enough to allow them to be executed on mobile devices, such as cell phones or PDAs. All high demand processes must be delegated to services.

FUSION@ is an open architecture that allows developers to modify the structure of the agents described before, so that agents are not defined in a static manner. Developers can add new agent types or extend the existing ones to conform to their projects needs. However, most of the agents' functionalities should be modelled as services, releasing them from tasks that could be performed by services. Services represent all functionalities that the architecture offers to users and uses itself. As previously mentioned, services can be invoked locally or remotely. All information related to services is stored into a directory which the platform uses in order to invoke them, i.e., the services. This directory is flexible and adaptable, so services can be modified, added or eliminated dynamically. Services are always on "listening mode" to receive any request from

the platform. It is necessary to establish a permanent connection with the platform using sockets. Every service must have a permanent listening port open in order to receive requests from the platform. Services are requested by users through applications, but all requests are managed by the platform, not directly by applications. This provides more control and security when requesting a service because the agents can control and validate all messages sent to the services and applications. When the platform requests a service, the CommServ Agent sends an XML message to the specific service. The message is received by the service and creates a new thread to perform the task. The new thread has an associated socket which maintains communication open to the platform until the task is finished and the result is sent back to the platform. This method provides services the capability of managing multiple and simultaneous tasks, so services must be programmed to allow multi-threading. However, there could be situations where multi-tasks will not be permitted, for instance high demanding processes where multiple executions could significantly reduce the services performance. In these cases, the Admin Agent asks the CommServ Agent to consult the status of the service, which informs the platform that it is busy and cannot accept other requests until finished. The platform must then seek another service that can handle the request, or wait for the service to be idle. To add a new service, it is necessary to manually store its information into the directory list managed by the Directory Agent. Then, CommServ Agent sends a ping message to the service. The service responds to the ping message and the service is added to the platform. A service can be virtually any program that performs a specific task and shares its resources with the platform. These programs can provide methods to access data bases, manage connections, analyze data, get information from external devices (e.g. sensors, readers, screens, etc.), publish information, or even make use of other services. Developers have are free to use any programming language. The only requirement is that they must follow the communication protocol based on transactions of XML (SOAP) messages.

In the next section, a case study is presented, where FUSION@ has helped to develop a completely functional multi-agent system aimed at improving several aspects of dependent people. This system introduces all functionalities in FUSION@.

7.5. Using FUSION@ to Improve ALZ-MAS, a Multi-Agent System for Health Care

Ambient Intelligence based systems aim to improve quality of life, offering more efficient and easy ways to use services and communication tools to interact with other people, systems and environments.

Among the general population, those most likely to benefit from the development of these systems are the elderly and dependent persons, whose daily lives, with particular regard to health care, will be most enhanced (Corchado, *et al*., 2008) (van Woerden, 2006). Dependence is a permanent situation in which a person needs important assistance from others in order to perform basic daily life activities such as essential mobility, object and people recognition, and domestic tasks (Costa-Font, *et al*., 2005). Dependent persons can suffer from degenerative diseases, dementia, or loss of cognitive ability (Costa-Font, *et al*., 2005). In Spain, dependency is classified into three levels (Costa-Font, *et al*., 2005): Level 1 (moderate dependence) refers to all people that need help to perform one or several basic daily life activities, at least once a day; Level 2 (severe dependence) consists of people who need help to perform several daily life activities two or three times a day, but who do not require the support of a permanent caregiver; and finally Level 3 (great dependence) refers

to all people who need support to perform several daily life activities numerous times a day and, because of their total loss of mental or physical autonomy, need the continuous and permanent presence of a caregiver.

There is an ever growing need to supply constant care and support to the disabled and elderly, and the drive to find more effective ways of providing such care has become a major challenge for the scientific community (Nealon, *et al*., 2003). The World Health Organization has determined that in the year 2025 there will be 1 billion people in the world over the age of 60 and twice as many by 2050, with nearly 80% concentrated in developed countries (WHO, 2007). Spain will be the third "oldest country" in the world, just behind Japan and Korea, with 35% of its citizens over 65 years of age (Sancho, *et al*., 2002). In fact, people over 60 years old represent more than 21% of the European population (WHO, 2007), and people over 65 are the fastest growing segment of the population in the United States of America (Anderson, 1999). Furthermore, over 20% of those people over 85 have a limited capacity for independent living, requiring continuous monitoring and daily assistance (Erickson, *et al*., 1995). The importance of developing new and more reliable ways of providing care and support for the elderly is underscored by this trend, and the creation of secure, unobtrusive and adaptable environments for monitoring and optimizing health care will become vital. Some authors (Nealon, *et al*., 2003) consider that tomorrow's health care institutions will be equipped with intelligent systems capable of interacting with humans. Multi-agent systems and architectures based on intelligent devices have recently been explored as supervision systems for medical care for dependent people. These intelligent systems aim to support patients in all aspects of daily life, predicting potential hazardous situations and delivering physical and cognitive support.

Agents and multi-agent systems in dependency environments are becoming a reality, especially in health care. Most agents-based applications are related to the use of this technology in the monitoring of patients, treatment supervision and data mining. Lanzola *et al*. (1999) present a methodology that facilitates the

development of interoperable intelligent software agents for medical applications, and propose a generic computational model for implementing them. The model may be specialized in order to support all the different information and knowledge-related requirements of a hospital information system. Meunier (1999) proposes the use of virtual machines to support mobile software agents by using a functional programming paradigm. This virtual machine provides the application developer with a rich and robust platform upon which to develop distributed mobile agent applications, specifically when targeting distributed medical information and distributed image processing. While an interesting proposal, it is not viable due to the security reasons that affect mobile agents, and there is no defined alternative for locating patients or generating planning strategies. There are also agents-based systems that help patients to get the best possible treatment, and that remind the patient about follow-up tests (Miksch, 1999). They assist the patient in managing continuing ambulatory conditions (chronic problems). They also provide health-related information by allowing the patient to interact with the on-line health care information network. Decker and Li (1998) propose a system to increase hospital efficiency by using global planning and scheduling techniques. They propose a multi-agent solution that uses the generalized partial global planning approach which preserves the existing human organization and authority structures, while providing better system-level performance (increased hospital unit throughput and decreased inpatient length of stay time). To do this, they use resource constraint scheduling to extend the proposed planning method with a coordination mechanism that handles mutually exclusive resource relationships. Other applications focus on home scenarios to provide assistance to elderly and dependent persons. RoboCare presents a multi-agent approach that covers several research areas, such as intelligent agents, visualization tools, robotics, and data analysis techniques to support people with their daily life activities (Pecora, *et al*., 2007). TeleCARE is another application that makes use of mobile agents and a generic platform in order to provide remote services and automate

an entire home scenario for elderly people (Camarinha-Matos, *et al*., 2007). Although these applications expand the possibilities and stimulate research efforts to enhance the assistance and health care provided to elderly and dependent persons, none of them integrate intelligent agents, distributed and dynamic applications and services approach, or the use of reasoning and planning mechanisms into their model.

Given that developing novel solutions for dependent people is one of the priorities of AmI, FUSION@ has been employed to develop ALZ-MAS, a multi-agent system aimed at enhancing the assistance and health care for Alzheimer patients living in geriatric residences. The main functionalities in this system include planning mechanisms for scheduling the activities of the medical staff. The ALZ-MAS structure has five different deliberative agents based on the deliberative BDI model, each one with specific roles and capabilities:

- **Patient Agent.** It manages the patients' personal data and behaviour (monitoring, location, daily tasks, and anomalies).
- **Mobile Agent.** It runs on mobile devices and executes the nurse and doctor roles. This agent corresponds to an Interface Agent in FUSION@.
- **Nurse Agent.** It schedules the nurses' daily activities and obtains dynamic plans depending on the tasks needed for each patient. It manages scheduled-users profiles (preferences, habits, holidays, etc.), tasks, available time and resources.
- **Manager Agent.** This agent plays two roles: the security role that monitors the users' location and physical building status (temperature, lights, alarms, etc.); and the manager role that handles the databases and the task assignment.

Two versions of ALZ-MAS (ALZ-MAS 1.0 and ALZ-MAS 2.0) have been developed in parallel with the goal of obtaining an appropriate test bench to check the efficiency and robustness of FUSION@. The entire ALZ-MAS 2.0 structure has been modified according to FUSION@ model, separating most of

the agents' functionalities from those to be modelled as services. However, all functionalities are the same in both approaches, since we have considered it appropriated to compare the performance of both systems to prove the efficiency of FUSION@.

The main problem with ALZ-MAS 1.0 is the efficiency of the computational planning processes as well as the integration with other existing architectures. In ALZ-MAS 2.0, the main functionalities have been modelled as distributed and independent services by means of FUSION@, releasing the agents from high demanding computational processes. In ALZ-MAS 1.0, each agent integrates its own functionalities into their structure. If an agent needs to perform a task which involves another agent, it must communicate with that agent to request it. So, if the agent is disengaged, all its functionalities will be unavailable to the rest of agents. This is an important issue in ALZ-MAS 1.0, since agents running on mobile devices (i.e. PDAs) are constantly disconnecting from the platform and consequently crashing, making it necessary to restart (killing and launching new instances) those agents. Another important issue is that the planning mechanisms are integrated into the agents. These mechanisms are busy almost all the time, overloading the respective agents. Because these mechanisms are the core of the system, they must be available at all times. The system depends on these mechanisms to generate all decisions, so it is essential that they have all processing power available in order to increase overall performance. In addition, the use of planning mechanisms into deliberative BDI agents makes these agents complex and unable to be executed on mobile devices. In ALZ-MAS 2.0, these mechanisms have been modelled as services, so any agent can make use of them.

The agents in ALZ-MAS make use of RFID, Wi-Fi networks and ZigBee automation devices. These technologies provide automatic and real time information about the environment, and allow the users to interact with their surroundings, controlling and managing physical services (i.e. heating, lights, switches, etc.). All hardware is some way integrated to agents, providing

information about the environment that is processed by the agents to automate tasks and manage multiple services.

Next, the results obtained after applying a distributed approach in ALZ-MAS by means of FUSION@ are presented.

7.6. Results and Conclusions

FUSION@ facilitates the development of Ambient Intelligence based multi-agent systems. Its model is based on a services-oriented approach, by formalizing services, applications, communications and deliberative agents. The architecture proposes an alternative where agents act as controllers and coordinators. FUSION@ exploits the agents' characteristics to provide a robust, flexible, modular and adaptable solution that can cover most requirements of a wide diversity of Ambient Intelligence projects. All functionalities, including those of the agents, are modelled as distributed services and applications.

Initially, FUSION@ was mainly designed to develop Ambient Intelligence based systems. This architecture has been employed to develop ALZ-MAS, a system aimed at enhancing assistance and health care for Alzheimer patients in geriatric residences. Two versions of this system have been developed (ALZ-MAS and ALZ-MAS 2.0), both with the same functionalities. However, the performance of ALZ-MAS 2.0 has been highly improved, mainly because the functionalities have been modelled distributed services and applications.

Several tests have been done to compare the overall performance of ALZ-MAS 1.0 and ALZ-MAS 2.0, the latter making use of FUSION@. The tests consisted of a set of requests delivered to the planning mechanism which in turn had to

generate paths for each set of tasks (i.e. nurses' activities). The results obtained demonstrate that ALZ-MAS 1.0 was unable to handle multiple simultaneous agendas (i.e. sets of tasks). However, ALZ-MAS 2.0 had 5 replicated services available, so the workflow was distributed and allowed the system to complete the plans for multiple simultaneous agendas. Another important data is that although ALZ-MAS 1.0 performed slightly faster when processing a single agenda, performance was constantly reduced when new simultaneous agendas were added. This fact demonstrates that the overall performance of ALZ-MAS 2.0 is better when handling distributed and simultaneous tasks (e.g. agendas), instead of single tasks. In addition, ALZ-MAS 1.0 is far more unstable than ALZ-MAS 2.0, especially when comparing the number of crashed agents during these tests. These data demonstrate that a distributed approach provides a higher ability to recover from errors.

As a conclusion we can say that although FUSION@ is still under development, preliminary results demonstrate that it is adequate for building complex systems and exploiting composite services, in this case ALZ-MAS. The mechanisms introduced are just an example that shows how to deploy distributed services using FUSION@. However, services can be any functionality (mechanisms, algorithms, routines, etc.) designed and deployed by developers. FUSION@ has laid the groundwork to boost and optimize the development of future projects and systems based on Ambient Intelligence. This architecture facilitates the development of systems based on the paradigm of Ambient Intelligence because it is able to solve problems at execution time over highly dynamic and distributed environments. FUSION@ makes it easier for developers to integrate independent services and applications because they are not restricted to programming languages supported by the agent frameworks used (e.g. JADE, OAA, RETSINA). The distributed approach of FUSION@ optimizes usability and performance because it can be obtained lighter agents by modelling the systems' functionalities as independent services and applications outside of the agents' structure, thus these may be used in other developments.

Bibliografía

AAL. (2008). *Ambient Assisted Living*. Retrieved 02 21, 2008, from http://www.aal-europe.eu/

Aamodt, A., & Plaza, E. (1994). Case-Based Reasoning: foundational Issues, Methodological Variations, and System Approaches. *AI Communications , 7* (1), 39-59.

Aarts, E. (2005). Ambient intelligence drives open innovation. *Interactions , 12* (4), 66-68.

Aarts, E. H., & Encarnação, J. L. (2006). *True Visions: the Emergence of Ambient Intelligence.* Springer-Verlag New York, Inc.

Aarts, E., & Roovers, R. (2003a). Embedded system design issues in ambient intelligence. In T. Basten, M. Geilen, & H. d. Groot, *Ambient intelligence: Impact on Embedded System Design* (pp. 11-29). Norwell, MA: Kluwer Academic Publishers.

Aarts, E., & Roovers, R. (2003b). IC Design Challenges for Ambient Intelligence. *Design, Automation, and Test in Europe. Proceedings of the Conference on Design, Automation and Test in Europe. 1.* Washington, DC: IEEE Computer Society.

Aarts, E., Harwig, R., & Schuurmans, M. (2002). Ambient intelligence. In P. J. Denning, *The invisible Future: the Seamless integration of Technology into Everyday Life* (pp. 235-250). New York, NY: McGraw-Hill.

Addlesee, M., Curwen, R., Hodges, S., Newman, J., Steggles, P., Ward, A., et al. (2001). Implementing a Sentient Computing System. *Computer , 34* (8), 50-56.

Alcatel. (2005a). *Tecnologías y actividades de estandarización para la interconexión de Home Networks, Anexo A: Descripción de tecnologías para telecontrol, Cuadernos de Tecnología, No. 2.* Fundación AUNA.

Alcatel. (2005b). *Tecnologías y actividades de estandarización para la interconexión de Home Networks, Anexo C: Descripción de tecnologías para middleware, software y pasarelas residenciales, Cuadernos de Tecnología, No. 2.* Fundación AUNA.

Alcatel. (2005c). *Tecnologías y actividades de estandarización para la interconexión de Home Networks, Cuadernos de Tecnología, No. 1.* Fundación AUNA.

ANA-MAS. (2005). *Agentes Software y Sistemas Multi-Agente: Conceptos, Arquitecturas y Aplicaciones.* (J. L. Pérez de la Cruz, & J. Pavón, Eds.) Pearson Prentice-Hall.

Anastasopoulos, M., Niebuhr, D., Bartelt, C., Koch, J., & Rausch, A. (2005). Towards a Reference Middleware Architecture for Ambient Intelligence Systems. *ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications* .

Anderson, R. N. (1999). A Method for constructing complete annual U.S. life tables. Vital Health Statistics. *National Center for Health Statistics , 2* (129), 1-28.

Angulo, C., & Tellez, R. (2004). Distributed Intelligence for smart home appliances. *Tendencias de la minería de datos en España. Red Española de Minería de Datos* .

Antoniol, G., Cimitile, A., Di Lucca, G. A., & Di Penta, M. (2004). Assessing Staffing Needs for a Software Maintenance Project through Queuing Simulation. *IEEE Transactions on Software Engineering , 30* (1), 43-58.

Aracil, R., Basañez, L., Dormingo, S., & Martínez, M. (2003). *Conclusiones de las XXIV Jornadas de Automática.* León, España.

Ardissono, L., Petrone, G., & Segnan, M. (2004). A conversational approach to the interaction with Web Services. *Computational Intelligence , 20*, 693-709.

AUML.org. (2007). *The FIPA Agent UML Web Site*. Retrieved 9 6, 2007, from http://www.auml.org

Bahadori, S., Cesta, A., Grisetti, G., Iocchi, L., Leone, R., Nardi, D., et al. (2003a). RoboCare: an Integrated robotic system for the domestic care of the elderly. *In Proceedings of workshop on Ambient Intelligence.* Pisa, Italy.

Bahadori, S., Cesta, A., Grisetti, G., Iocchi, L., Leonel, R., Nardi, D., et al. (2003b). RoboCare: Pervasive Intelligence for the Domestic Care of the Elderly. *AI*IA Magazine Special Issue* .

Bajo, J., De Luis, A., González, A., Tapia, D. I., Saavedra, A., & Corchado, J. M. (2006a). Ambient Intelligence Agent for Health Care. *In Proceedings of WUCAMI'06.* Puertollano, España: José Bravo - Universidad de Castilla la Mancha (Eds.).

Bajo, J., De Luis, A., Tapia, D. I., & Corchado, J. M. (2006d). Wireless Multi-Agent Systems based on CBR-BDI Agents: from Theory to Practice. *Proceedings of IWPAAMS'06* (pp. 85-96). Universidad de Valladolid.

Bajo, J., De Paz, J. F., Tapia, D. I., & Corchado, J. M. (2006b). Distributed Prediction of Carbon Dioxide Exchange Using CBR-BDI Agents. *International Journal of Computer Science (INFOCOMP), Special Edition* , 16-25.

Bajo, J., De Paz, Y., De Paz, J. F., Martín, Q., & Corchado, J. M. (2006c). SMas: A Shopping Mall Multiagent Systems. *Proccedings of IDEAL'06, Lecture Notes in Artificial Intelligence (LNAI). 4224*, pp. 1166-1173. Springer-Verlag.

Bajo, J., Tapia, D. I., De Luis, A., & Corchado, J. M. (2007b). Nature-Inspired Planner Agent for Health Care. *Proceedings of IWANN'07, Lecture Notes in Artificial Intelligence (LNAI)* (pp. 1090-1097). Springer-Verlag.

Bajo, J., Tapia, D. I., Rodríguez, S., & Corchado, J. M. (2007c). Planning Agent for Geriatric Residences. (J. R. Rabuñal, J. Dorado, & A. Pazos, Eds.) *Encyclopedia of Artificial Intelligence* , In press.

Bajo, J., Tapia, D. l., De Luis, A., Rodríguez, S., De Paz, J. F., & Corchado, J. M. (2007a). Hybrid Architecture for a Reasoning Planner Agent. *In Proceedings of KES 2007, Lecture Notes in Artificial Intelligence (LNAI). 4693*, pp. 461-468. Apolloni, B.; et al. (Eds.), Springer-Verlag.

Bätzold, M., Navarro, M., Julián, V., & Botti, V. (2003). Desarrollo de servicios turísticos a usuarios. *In Proceedings of the Conference of the Spanish Association for Articial Intelligence (CAEPIA 2003).* San Sebastian, Spain.

Bauer, B., & Huget, M. P. (2003). *FIPA Modeling: Agent Class Diagrams, Working Draft.* Geneva, Switzerland: Foundation for Intelligent Physical Agents.

Bauer, B., Muller, J. P., & Odell, J. (2000a). Agent UML: A formalism for specifying multiagent software systems. *In Proceedings of ICSE 2000 - Workshop on Agent-Oriented Software Engineering AOSE 2000.* Limerick, Ireland.

Bauer, B., Muller, J. P., & Odell, J. (2000b). An extension of UML by protocols for multiagent interaction. *In proceedings of ICMAS 2000.* Boston, USA.

Bellifemine, F., Poggi, A., & Rimassa, G. (1999). Jade: A FIPA-compliant agent framework. *In Proceedings of the Fourth International Conference on the*

Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'99), (pp. 97-108). London.

Bellman, R. E. (1957). *Dynamic Programming.* Princeton, NJ, USA: Princeton University Press.

Bénédet, P., & Wiedemann, F. (2004). *An Introduction to control networks based on LONWORKS® Technology. Ver. 5.11.* EBV Elektronik GmbH & Co KG.

Ben-Natan, R., & Sasson, O. (2002). *IBM Websphere Application Server: The Complete Reference.* Osborne/McGraw-Hill.

Bergenti, F., & Poggi, A. (2001). LEAP: a FIPA Platform for Handheld and Mobile Devices. *In Proceedings of the ATAL 2001 Conference.* Seattle, USA.

Blefari-Melazzi, N., Bianchi, G., Ceneri, G., Cortese, G., Davide, F., Dellas, N., et al. (2004). The Simplicity Project: Managing Complexity in a Diverse ICT World. In *Ambient Intelligence.* IOS Press.

Bonino da Silva, L. O., Ramparany, F., Dockhorn, P., Vink, P., Etter, R., & Broens, T. (2007). A Service Architecture for Context Awareness and Reaction Provisioning. *IEEE Congress on Services (Services 2007)* (pp. 25-32). IEEE Computer Society.

Bratman, M. E. (1987). *Intentions, plans and practical reason.* Cambridge, MA, USA: Harvard University Press.

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

Bravo, J., Ortega, M., & Verdejo, F. (2000). Planning in problem solving: A case study in domotics. *30th ASEE/IEEE Frontiers in Education Conference.* Kansas City, Missouri, USA.

Brenner, W., Wittig, H., & Zarnekow, R. (1998). *Intelligent software agents: Foundations and applications.* Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., & Perini, A. (2004). Tropos: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems , 8* (3), 203-236.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2002). Modeling Early Requirements in Tropos: A Transformation Based Approach. *In Revised Papers and invited Contributions From the Second international Workshop on Agent-Oriented Software Engineering II. 2222*, pp. 151-168. Lecture Notes In Computer Science (LNCS). Springer-Verlag, London.

Brooks, K. (2003). The context quintet: narrative elements applied to context awareness. *In Proceedings of the International Conference on Human Computer Interaction (HCI 2003).* Creta, Grece: Erlbaum Associates, Inc.

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

Camarinha-Matos, L., & Afsarmanesh, H. (2002). Design of a virtual community infrastructure for elderly care. *In Proceedings of PRO-VE'02 – 3rd IFIP Working Conference on Infrastructures for Virtual Enterprises.* Sesimbra, Portugal.

Camarinha-Matos, L., Rosas, J., & Oliveira, A. (2004). A mobile agents platform for telecare and teleassistance. *In Proceedings of the 1st International Workshop on Tele-Care and Collaborative Virtual Communities in Elderly Care, TELECARE 2004.* Porto, Portugal.

Carbonell, J. G. (1983). Learning by Analogy: Formulating and Generalizing Plans from Past Experience. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell

(Eds.), *Machine Learning, An Artificial Intelligence Approach* (Vol. 1, pp. 137-161). Tioga, Palo Alto, California.

Carrascosa, C. (2004). Meta-razonamiento en agentes con restricciones temporales críticas. *Tesis Doctoral* . Valencia, España: Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia.

Carretero, N., & Bermejo, A. B. (2005). *Inteligencia Ambiental.* Madrid, España: CEDITEC: Centro de Difusión de Tecnologías, Universidad Politécnica de Madrid.

Casadomo. (2005). *El portal de la domótica y el hogar digital.* Retrieved 6 12, 2007, from Casadomo Soluciones S. L.: http:// www.casadomo.com

Cavalieri, S., & Mirabella, O. (2000). Small EHS: Proposal for a Profile of the European Home System Protocol. *In Proceedings of IEEE International Conference on Industrial Technology 2000*, *2*, pp. 486-490.

Cavedon, L., & Rao, A. S. (1996). Bringing about rationality: Incorporating plans into a BDI agent architecture. In N. Y. Foo, & R. Goebel (Ed.), *In Proceedings of the 4th Pacific Rim international Conference on Artificial intelligence: Topics in Artificial intelligence. Lecture Notes In Computer Science (LNCS). 1114.* Springer-Verlag, London, UK.

CEDOM. (2006). *Asociación Española de Domótica*. Retrieved 4 7, 2007, from http://www.cedom.org

Cerami, E. (2002). *Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL* (1st ed.). O'Reilly & Associates, Inc.

Cesta, A., Bahadori, S., Cortellesa, G., Grisetti, G., & Giuliani, M. (2003). The RoboCare Project, Cognitive Systems for the Care of the Elderly. *iN Proceedings of International Conference on Aging, Disability and Independence (ICADI'03).* Washington, DC, USA.

Chavez, A., Moukas, A., & Maes, P. (1997). Challenger: a multi-agent system for distributed resource allocation. *In Proceedings of the first international conference on Autonomous agents*, (pp. 323-331). Marina del Rey, CA, USA.

Chavira, G., Nava, S., Hervás, R., Bravo, J., & Sánchez, C. (2007). Combining RFID and NFC Technologies in an AmI Conference Scenario. *In Proceedings of Mexican International Conference on Computer Science: ENC 2007.* Morelia, Michoacán, México: Sociedad Mexicana de Ciencia de la Computación.

CHD. (2007). *Comisión del Hogar Digital.* Retrieved 2 8, 2007, from GT 2: Sistemas de Control: http://www.comisionhogardigital.org/GT2_sistemascontrol.htm

Cisco Systems. (2004). *A comprehensive review of 802.11 wireless LAN security and the Cisco wireless security suite.* White Paper.

Cisco Systems. (2005). *Wireless Systems and RF Safety Issues.* White Paper.

Coleman, D., Arnold, P., Bodoff, S., C., D., Gilchrist, H., Hayes, F., et al. (1994). *Object-oriented development: The fusion method.* Upper Saddle River, NJ, USA: Prentice Hall, Inc.

Colton, S., & Steel, G. (1999). Artificial Intelligence and Scientific Creativity. *Artificial Intelligence and the Simulation of Behaviour Quarterly , 2*.

Corchado, J. M., & Laza, R. (2003). Constructing Deliberative Agents with Case-based Reasoning Technology. *International Journal of Intelligent Systems , 18* (12), 1227-1241.

Corchado, J. M., & Lees, B. (2001). A hybrid case-based model for forecasting. *Applied Artificial Intelligence , 15* (2), 105-127.

Corchado, J. M., Bajo, J., & Tapia, D. I. (2006). ALZ-MAS: Alzheimer's special care multi-agent system. In A. Moreno, U. Cortes, R. Annicchiarico, & J. Nealon

(Ed.), *In Proceedings of the workshop on Agents Applied in Health Care (ECAI 2006).* Riva del Garda, Italy: Springer.

Corchado, J. M., Bajo, J., De Paz, Y., & Tapia, D. I. (2008). Intelligent Environment for Monitoring Alzheimer Patients, Agent Technology for Health Care. *Decision Support Systems* , In Press.

Corchado, J. M., Pavón, J., Corchado, E., & Castillo, L. F. (2005). Development of CBR-BDI agents: A tourist guide application. *In Proceedings of the 7th European Conference on Case-based Reasoning 2004, Lecture Notes in Artificial Intelligence (LNAI). 3155*, pp. 547-559. Springer-Verlag.

Costa-Font, J., & Patxot, C. (2005). The design of the long-term care system in Spain: Policy and financial constraints. *Social Policy and Society , 4* (1), 11-20.

Cox, M. T., Muñoz-Avila, H., & Bergmann, R. (2006). Case-Based Planning. *Knowledge Engineering Review , 20* (3), 283-287.

De Paz, Y., Martin, Q., Bajo, J., & Tapia, D. I. (2007). Combining Improved FYDPS Neural Networks and Case-Based Planning: A Case Study. In *Innovation in Hybrid Intelligent Systems, Advances in Soft Computing* (Vol. 44, pp. 296-303). Springer.

Decker, K., & Li, J. (1998). Coordinated hospital patient scheduling. *In Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS'98)* (pp. 104-111). IEEE Computer Society.

DeLoach, S. A. (2001). Analysis and Design using MaSE and agentTool. *In Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001).* Oxford, Ohio: Miami University.

Dey, A. K., & Abowd, G. D. (2000). Towards a Better Understanding of Context and Context-Awareness. *In Proceedings of the CHI 2000 Workshop on The What, Who, Where, When, and How of Context-Awareness.* The Hague, Netherlands.

Domologic. (2000). *Home bus systems in Europe - A short overview. Standards for Home and Building Automation - Konnex.* DOMOLOGIC Home Automation GmbH.

Duc, D. N., Park, J., Lee, H., & Kim, K. (2006). Enhancing Security of EPCglobal Gen-2 RFID Tag against Traceability and Cloning. *In Proceedings of the Symposium on Cryptography and Information Security 2006, Abstracts*, (pp. 97-97). Hiroshima, Japan.

Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., & Burgelman, J. C. (2001a). *Scenarios for Ambient Intelligence in 2010. Draft Final Report.* Retrieved from Information Society Technologies Advisory Group (ISTAG): ftp://ftp.cordis.europa.eu/pub/ist/docs

Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., & Burgelman, J. C. (2001b). That's what friends are for. Ambient Intelligence (AmI) and the IS in 2010. *Congress of Innovations for an e-Society, Challenges for Technology Assessment.* Berlin, Germany.

Duch, W. (2006). Computational Creativity. *In Proceedings of the IEEE World Congress on Computational Intelligence* (pp. 1162-1169). Vancouver, Canada: IEEE Press.

Durfee, E. H., Lesser, V. R., & Corkill, D. D. (1989). Trends in Cooperative Distributed Problem Solving. *IEEE Transactions on Knowledge and Data Engineering , 1* (1), 63-83.

Echelon. (1999). *Introduction to the LonWorks System, Ver. 1.0.* Echelon Corporation, USA.

Echelon. (2005). *NES System Software, Model 12101.* Echelon Corporation, USA.

EIBA. (2000). *Técnica de Proyectos en Instalaciones con EIB, Principios Básicos.* European Installation Bus Association.

Emiliani, P. L., & Stephanidis, C. (2005). Universal access to ambient intelligence environments: opportunities and challenges for people with disabilities. *IBM Systems Journal , 4* (3), 605-619.

Ergen, S. C. (2004). *ZigBee/IEEE 802.15.4 Summary. Internal Report to Advanced Technology.* Santa Clara, CA, USA: Lab of National Semiconductor.

Erickson, P., Wilson, R., & Shannon, I. (1995). Years of Healthy Life. *Statistical Notes* (7).

Erlang, A. K. (1909). The theory of probabilities and telephone conversations. *Nyt Tidsskrift for Matematik B , 20*, 131-137.

EURESCOM. (2001). *MESSAGE: Methodology for engineering systems of software agents.* EURESCOM.

Fernández de Palencia, L., Goti, A., San Telmo, E., Ganzábal, A., Angulo, J., & Romero, S. (2001). Domótica: Comunicaciones por red eléctrica. *Revista Española de Electrónica* , 36-39.

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

Foster, D., McGregor, C., & El-Masri, S. (2006). A Survey of Agent-Based Intelligent Decision Support Systems to Support Clinical Management and Research. In G. Armano, E. Merelli, J. Denzinger, A. Martin, S. Miles, H. Tianfield, et al. (Ed.), *In Proceedings of Multi-Agent Systems for Medicine (MAS*BIOMED'05).* Utretch, Netherlands.

Franklin, S., & Graesser, A. (1996). Is it an agent, or just a program?: A taxonomy for autonomous agents. In J. P. Müller, M. Wooldridge, & N. R. Jennings (Ed.), *In Proceedings of the 3rd International Workshop on Agent Theories, Architectures and Languages. Lecture Notes In Computer Science (LNCS). 1193*, pp. 21-35. Springer-Verlag.

Freeman, M., & Bailey, C. (2005). Developing Ubiquitous Computing Applications. *In Proceedings of IADIS International Conference on Applied Computing*, (pp. 263-270). Algarve, Portugal.

Freescale. (2007). *Freescale Semiconductor, Inc.* Retrieved 4 18, 2007, from www.freescale.com/files/wireless_comm/doc/brochure/BRZIGBEETECH.pdf

Garfinkel, S., & Rosenberg, B. (2005). *RFID: Applications, security, and privacy.* Addison-Wesley Professional.

Georgeff, M., & Lansky, A. L. (1987). Reactive reasoning and planning. *In Proceedings of the 6th National Conference on Artificial Intelligence (AAAI'87).* Seattle, WA, USA.

Georgeff, M., & Rao, A. (1998). Rational software agents: from theory to practice. In N. R. Jennings, & M. J. Wooldridge (Eds.), *Agent Technology: Foundations, Applications, and Markets* (pp. 139-160). Secaucus, NJ: Springer-Verlag New York.

González, V., F., M., López, A., J., E., García, M., & Oláiz, R. (2001). Visir, a simulation software for domotics installations to improve laboratory training. *In Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference.* Reno, Nevada, USA.

González-Bedia, M. (2004). Fundamentos cognitivos para el diseño de arquitecturas de agentes planificadores en contextos dinámicos de acción. *Tesis Doctoral* . Salamanca, Salamanca, España: Universidad de Salamanca.

González-Bedia, M., & Corchado, J. M. (2002). A planning Strategy based on Variational Calculus for Deliberative Agents. *Computing and Information Systems Journal , 10*, 2-14.

Grill, T., & Ibrahim, I. K. (2005). Agent Interaction in Ambient Intelligent Environments. *In Proceedings of DPSWS'13*, *2005*, pp. 225-229. Okinawa, Japan.

Halkia, M. (2003). Privacy in AmI space: Designing for private and public worlds. *EMTEL Conference: New Media Technology and Everyday Life Network.* London, England.

Hammond, K. J. (1989). *Case-Based Planning: Viewing Planning as a Memory Task.* New York, USA: Academic Press Professional, Inc.

Haya, P. A., Montoro, G., & X., A. (2005). Un mecanismo de resolución de conflictos en entornos de Inteligencia Ambiental. *Actas del Simposio de Computación Ubicua e Inteligencia Ambiental (UCAmI2005)*, (pp. 11-18). Granada, España.

Hewlett-Packard. (2002). *Understanding Wi-Fi.* Hewlett-Packard Development Company.

Hohl, F., & Rothermel, K. (1999). A protocol preventing blackbox tests of mobile agents. *In ITG/VDE Fachtagung Kommunikation in Verteilten Systemen (KiVS'99)* (pp. 170-181). Berlin, Germany: Springer-Verlag.

Hopper, A. (2000). The Clifford Paterson Lecture, 1999 sentient computing. *Philosophical Transactions of the Royal Society , 358* (1773), 2349–2358.

IEA-DSM. (1996). *International Standards Activity in Customer/Utility Communications for Demand Side Management and Related Functions.* Chester, UK: International Energy Agency/Demand-Side Management. EA Technology.

IEEE. (2003a). *IEEE 802.11a Standard.* NY, USA: IEEE LAN/MAN Standards Committee. IEEE Computer Society.

IEEE. (2003b). *IEEE 802.11b Standard.* NY, USA: IEEE LAN/MAN Standards Committee. IEEE Computer Society.

IEEE. (2003c). *IEEE 802.11g Standard.* NY, USA: IEEE LAN/MAN Standards Committee. IEEE Computer Society.

Iglesias, C., Garijo, M., Gonzales, J. C., & Velasco, J. R. (1998). Analysis and design of multiagent systems using mas-commonkads. *In Intelligent Agents IV (ATAL'97), Lecture Notes in Artificial Intelligence (LNAI). 1365*, pp. 313-326. Springer-Verlag.

IMSERSO. (2005). *Atención a las personas en situación de dependencia en España. Libro Blanco.* Madrid, España: Ministerio de Trabajo y Asuntos Sociales. Instituto de Mayores y Servicios Sociales (IMSERSO).

ISTAG. (2003). *Ambient intelligence: from vision to reality. Technical report, Draft Version.* Brussels, Belgium: Information Society Technologies Advisory Group (ISTAG), The European Commission.

ITAA. (2004). *Radio Frequency Identification. RFID...coming of age.* Information Technology Association of America.

Iverson, W. (2004). *Real World Web Services - Integrating EBay, Google, Amazon, FedEx and more* (1st ed.). O'Reilly & Associates, Inc.

Jayaputera, G. T., Zaslavsky, A. B., & Loke, S. W. (2007). Enabling run-time composition and support for heterogeneous pervasive multi-agent systems. *Journal of Systems and Software , 80* (12), 2039-2062.

Jennings, N. R., & Wooldridge, M. (1995). Applying agent technology. *Applied Artificial Intelligence , 9* (4), 351-361.

Jennings, N. R., Sycara, K., & Wooldridge, M. (1998). A Roadmap of Agent Research and Development. (N. R. Jennings, K. Sycara, & M. Georgeff, Eds.) *Autonomous Agents and Multi-Agent Systems Journal , 1* (1), 7-38.

Jin, H. D., Leung, K. S., Wong, M. L., & Xu, Z. B. (2003). An Efficient Self-Organizing Map Designed by Genetic Algorithms for the Traveling Salesman Problem. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics , 33* (6), 877-888.

Joh, D. Y. (1997). CBR in a Changing Environment. In D. B. Leake, & E. Plaza (Ed.), *In Proceedings of the Second international Conference on Case-Based Reasoning Research and Development. Lecture Notes In Computer Science (LNCS). 1266*, pp. 53-62. Springer-Verlag.

Juan, T., & Sterling, L. (2003a). A Meta-Model for Intelligent Adaptive Multi-Agent Systems in Open Environments. *In Proceedings 2nd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'03).* Melbourne, Australia.

Juan, T., & Sterling, L. (2004). Achieving Dynamic Interfaces with Agent Concepts. *In Proceedings of the Third international Joint Conference on Autonomous Agents and Multiagent Systems. 2*, pp. 688-695. New York, USA: IEEE Computer Society, Washington, DC.

Juan, T., & Sterling, L. (2003b). The ROADMAP Meta-Model for Intelligent Adaptive Multi-Agent Systems in Open Environments. *In Proceedings of the 4th International Workshop on Agents Oriented Software Engineering (AOSE'03).* Melbourne, Australia.

Juan, T., Pearce, A., & Sterling, L. (2002b). ROADMAP: extending the gaia methodology for complex open systems. *In Proceedings of the First international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '02)* (pp. 3-10). Bologna, Italy: ACM, New York, NY.

Juan, T., Sterling, L., & Winikoff, M. (2002a). Assembling agent oriented software engineering methodologies from features. *In Proceedings of the 3rd International Workshop on AOSE, at AAMAS '02.* Bologna, Italy.

Kell, A., & Colebrook, P. (2004). *Open Systems for Homes and Buildings: Comparing LonWorks and KNX.* Watford, UK: i&i limited.

King, J. S. (2001). *An IEEE 802.11 Wireless LAN Security White Paper.* U.S. Department of Energy. USA.

Kinney, P. (2003). *ZigBee Technology: Wireless Control that Simply Works. White Paper.*

Kinny, D., & Georgeff, M. (1991). Commitment and effectiveness of situated agents. *In Proceedings of Thirteenth International Joint Conference on Artificial Intelligence (IJCAI'91)* (pp. 82-88). San Mateo, CA, USA: Morgan Kaufmann.

Klein, G. A., & Calderwood, R. (1988a). How do people use analogues to make decisions. *In Proceedings of the DARPA Case-Based Reasoning Workshop.* Morgan Kaufmann.

Klein, G. A., Whitaker, L. A., & King, J. A. (1988b). Using Analogues to Predict and Plan. *In Proceedings of the DARPA Case-Based Reasoning Workshop* (pp. 224-232). Morgan Kaufmann.

Kleindienst, J., Macek, T., Seredi, L., & Šedivý, J. (2004). Vision-enhanced multi-modal interactions in domotic environments. *In Proceedings of the 8th ERCIM Workshop "User Interfaces For All".* Viena, Austria.

KNX. (2006). *KNX Association*. Retrieved 4 2, 2007, from http://www.knx.org

Kohonen, T. (1989). *Self-organization and associative memory* (3rd ed.). NY, USA: Springer-Verlag New York, Inc.

Kolodner, J. L. (1993). *Case-based reasoning.* San Mateo, CA, USA: Morgan Kauffman.

Krechmer, K. (1998). The Principles of Open Standards. *Standards Engineering , 50* (6), 1-6.

Kruskal, W., & Wallis, W. (1952). Use of ranks in one-criterion variance analysis. *Journal of American Statistics Association*, *47* (260) 583-621.

Kuhn, R. & Mostafavi, S. M. (2008). Optimal Routing Policy. *Communications Letters, IEEE, 12* (3), 222-224.

Lalley, F. (2004). *Government information technology issues 2004, A View to the Future.* Washington, D.C., USA: Government Information Technology Executive Council (GITEC).

Langen, P. H., Wijngaards, N. J., & Brazier, F. M. (2004). Towards Designing Creative Artificial Systems. (AIEDAM, F. Brazier, & A. Duffy, Eds.) *AI EDAM (Artificial Intelligence for Engineering Design, Analysis and Manufacturing). Special Issue on Learning and Creativity in Design , 8* (4), 217-225.

Lanzola, G., Gatti, L., Falasconi, S., & Stefanelli, M. (1999). A Framework for Building Cooperative Software Agents in Medical Applications. *Artificial Intelligence in Medicine , 16* (3), 223-249.

Lapine, P., Puschini, G., Wainerman, E., Crespo, A., Ballari, T., Molina, H., et al. (2001). Una visión arquitectónica de sistema para aplicaciones en domótica. *Congreso Argentino de Ciencias de la Computación.* El Calafante, Argentina.

Leung, K. S., Jin, H. D., & Xu, Z. B. (2004). An expanding Self-organizing Neural Network for the Traveling Salesman Problem. *Neurocomputing , 62*, 267-292.

Li, Y., Ghenniwa, H., & Shen, W. (2003). Integrated Description for Agent-Oriented Web Services in e-Marketplaces. *In Proceedings of AI2003, Workshop on Business Agents & the Semantic Web*, (pp. 11-17). Halifax, NS, Canada.

Li, Y., Shen, W., & Ghenniwa, H. (2004). Agent-Based Web Services Framework and Development Environment. *Computational Intelligence , 20* (4), 678-692.

Liu, X. (2007). A Multi-Agent-Based Service-Oriented Architecture for Inter-Enterprise Cooperation System. *In Proceedings of the Second international Conference on Digital Telecommunications (ICDT'07).* IEEE Computer Society, Washington, DC.

LonMark España. (2007). *LonMark España*. Retrieved 2 5, 2007, from http://www.lonusers.es

López de Mántaras, R., & Plaza, E. (1997). Case-Based Reasoning: An overview. *AI Communications Journal , 10* (1), 21-29.

Lyytinen, K., & Yoo, Y. (2002). Issues and Challenges in Ubiquitous Computing. *Communications of the ACM , 45* (12), 63-65.

Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM , 37* (7), 30-40.

Maes, P. (1989). How To Do the Right Thing. *Connection Science Journal , 1* (3).

Mainardi, E., Banzi, S., Bonfè, M., & Beghelli, S. (2005). A low-cost Home Automation System based on Power-Line Communication Links. *In 22nd International Symposium on Automation and Robotics in Construction ISARC 2005.* Ferrara, Italy.

Manmoud, Q. H. (2001). MobiAgent: A Mobile Agent-based Approach to Wireless Information Systems. *In Proceedings of Agent-Oriented Information Systems workshop at Autonomous Agents 2001.* Montreal, Canada.

Martin, D., Cheyer, A., & Moran, D. (1999). The open agent architecture: a framework for building distributed software systems. *Applied Artificial Intelligence , 13* (1/2), 91-128.

Martín, Q., De Paz, J. F., De Paz, Y., & Pérez, E. (2007). Solving TSP with a modified kohonen network. *European Journal of Operational Research* .

Mateos, F., González, V., Poo, R., García, M., & Oláiz, R. (2001). Design and development of an automatic small-scale house for teaching domotics. *In Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference.* Reno, Nevada, USA.

Matthew, G. (2005). *802.11 Wireless Networks: The Definitive Guide.* O'Reilly & Associates, Inc.

Meadows, C. (1997). Detecting attacks on mobile agents. *In Proceedings of Foundations for Secure Mobile Code Workshop*, (pp. 64-65). Monterey, CA, USA.

Menasce, D. A. (2002). Trade-offs in designing Web clusters. *Internet Computing , 6* (5), 76-80.

Mengual, L., Bobadilla, J., & Triviño, G. (2004). A fuzzy multi-agent system for secure remote control of a mobile guard robot. *Advances in Web Intelligence, Second International Atlantic Web Intelligence Conference (AWIC 2004)*, (pp. 44-53). Cancun, Mexico.

Meunier, J. A. (1999). A Virtual Machine for a Functional Mobile Agent Architecture Supporting Distributed Medical Information. *In Proceedings of the 12th IEEE Symposium on Computer-Based Medical Systems (CBMS'99).* IEEE Computer Society, Washington, DC.

Miksch, S. (1999). Plan management in the medical domain. *AI Communications , 12* (4), 209-235.

Miksch, S., Cheng, K., & Hayes-Roth, B. (1997). An intelligent assistant for patient health care. *In Proceedings of the 1st international Conference on Autonomous Agents (AGENTS'97)* (pp. 458-465). California, USA: ACM, New York.

Miori, V., Tarrini, L., Manca, M., & Tolomeo, G. (2005). An innovative, open-standards solution for Konnex interoperability with other domotic middlewares. *Konnex Scientific Conference 2005.* Pisa, Italia.

MIT. (2004). *MIT Project Oxigen*. Retrieved 9 15, 2006, from Pervasive Human-Centered Computing. Computer Science and Artificial Intelligence Laboratory: http://oxygen.csail.mit.edu

Mitrovic, N., & Arronategui, U. (2002). Mobile Agent security using Proxy-agents and Trusted Domains. *In Proceedings of the Second International Workshop on Security of Mobile Multiagent Systems (SEMAS 2002)* (pp. 81-83). German AI Research Center (DFKI) Research Report.

Molina, A. I., Redondo, M. A., & Ortega, M. (2004). Virtual Reality for Teaching Domotics. *In Proceedings of International Conference-Applied Computing (IADIS 2004).* Lisboa, Portugal.

Mukherjee, S., Aarts, E., Roovers, R., Widdershoven, F., & Ouwerkerk, M. (2006). *Amiware: Hardware Technology Drivers of Ambient Intelligence.* (Philips Research Book Series). Springer-Verlag New York, Inc.

Nealon, J. L., & Moreno, A. (2003). *Applications of Software Agent Technology in the Health Care domain* (Vol. 212). (A. Moreno, & J. L. Nealon, Eds.) Basel, Germany: Birkhäuser Verlag AG, Whitestein series in Software Agent Technologies.

Newcomer, E. (2002). *Understanding Web Services: XML, WSDL, SOAP, and UDDI.* Addison Wesley.

Nwana, H. S., Ndumu, D. T., Lee, L. C., & Collis, J. C. (1999). ZEUS: a toolkit and approach for building distributed multi-agent systems. In O. Etzioni, J. P. Müller, & J. M. Bradshaw (Ed.), *In Proceedings of the Third Annual Conference on Autonomous Agents (AGENTS '99)* (pp. 360-361). Seattle, Washington, USA: ACM, New York, NY.

O'Donoghue, D., Loughlin, A., Bohan, A., & Keane, M. T. (2005). Geometric Analogies and Topographic Maps. *IJCAI 2005 Workshop on Computational Creativity*, (pp. 30-37). Edinburgh, Scotland.

Odell, J., & Huget, M. P. (2003). *FIPA modeling: interaction diagrams. Working Draft.* Foundation for Intelligent Physical Agents.

Odell, J., Levy, R., & Nodine, M. (2004). *FIPA modeling TC: agent class superstructure metamodel.* FIPA meeting and interim work, London, UK.

Odell, J., Van Dyke Parunak, H., & Bauer, B. (2000). Extending UML for agents. In G. Wagner, & Y. Lesperance (Ed.), *In Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence*, (pp. 3-17). Austin, TX, USA.

Odell, J., Van Dyke Parunak, H., & Bauer, B. (2001). Representing agent interaction protocols in UML. In M. J. Wooldridge, & P. Ciancarini (Ed.), *In First international Workshop, AOSE 2000 on Agent-Oriented Software Engineering* (pp. 121-140). Limerick, Ireland: Springer-Verlag New York.

OMG. (2006). *Common Object Request Broker Architecture - for embedded.* Object Management Group.

OMG. (2005). *Unified Modelling Language Specification v1.3*. Retrieved 7 9, 2006, from The Object Management Group: http://www.omg.org/uml

Ossowski, S., & García-Serrano, A. (1998). Social Co-ordination among Autonomous Problem-Solving Agents. In W. Wobcke, M. Pagnucco, & C. Zhang (Ed.), *In Proceedings of the Workshops on Commonsense Reasoning, intelligent Agents, and Distributed Artificial intelligence: Agents and Multi-Agent Systems Formalisms, Methodologies, and Applications. Lecture Notes In Computer Science. 1441*, pp. 134-148. Springer-Verlag, London.

Padgham, L., & Winikoff, M. (2002). Prometheus: a methodology for developing intelligent agents. *Proceedings of the Third International Workshop on AgentOriented Software Engineering, at AAMAS 2002*, (pp. 37-38). Bologna, Italy.

Pecora, F., & Cesta, A. (2007). Dcop for smart homes: A case study. *Computational Intelligence , 23* (4), 395-419.

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

Poole, I. (2004). What exactly is … ZigBee? *Communications Engineer , 2* (4), 44-45.

RAE. (2008). *Real Academia Española*. Retrieved 5 24, 2008, from Diccionario de la Real Academia Española: http://www.rae.es

Ramakrishnan, K. M., & Deavours, D. D. (2006). Performance Benchmarks for Passive UHF RFID Tags. *In Proceedings of the 13th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems.* Nürnberg, Germany.

Rao, A. S., & Georgeff, M. P. (1995). BDI agents: from theory to practice. *In Proceedings of the First International Conference on Multi-Agents Systems (ICMAS'95).* San Francisco, CA, USA.

Rao, A. S., & Georgeff, M. P. (1991). Modeling rational agents within a BDI-Architecture. In J. Allen, R. Fikes, & E. Sandewall (Ed.), *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning ({KR}'91)* (pp. 473-484). San Mateo, CA, USA: Morgan Kaufmann publishers, Inc.

Reynolds, F. (2006). *The Ubiquitous Web, UPnP and Smart Homes.* Cambridge, UK: Pervasive Computing Group. Nokia Research Center.

Ricci, A., Buda, C., & Zaghini, N. (2007). An agent-oriented programming model for SOA & web services. *In 5th IEEE International Conference on Industrial Informatics (INDIN'07)*, (pp. 1059-1064). Vienna, Austria.

Ricci, A., Buda, C., Zaghini, N., Natali, A., Viroli, M., & Omicini, A. (2006). simpA-WS: An Agent-Oriented Computing Technology for WS-based SOA Applications. *In proceedings of WOA'06.* Catania,Italy.

Richter, K., & Hellenschmidt, M. (2004). Interacting with the Ambience: Multimodal Interaction and Ambient Intelligence. *Position Paper to the W3C Workshop on Multimodal Interaction.* Sophia Antipolis, France.

Riesbeck, C. K., & Schank, R. C. (1989). *Inside Case-Based Reasoning.* Hillsdale, NJ, USA: Lawrence Erlbaum Associates, Inc.

Rigole, P., Holvoet, T., & Berbers, Y. (2002). Using Jini to Integrate Home Automation in a Distributed Software-System. In J. Plaice, P. G. Kropf, P. Schulthess, & J. Slonim (Ed.), *In Revised Papers From the 4th international Workshop on Distributed Communities on the Web. Lecture Notes In Computer Science. 2468*, pp. 291-304. Springer-Verlag, London.

Ritchie, G. (2006). The transformational creativity hypothesis. *New Generation Computing , 24*, 241-266.

RoboCare. (2005). *The RoboCare Project. Institute of Cognitive Sciences and Technologies*. Retrieved from http://robocare.istc.cnr.it

Ross, B. H. (1989). Some psychological results on case-based reasoning. In K. J. Hammond (Ed.), *In Proceedings of the Case-Based Reasoning Workshop* (pp. 318-323). Pensacola Beach, Florida, USA: Morgan Kaufmann.

Roth, E. (1994). The Relaxation Time Heuristic for the Initial Transient Problem in M/M/k Queueing Systems. *European Journal of Operational Research*, 72, 376- 386.

Roth, V. (1999). Mutual protection of co-operating agents. In J. Vitek, & C. D. Jensen (Ed.), *In Secure internet Programming: Security Issues For Mobile and Distributed Objects. Lecture Notes In Computer Science* (pp. 275-285). Springer-Verlag, London.

Russell, S., & Norvig, P. (1995). *Artificial Intelligence: A modern approach.* Englewood Cliffs, NJ, USA: Prentice-Hall Series in Artificial Intelligence.

Rutishauser, U., Joller, J., & Douglas, R. (2005). Control and learning of ambience by an intelligent building. *IEEE Transactions on Systems, Man and Cybernetics Part A: Systems and Humans. Special Issue on Ambient Intelligence , 35* (1), 121-132.

Sadeh, N., Gandon, F., & Kwon, O. B. (2005). *Ambient Intelligence: The MyCampus Experience.* Carnegie Mellon University.

Sancho, M., Abellán, A., Pérez, L., & Miguel, J. A. (2002). Ageing in Spain. *Second World Assembly on Ageing.* Madrid, Spain: IMSERSO.

Satyanarayan, M. (2001). Pervasive computing: Vision and challenges. *IEEE Personal Communications , 8* (4), 10-17.

Schmidt, A. (2005). Interactive Context-Aware Systems Interacting with Ambient Intelligence. In G. Riva, F. Vatalaro, F. Davide, & M. Alcañiz (Eds.), *Ambient Intelligence* (pp. 159-178). IOS Press.

Schön, B., O'Hare, G. M., Duffy, B. R., Martin, A. N., & Bradley, J. F. (2005). Agent Assistance for 3D World Navigation. *Lecture Notes in Computer Science , 3661*, 499-499.

Schroots, J. J., Fernandez-Ballesteros, R., & Rudinger, G. (1997). *Aging in Europe.* Amsterdam, Netherlands: IOS Press.

SDSU. (2005). Radio Frequency Identification. *Masters of Homeland Security* . San Diego, CA, USA: San Diego State University.

Serban, R., & Van de Riet, R. (2001). Enforcing policies with privacy guardians. *First International Workshop on Security of Mobile Multiagent Systems (SEMAS 2001). Electronic Notes in Theoretical Computer Science. 63*, pp. 140-158. Elsevier B.V.

Shadbolt, N. (2003). Ambient Intelligence. *IEEE Intelligent Systems , 18* (4), 2-3.

Shafiq, M. O., Ding, Y., & Fensel, D. (2006). Bridging Multi Agent Systems and Web Services: towards interoperability between Software Agents and Semantic Web Services. *In Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Computing Conference (EDOC'06)* (pp. 85-96). IEEE Computer Society, Washington, DC.

Shanthikumar, J. G., Shengwei, D., & Zhang, M. T. (2007). Queueing Theory for Semiconductor Manufacturing Systems: A Survey and Open Problems. *Automation Science and Engineering, IEEE Transactions , 4* (4), 513-522.

Sheu, R. Y., Czajkowski, M., & Hofmann, M. (2007). Adaptive Peer-to-Peer Agent Sensor Networks. *In Proceedings of the First International Workshop on Agent Technology for Sensor Networks (ATSN'07).* Honolulu, HI, USA.

Shirky, C. (2002). *Planning for Web Services: Obstacles and Opportunities* (1st ed.). O'Reilly & Associates, Inc.

Snidaro, L., & Foresti, G. L. (2007). Knowledge representation for ambient security. *Expert Systems , 24* (5), 321-333.

Spanoudakis, N., & Moraitis, P. (2008). An Ambient Intelligence Application Integrating Agent and Service-Oriented Technologies. *In Proceedings of the*

27th SGAI International Conference on Artificial Intelligence (AI2007). Research and Development in Intelligent Systems XXIV (pp. 393-398). Cambridge, UK: Springer London.

St. Laurent, S., Johnston, J., & Dumbill, E. (2001). *Programming Web Services with XML-RPC.* O'Reilly & Associates, Inc.

Sun Microsystems. (2000). *Applications for Mobile Information Devices. Helpful Hints for Application Developers and User Interface Designers using the Mobile Information Device Profile.* Sun Microsystems, Inc.

Sun Microsystems. (2007). *The Java ME Platform*. Retrieved 5 16, 2007, from the Most Ubiquitous Application Platform for Mobile Devices: http://java.sun.com/javame

Susperregi, L., Maurtua, I., Tubío, C., A., P. M., Segovia, I., & Sierra, B. (2004). Una arquitectura multiagente para un Laboratorio de Inteligencia Ambiental en Fabricación. *1er. Taller de Desarrollo de Sistemas Multiagente (DESMA).* Málaga, España.

Sycara, K., Paolucci, M., Van Velsen, M., & Giampapa, J. (2003). The RETSINA MAS Infrastructure. *Autonomous Agents and Multi-Agent Systems , 7* (1/2), 29-48.

Symbian. (2003). *Symbian White Paper Rev 3.0: Why is a different operating system needed?* Symbian Software Ltd.

SysML.org. (2007, September). OMG System Modelling Language (OMG SysML) V1.0. *OMG Available Specification* .

Tadj, L. (1995). Waiting in line [queuing theory]. *Potentials , 14* (5), 11-13.

Tapia, D. I., Bajo, J., Corchado, J. M., Rodríguez, S., De Paz, J. F., Sánchez, J. M., et al. (2007c). Arquitectura Multiagente para Entornos Dinámicos: Tecnología e Inteligencia Aplicadas. *Ubiquitous Computing and Ambient Intelligence,*

Congreso Español de Informática (CEDI-UCAMI 2007). Zaragoza, España: Thompson-Paraninfo.

Tapia, D. I., Bajo, J., De Paz, J. F., & Corchado, J. M. (2006b). Hybrid Multi-Agent System for Alzheimer Health Care. In R. Solange Oliveira, & A. C. Roque da Silva Filho (Ed.), *In Proceedings of HAIS 2006.* Ribeirao Preto, Brasil.

Tapia, D. I., Bajo, J., De Paz, J. F., & Corchado, J. M. (2007e). Hybrid Multi-Agent System for Intelligent Health Care. In J. Bajo, E. S. Corchado, Á. Herrero, & J. M. Corchado (Ed.), *Selected Papers of Hybrid Artificial Intelligence Systems (HAIS) 2006.* Universidad de Salamanca.

Tapia, D. I., Bajo, J., Rodríguez, S., Manzano, J. M., & M., C. J. (2007d). Hybrid Agents Based Architecture on Automated Dynamic Environments. *In Proceedings of KES-HAIS'07. Knowledge-Based Intelligent Information and Engineering Systems. Lecture Notes in Artificial Intelligence (LNAI). 4693*, pp. 453-460. Vietri sul Mare, Italy: Springer Berlin / Heidelberg.

Tapia, D. I., Bajo, J., Sánchez, J. M., & Corchado, J. M. (2007b). An Ambient Intelligence Based Multi-Agent Architecture. In A. Maña Gómez, & C. Rudolph (Ed.), *In Proceedings of the second International Conference on Ambient Intelligence developments (AmI.d '07). Developing Ambient Intelligence.* Sophia Antipolis, France: Springer-Verlag.

Tapia, D. I., Bajo, J., Sánchez, J. M., De Paz, J. F., Rodríguez, S., & Corchado, J. M. (2007g). Mecanismos inteligentes para la localización de pacientes en residencias geriátricas. *Proceedings de las primeras Jornadas Científicas sobre RFID.* Ciudad Real, España.

Tapia, D. I., Corchado, J. M., & Bajo, J. (2006a). 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)* (pp. 63-74). Segovia, España: Universidad de Valladolid.

Tapia, D. I., Cueli, J. R., García, O., Corchado, J. M., Bajo, J., & Saavedra, A. (2007f). Identificación por Radiofrecuencia: Fundamentos y Aplicaciones. *Proceedings de las primeras Jornadas Científicas sobre RFID.* Ciudad Real, España.

Tapia, D. I., De Paz, J. F., Rodríguez, S., & Corchado, J. M. (2009). Sistema multiagente para la gestión y monitorización de rutas de vigilancia. *IEEE Latin America Transactions , 6* (3), In Press.

Tapia, D. I., De Paz, Y., & Bajo, J. (2007a). Ambient Intelligence Based Architecture for Automated Dynamic Environments. In D. Borrajo, L. Castillo, & J. M. Corchado (Ed.), *Actas de la XII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA) 2007. II*, pp. 171-180. Salamanca, España: Universidad de Salamanca.

Telefónica. (2003). *Libro Blanco del Hogar Digital y las Infraestructuras Comunes de Telecomunicaciones.* Telefónica de España, S.A.

Tiwari, A. K., & Shukla, K. K. (2004). Implementation of generalized cross validation based image denoising in parallel virtual machine environment. *Digital Signal Processing , 14*, 138-157.

Troyk, P. R. (1999). Injectable Electronic Identification, Monitoring, and Stimulation Systems. *Annual Review of Biomedical Engineering , 1*, 177-209.

Tweed, C., & Quigley, G. (2000). *The design and technological feasibility of home systems for the elderly.* Belfast, UK: School of Architecture. The Queen's University of Belfast.

U.S. Robotics. (2003). *802.11g Wireless Turbo White Paper.* U.S. Robotics Corporation.

USDC. (2005). *Radio Frequency Identification. Opportunities and Challenges in Implementation.* Washington, DC, USA: U.S. Department of Commerce.

Vallée, M., Ramparany, F., & Vercouter, L. (2005). Dynamic service composition in ambient intelligence environments: a multi-agent approach. *In Proceeding of the 1st European Young Researcher Workshop on Service-Oriented Computing.* Leicester, UK.

van Woerden, K. (2006). Mainstream Developments in ICT: Why are They Important for Assistive Technology? *Technology and Disability , 18* (1), 15-18.

Vázquez, I., & López de Ipiña, D. (2005). Inteligencia Ambiental: la presencia invisible. *Revista solo programadores* (127), 16-19.

Velez, J., Lourenço, A., Pechirra, C., & Almeida, L. (2001). Broadband access technologies. *In Proceedings of the 2001 EC/NSF Workshop on Universal Accessibility of Ubiquitous Computing: Providing For the Elderly* (pp. 56-59). Alcácer do Sal, Portugal: ACM, New York, NY.

Verde, V. (2004, 10 1). Domotics systems: an emerging market. *Construction Magazine* .

Verhaegh, W., Aarts, E., & Korst, J. (2006). *Intelligent Algorithms in Ambient and Biomedical Computing.* (Philips Research Book Series). Springer-Verlag New York, Inc.

Verichip. (2006). VeriMed Patient Identification. Verichip Corporation.

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 (DIS'06)* (pp. 157-162). IEEE Computer Society, Washington, DC.

W3C. (2004). *Web Services Architecture. W3C Working Group Note 11, February 2004.* The World Wide Web Consortium.

Walton, C. A. (1983). *Patent No. 4384288.* Los Gatos, CA, USA.

Walton, C. (2006). *Agency and the Semantic Web.* Oxford University Press, Inc.

Walton, C. (2004). Model Checking Multi-Agent Web Services. *In Proceedings of AAAI Spring Symposium on Semantic Web Services.* Stanford, CA, USA.

Want, R. (2006). An introduction to RFID technology. *IEEE Pervasive Computing , 1* (5), 25- 33.

Watson, I., & Marir, F. (1994). Case-Based Reasoning: A Review. *The knowledge Engineering Review , 9* (3).

Weber, W., Rabaey, J. M., & Aarts, E. (2005). *Ambient Intelligence.* Springer-Verlag New York, Inc.

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM. Special issue on computer augmented environments: back to the real world , 36* (7), 75-84.

Weiser, M. (1995). The computer for the 21st century. In R. M. Baecker, J. Grudin, W. A. Buxton, & S. Greenberg (Eds.), *Human-computer interaction: toward the year 2000* (pp. 933-940). San Francisco, CA, USA: Morgan Kaufmann Publishers, Inc.

Weyns, D., Steegmans, E., & Holvoet, T. (2004). Towards active perception for situated multi-agent systems. *Applied Artificial Intelligence , 18* (8-9), 867-883.

WHO. (2007). *Global Age-friendly Cities: A Guide.* World Health Organization.

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems.* Chichester, England: John Wiley & Sons.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review , 10* (2), 115-152.

Wooldridge, M., Jennings, N. R., & Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems , 3* (3), 285-312.

WS-I. (2007, 10 24). *Basic Profile Version 1.2. Working Group Approval Draft.* Retrieved from Web Services Interoperability Organization: http://www.ws-i.org/Profiles/BasicProfile-1_2(WGAD).html

X10. (1997a). *The X-10 Powerhouse Power Line Interface Model # PL513 and Two-Way Power Line Interface Model # TW523 Technical Note.* X10, Inc.

X10. (1997b). *X10.com*. Retrieved 3 6, 2006, from http://www.x10.com

Xing, G., & Lyu, M. (1999). Testing, Reliability, and Interoperability Issues in the CORBA Programming Paradigm. *Proceedings of the Sixth Asia Pacific Software Engineering Conference (APSEC)* (pp. 530-536). Washington, DC, USA: IEEE Computer Society.

Zambonelli, F., Jennings, N. R., & Wooldridge, M. (2003). Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM) , 12* (3), 317-370.

Zaslavsky, A. B. (2004). Mobile Agents: Can They Assist with Context Awareness? *Proceedings of the IEEE International Conference on Mobile Data Management 2004* (pp. 304-305). IEEE Computer Society.

ZigBee. (2006). *ZigBee Specification Document 053474r13.* ZigBee Standards Organization. ZigBee Alliance.

Apéndice

A

Estándares de Control y Automatización

A.1. X10

La tecnología X10 fue desarrollada por Pico Electronics Ltd. en Glenrothes, Escocia, en el periodo de tiempo comprendido entre 1974 y el lanzamiento de los primeros dispositivos en 1978. X10 se planteó como principal objetivo, diseñar y producir un alto volumen de productos a precios competitivos (X10, 1997b) (Tweed, *et al*., 2000). En la actualidad, Pico Electronics es parte de X10 Ltd., por lo que X10 es tanto un estándar como un fabricante de productos con esta tecnología.

Los dispositivos X10 se comunican utilizando corrientes portadoras (*PowerLine Communications*) sobre las líneas de energía convencionales, 220V/50Hz para Europa y 120V/60Hz para la mayor parte del continente Americano (Telefónica, 2003) (Tweed, *et al*., 2000). Aunque las líneas de energía tienen un gran ancho de banda, éste se ve drásticamente disminuido a un rango máximo de 1Mbps, debido al ambiente altamente ruidoso en el que están distribuidas, pero siendo suficiente para proporcionar una adecuada comunicación de datos dentro del hogar (Velez, *et al*., 2001).

Como se muestra en la **Figura A.1**, el funcionamiento de las corrientes portadoras se basa en la superposición sobre la señal de la red eléctrica de una serie de pulsos de 120KHz. Estos pulsos serán los que transporten la información, en formato digital, a todos los componentes X10 del sistema (Fernández de Palencia, *et al*., 2001). De esta forma, es posible optimizar el ancho de banda disponible en el cableado eléctrico y utilizar un espectro de frecuencias para el envío de datos que se encuentra alejado del espectro utilizado para el envío de la señal de energía. Las transmisiones X10 se sincronizan con el paso por el cero de la corriente alterna, con un retraso máximo de 100μseg. El retraso máximo entre el comienzo del envío y los pulsos de 120KHz es de 50μseg. Un 1

binario se representa por un pulso de 120KHz durante 1 milisegundo en el punto cero y se transmite tres veces para que coincida con el paso por el cero de la onda portadora, mientras que el 0 binario se representa por la ausencia de ese pulso. La señal completa X10 es de 48 bits, transmitiéndose a 50Hz o 60Hz, por lo que tarda casi un segundo el envío de dicha señal a un dispositivo (X10, 1997a) (Alcatel, 2005a).

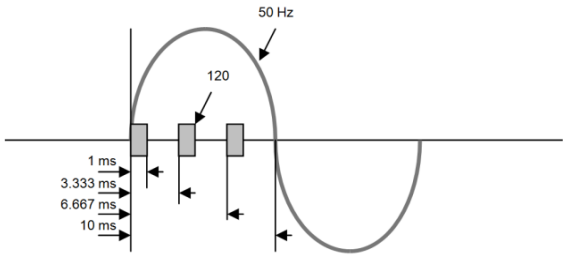

Figura A.1. Relación entre pulsos y el punto cero de la corriente alterna

Las transmisiones de datos a través de las líneas de energía presentan menos distorsiones cuando se sincronizan con el paso por cero de la señal alterna, por lo tanto, será más fácil para el receptor filtrar la señal portadora y recopilar los datos recibidos con menor probabilidad de errores (Fernández de Palencia, *et al*., 2001). En la **Figura A.2** se muestra tal como se verían las señales a través de un filtro paso-alto, con la forma de la curva de 50Hz sólo como referencia.

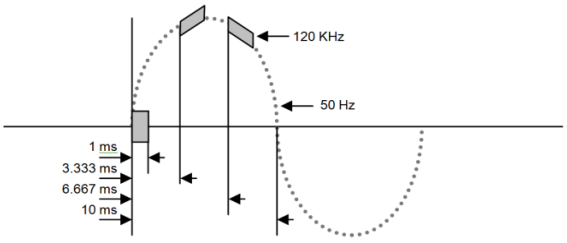

Figura A.2. Superposición de la señal con la curva de 50Hz

Las direcciones para los dispositivos se asignan a través de un grupo de dieciséis letras (A – P) llamadas "*house codes*" y dieciséis números individuales (1–16) llamados "*unit codes*", dando como resultado un total de 256 combinaciones posibles, por ejemplo: D9. El protocolo para el envío de mensajes entre los dispositivos tiene un total de seis comandos básicos llamados "*control strings*": encendido, apagado, reducir intensidad, aumentar intensidad, todo encendido y todo apagado. Estos comandos son enviados a todos los módulos, pero sólo actúa el dispositivo al que va dirigido (X10, 1997a) (Alcatel, 2005a).

En el protocolo X10 se emplea código redundante, en donde cada bit se envía dos veces, una en su verdadero valor e inmediatamente otra negado, con el fin de disminuir los posibles errores de comunicación. Esto se debe a que las redes eléctricas no son el medio más idóneo para la transmisión de datos (Fernández de Palencia, *et al*., 2001). Un código X10 requiere de once ciclos de corriente para su completa transmisión. En la **Figura A.3** se aprecia que todo el bloque se transmite dos veces, separados cada uno por tres ciclos de corriente, con excepción para las funciones de regulación de intensidad, las cuales se transmiten de manera continua sin separación entre códigos (X10, 1997a).

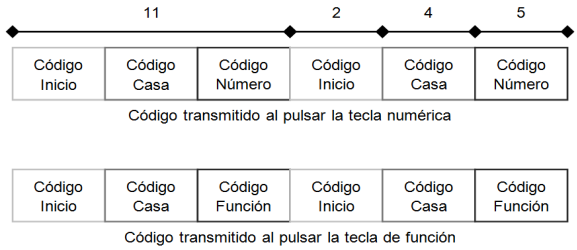

Figura A.3. Códigos de transmisión

Existen tres tipos de dispositivos X10 (X10, 1997b):

1. Transmisores. Envían las señales X10 a través de las líneas de energía.

2. Receptores. Reciben las señales X10 y las interpretan, dependiendo si la señal va dirigida hacia su dirección (código de casa + código de unidad).
3. Transmisores-Receptores. Reciben y confirman la correcta recepción de un comando.

A.2. LonWorks

LonWorks es una tecnología desarrollada por la compañía Echelon a principio de los años 90, como una solución para cualquier proyecto de automatización. Presenta una arquitectura distribuida y robusta, con especificaciones que cubren desde el nivel físico hasta el nivel de aplicación del modelo de referencia OSI (*Open Systems Interconection*) de la ISO (*International Organization for Standarization*) (Tweed, *et al*., 2000) (IEA-DSM, 1996) (Bénédet, *et al*., 2004) (Kell, *et al*., 2004). LonMark es una asociación de fabricantes que desarrollan productos o servicios basados la tecnología LonWorks, permitiendo la interoperabilidad entre sus elementos, ya sean hardware o software (LonMark España, 2007). Los cuatro elementos básicos de LonWorks son los siguientes:

1. Neuron Chip. Es el núcleo de LonWorks. Todos los dispositivos LonWorks, llamados "nodos" y mostrados en la **Figura A.4**, contienen un Neuron Chip. Este chip cuenta con un número único llamado "Neuron ID" de 48-bits, asignado desde fábrica, para identificar cada nodo dentro de una red (Alcatel, 2005a) (Bénédet, *et al*., 2004). Las funciones de los nodos (interruptores, sensores, medidores, etc.) se cargan en el firmware del Neuron Chip, programado en el lenguaje "Neuron C" basado en el estándar ANSI C (IEA-DSM, 1996) (Bénédet, *et al*., 2004) (Kell, *et al*., 2004). El Neuron Chip está compuesto por tres procesadores

en línea, dos de ellos procesan el protocolo de comunicación LonTalk y un tercero ejecuta los comandos de los nodos (Echelon, 1999) (Tweed, *et al.*, 2000).

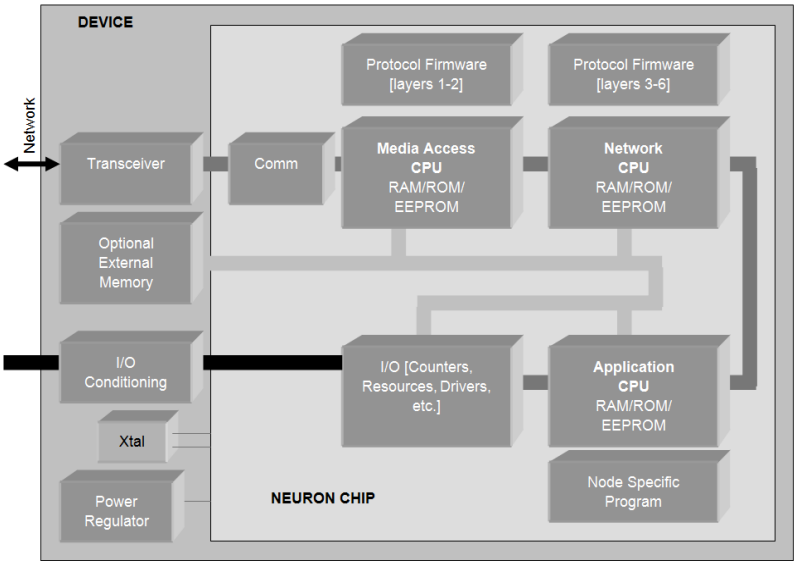

Figura A.4. Componentes de un nodo y el Neuron Chip

2. Protocolo de comunicación LonTalk. Es el protocolo propietario y cerrado que utilizan los nodos para comunicarse, proporcionando servicios de transporte y direccionamiento (Bénédet, *et al.*, 2004) (Alcatel, 2005b) (Kell, *et al.*, 2004) (Tweed, *et al.*, 2000) (Echelon, 1999). Soporta acuse de recibo extremo a extremo con reintentos automáticos y utiliza un algoritmo propietario para la detección de colisiones. LonTalk garantiza la compatibilidad entre cualquier producto de la asociación LonMark (Echelon, 1999).
3. Transceptores (transceivers). Estos dispositivos sirven de interfaz entre el Neuron Chip y el medio físico (Echelon, 1999).

4. Software. LonMaker es un paquete software que proporciona las herramientas necesarias para el diseño, instalación y mantenimiento de redes LonWorks (Echelon, 1999).

Una red LonWorks, compatible con las topologías de bus, anillo o estrella, está formada por un conjunto de nodos conectados a uno o más medios físicos llamados "canales", que pueden ser: Par trenzado, red eléctrica, radiofrecuencia (RF), cable coaxial y fibra óptica (Bénédet, *et al*., 2004) (Tweed, *et al*., 2000) (Echelon, 1999). Los nodos se comunican entre ellos utilizando el protocolo LonTalk (Bénédet, *et al*., 2004) (Echelon, 1999).

Dentro de la red, los nodos pueden actuar como (Bénédet, *et al*., 2004) (Echelon, 1999):

- ***Router*.** Tienen la finalidad de conectar dos subredes entre sí, transmitiendo paquetes entre dos canales del mismo o de diferente medio físico.
- ***Repeater*.** Retransmiten todos los paquetes entre dos canales del mismo medio físico. Básicamente son amplificadores físicos y se utilizan para largas distancias de transporte o cuando se supera el número máximo de 64 nodos en un segmento de par trenzado.
- ***Gateway*.** Conectan dos canales de diferentes dominios.
- ***Bridge*.** Permiten la transmisión de paquetes entre distintos canales. Todos los nodos de una subred han de estar en el mismo canal, o en canales conectados a bridges.

Los nodos, a través del Neuron Chip y empleando el protocolo LonTalk, reciben mensajes de otros nodos y los interpretan para realizar una determinada acción (Echelon, 1999). Los mensajes, de hasta 229 bytes generados por cada Neuron Chip, contienen la dirección de destino, el enrutamiento, los datos de control, los datos de la aplicación, y un código de detección de errores (Bénédet, *et al*., 2004).

LonWorks utiliza un direccionamiento jerárquico de tres niveles para acceder a los nodos de la red (Alcatel, 2005b) (Bénédet, *et al*., 2004) (Echelon, 1999), por lo que una dirección, almacenada en la memoria de cada nodo, está compuesta por "dominio + subred + nodo".

1. **Dominio.** Es una agrupación lógica de nodos que pertenecen a uno o más canales. Las comunicaciones sólo se pueden dar entre nodos de un mismo dominio; por tanto, un dominio constituye una red individual. El identificador de dominio tiene una longitud de 0, 1, 3, o 6 bytes. Un nodo puede ser miembro de hasta dos dominios.
2. **Subred.** Es una agrupación lógica de nodos de uno o más canales. Puede haber hasta 255 subredes por cada dominio. Los *routers* operan en este nivel.
3. **Nodo.** Puede haber hasta 127 nodos por subred, por lo que el número máximo de nodos es de 32.385 (255 x 127) en un solo dominio. Cualquier nodo puede ser miembro de uno o más dominios, permitiendo al nodo actuar como un *gateway*.

A.3. Konnex (KNX)

Konnex, también conocido como KNX, es un estándar europeo (EN 50090 - EN 13321-1) y mundial (ISO/IEC 14543) para el control de viviendas y edificios, desarrollado por la Asociación KNX (Miori, *et al*., 2005) (KNX, 2006).

Debido a la falta de un único estándar Europeo, diversos fabricantes optaron por desarrollar sus propios sistemas domóticos, surgiendo cada vez más normas y estándares incompatibles unos con otros y propiciando que el mercado domótico no creciera como se esperaba (Domologic, 2000).

La Asociación KNX, con sede en Bruselas, Bélgica, es una iniciativa de las asociaciones BatiBUS Club International (BCI), European Installation Bus Association (EIBA), European Home Systems Association (EHSA), además de desarrolladores, universidades y sociedades científicas, con el objetivo de desarrollar un único estándar domótico a partir de los estándares EIB, BatiBUS y EHS (KNX, 2006) (Domologic, 2000) (Miori, *et al*., 2005).

Los sistemas tipo bus más instalados en Europa son el BatiBUS y el EIB, que desarrollan productos similares pero con distintas topologías y medios físicos no compatibles entre sí (Alcatel, 2005c). BatiBUS se introdujo principalmente en Francia, Italia y España, mientras que EIB lo hizo en los países de lengua germana y del norte de Europa. Por su parte, EHS fue la solución preferida por los fabricantes de electrodomésticos (KNX, 2006).

Aprovechando la experiencia, capacidades y servicios de los tres estándares ya consolidados en el mercado europeo, se propuso una integración para crear una norma común (KNX) capaz de competir en calidad y prestaciones con tecnologías como LonWorks y, a su vez, aumentar la oferta de productos para el mercado doméstico, ofreciendo un conjunto de especificaciones para prácticamente cualquier aplicación domótica (Alcatel, 2005a) (KNX, 2006) (Domologic, 2000) (Miori, *et al*., 2005).

Como se aprecia en la **Figura A.5**, KNX presenta tres modos de funcionamiento, cada uno orientado a diferentes niveles de usuario final (KNX, 2006) (Alcatel, 2005a):

- **S.Mode (*System mode*).** Basado en el EIB, los dispositivos son instalados y configurados por profesionales con ayuda de aplicaciones específicamente diseñadas para este propósito.
- **E.mode (*Easy mode*).** Los dispositivos vienen programados de fábrica para realizar funciones específicas, aunque también son necesarios algunos ajustes al momento de instalarlos a través de pequeños interruptores situados dentro del dispositivo.

- **A.mode (*Automatic mode*).** Cuenta con una filosofía *Plug&Play*, en la que los usuarios no necesitan configurar los dispositivos. Está diseñado para ser utilizado en electrodomésticos, equipos de entretenimiento, aparatos electrónicos, etc.

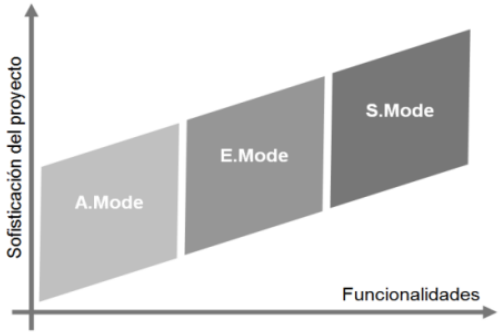

Figura A.5. Modos de funcionamiento del estándar KNX

KNX funciona sobre diversos medios físicos, integrando normas ya existentes heredadas de EIB, BatiBUS o EHS, y añadiendo algunas nuevas, proporcionando una amplia gama de soluciones para cada ámbito de aplicación, desde pequeños sistemas caseros hasta grandes proyectos industriales (KNX, 2006) (Miori, *et al*., 2005). Los medios físicos soportados por KNX se muestran en la **Tabla A.1**.

Tabla A.1. Medios físicos soportados por el estándar KNXMedios físicos soportados por el estándar KNX

Medio Físico	Frecuencia	Norma	Velocidad
TP-O (Par Trenzado, Tipo 0)	-	BatiBUS	4800 bps
TP-1 (Par Trenzado, Tipo 1)	-	EIB	9600 bps
PL-110 (Línea eléctrica)	110 KHz	EIB	1200 bps
PL-132 (Línea eléctrica)	132 KHz	EHS	2400 bps
RF (Radio Frecuencia)	868 MHz	KNX	38,4 Kbps
Ethernet	-	KNX	-

El estándar KNX es independiente de la plataforma de desarrollo, por lo que puede implementarse, a diferencia de X10 y LonWorks, sobre microprocesadores producidos por cualquier fabricante certificado, garantizando la interoperabilidad entre productos (KNX, 2006).

A continuación, se describen brevemente los estándares EIB, BatiBUS y EHS, los cuales dieron origen a KNX.

- **European Installation Bus (EIB).** es uno de los estándares más implementados en Europa, desarrollado por la EIBA (EIB Association) a principios de los años 90, con la finalidad de crear un sistema unificado para la gestión y automatización de edificios (Alcatel, 2005a) (IEA-DSM, 1996). Los principales integrantes de la EIBA, que ahora forman parte de la KNX Association, eran principalmente grandes compañías fabricantes de aparatos electrónicos y electrodomésticos, entre ellas Siemens, ABB, Temper, Grasslin, Niessen, etc. (Alcatel, 2005a) (IEA-DSM, 1996). Algunas empresas optaron por comercializar sus propias soluciones basadas en este estándar, como es el caso de Siemens con su sistema Instabus (IEA-DSM, 1996). EIB presenta una arquitectura descentralizada basada en el modelo OSI, con una comunicación entre sus componentes a través de un único bus de datos (EIBA, 2000) (Alcatel, 2005a). La estructura de bus permite unir hasta 64 dispositivos en una sola línea. Las líneas pueden unirse a través de acopladores para formar "áreas" de hasta 15 líneas, como se observa en la **Figura A.6** (Alcatel, 2005a) (EIBA, 2000).

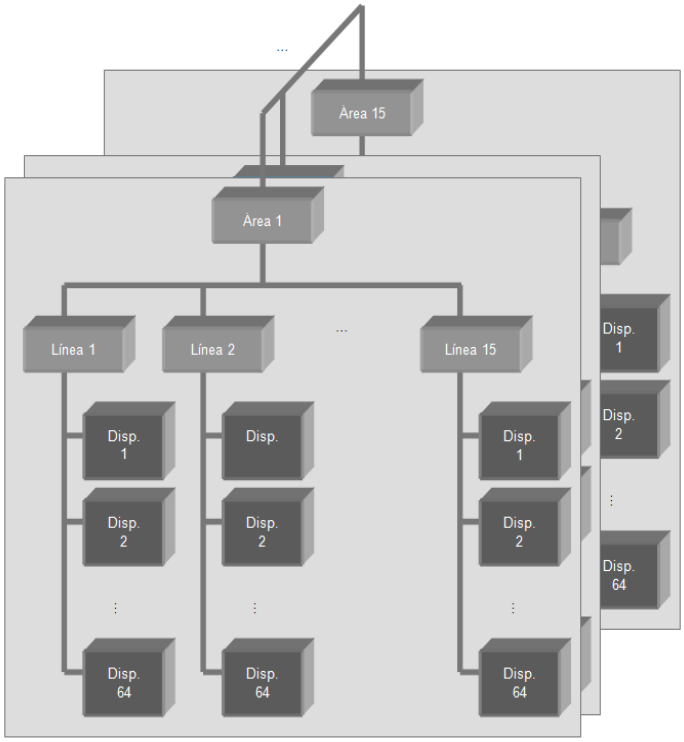

Figura A.6. Estructura de una red EIB

Una red EIB soporta las topologías de árbol, estrella y bus, y se compone básicamente de los siguientes dispositivos, los cuales se muestran en la **Figura A.7** (EIBA, 2000):

- Módulos de alimentación. Proveen de energía al sistema.
- Acopladores. Se utilizan para interconectar diferentes segmentos de red.
- Sensores. Detectan los cambios de estado (movimiento, luminosidad, temperatura, etc.), transmitiendo la información a los actuadores.
- Actuadores. Ejecutan los comandos generados por el sistema y sirven de enlace para los aparatos conectados a ellos (motores, interruptores, electrodomésticos, etc.).

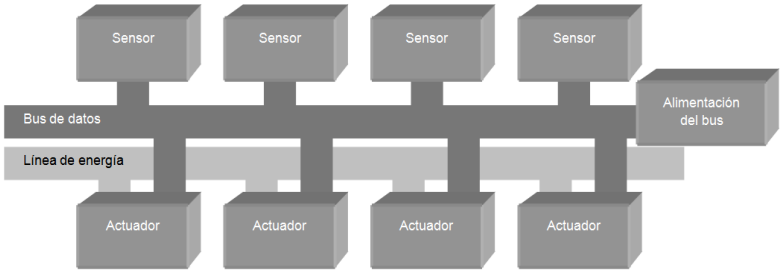

Figura A.7. Conexión de dispositivos EIB a través de un esquema tipo bus

El protocolo de comunicación es común en todos los dispositivos EIB, por lo que es posible utilizar par trenzado o la red eléctrica (corrientes portadoras) como medio físico para el bus de datos, aunque esta última es muy poco utilizada (Alcatel, 2005a) (EIBA, 2000). En par trenzado, la transmisión de datos se realiza de manera asíncrona, a 9600bps, codificando los datos en modo simétrico, de forma que un 0 (cero) se representa como un pulso de -5V a +5V, y un 1 (uno) como la ausencia de ese pulso (Alcatel, 2005a) (EIBA, 2000). EIB es un estándar abierto, tanto en especificaciones como en componentes, por lo que cualquier fabricante puede producir dispositivos con esta tecnología. Aunque muchos de los dispositivos EIB son compatibles con el nuevo estándar KNX, ya no existe soporte para EIB como tal, por lo que los fabricantes deben cumplir con las nuevas especificaciones fijadas por la asociación KNX al diseñar y producir sus dispositivos (KNX, 2006).

- **European Home System (EHS).** Surge a mediados de los años 80 como una iniciativa de la Comisión Europea, a través de los proyectos Eureka y ESPIRIT, junto con fabricantes de electrodomésticos, empresas eléctricas, operadoras de telecomunicaciones y fabricantes de equipo eléctrico, con la finalidad de desarrollar una tecnología abierta que cubriera las necesidades de automatización e interconexión de equipos eléctricos y electrónicos de distintos fabricantes y proveedores de

servicios europeos, además de lograr una mayor penetración de la domótica en el mercado residencial (Cavalieri, *et al*., 2000) (Tweed, *et al*., 2000) (IEA-DSM, 1996). Posteriormente se crearía la EHSA (*European Home System Association*) para promover el uso y desarrollo de la tecnología EHS, aunque a mediados de los años 90 dicho desarrollo se enfocó más en el mercado empresarial que en el residencial, causando que EHS entrara en competencia directa con EIB y BatiBUS (Alcatel, 2005a) (IEA-DSM, 1996). El estándar propuesto por EHS se basa en el modelo de referencia OSI, especificando las capas de aplicación, red, enlace de datos, y física, agregando una capa transversal para supervisar la comunicación en el sistema (Cavalieri, *et al*., 2000). Además, soporta las topologías de bus o estrella, sobre diversos medios físicos: red eléctrica, par trenzado, cable coaxial, radio frecuencia e infrarrojos (IEA-DSM, 1996) (Tweed, *et al*., 2000). En EHS, los dispositivos se definen como "unidades", las cuales pueden ser intercambiadas independientemente del fabricante, ya que comparten las capas físicas y de enlace de datos, y básicamente los mismos componentes: un microcontrolador, un módem y un conector al medio físico (Cavalieri, *et al*., 2000). Las capacidades y comunicación de las unidades están predefinidas como se aprecia en la **Figura A.8** y descritas a continuación (Alcatel, 2005a) (IEA-DSM, 1996):

- ***Medium Controller*** **(MC).** Dispositivo que detecta la conexión o desconexión de una unidad de red.
- ***Device Coordinator*** **(DvC).** Gestionan los dispositivos simples y los enlazan con los controladores de prestaciones.
- ***Feature Controller*** **(FC).** Administran los recursos de los dispositivos simples y complejos a través de los coordinadores.
- ***Simple Device*** **(SiDs).** Son simples unidades de comunicación que no tienen la capacidad de integrarse autónomamente con el sistema, como por ejemplo, interruptores, motores, etc.

- ***Complex Device* (CD).** Funcionan exactamente igual que los dispositivos simples, pero son capaces de integrarse autónomamente con el sistema.
- ***Routers*.** Interconectan diferentes medios.
- ***Gateways*.** Integran distintas redes.

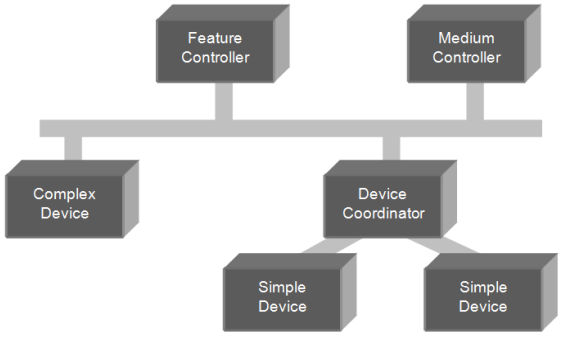

Figura A.8. Comunicación entre unidades EHS

El protocolo de comunicación presenta una filosofía *Plug&Play*, permitiendo una configuración automática del sistema y sus componentes (IEA-DSM, 1996). EHS se pensó como una tecnología de bajo coste para los hogares europeos, que superase los servicios de tecnologías como X10, pero sin llegar a la sofisticación y alto costo de LonWorks, aunque pronto se encontró con EIB y BatiBUS, los cuales fueron ganándole cada vez más terreno en el mercado (Tweed, *et al*., 2000). En la actualidad, EHSA es parte de la Asociación KNX, integrando parte de EHS en el nuevo estándar KNX.

- **BatiBUS.** Es uno de los estándares europeos más difundidos, especialmente en Francia, desarrollado inicialmente por la compañía Merlin Gerin y posteriormente por un conglomerado de empresas europeas que forman la asociación BCI (BatiBUS Club Internacional), específicamente diseñado para el mercado residencial y pequeños

edificios (Alcatel, 2005c) (Tweed, *et al*., 2000) (IEA-DSM, 1996). Se basa en parte en el modelo OSI, especificando las capas física, aplicación y enlace de datos, soportando las topologías de bus, estrella, anillo, árbol o cualquier combinación de éstas (Tweed, *et al*., 2000). BatiBUS utiliza una estructura de bus sobre par trenzado, con una velocidad máxima de 4800bps utilizando la técnica CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) con resolución positiva de las colisiones, en donde sólo el dispositivo que tiene más prioridad continua transmitiendo (Alcatel, 2005c) (Tweed, *et al*., 2000) (IEA-DSM, 1996). Los datos se envían a través del bus y todos los dispositivos conectados los reciben y procesan, pero solo actúan los de la dirección de destino (Alcatel, 2005a). Al igual que los dispositivos X-10, todos los dispositivos BatiBUS disponen de interruptores que permiten asignar una dirección física y lógica para identificar a cada dispositivo dentro de la red (Alcatel, 2005a). BatiBUS es un estándar abierto, con una instalación flexible y sencilla de configurar, pero su principal desventaja es contar con el par trenzado como único medio físico, prácticamente limitando su instalación a edificios de nueva construcción (Alcatel, 2005a).

A.4. ZigBee

ZigBee es un estándar de comunicación inalámbrica enfocado a la automatización y el control remoto de aplicaciones, desarrollado por la organización ZigBee Alliance. Esta organización está integrada por fabricantes y proveedores de tecnología, entre ellos Honeywell, Invensys, Mitsubishi, Motorola, Samsung y Philips, con la finalidad de crear un estándar global, abierto y flexible, que proporcione capacidades inalámbricas de manera sencilla,

robusta, eficiente y de bajo coste entre dispositivos cotidianos (ZigBee, 2006) (Freescale, 2007) (Kinney, 2003) (Ergen, 2004) (Poole, 2004).

El estándar ZigBee está diseñado para implementarse en aparatos electrónicos, automatización de hogares y edificios, control industrial, dispositivos computacionales, aplicaciones médicas, telecomunicaciones, juguetes, etc. (Alcatel, 2005a) (ZigBee, 2006).

Como se muestra en la **Tabla A.2**, ZigBee opera en las frecuencias 868MHz (Europa), 915MHz (Estados Unidos) y 2.4GHz (global), con distintas tasas de transferencia y sobre distintos canales, utilizando DSSS con CSMA/CA y mecanismos para detectar y evitar otro tipo de redes inalámbricas, entre ellas las Wi-Fi (ZigBee, 2006) (Kinney, 2003) (Ergen, 2004) (Poole, 2004). La distancia de transmisión entre dispositivos varía desde los 10m hasta los 75m (Poole, 2004).

Tabla A.2. Características de frecuencia y velocidad del estándar ZigBee

Frecuencia	Número de canales	Tasa de transferencia	Modulación
868MHz	1	20Kbps	BPSK
915MHz	10	40Kbps	BPSK
2.4GHz	16	250Kbps	O-QPSK

ZigBee se basa en el modelo de referencia OSI y en el estándar para redes WPAN (Wireless Personal Area Network) 802.15.4 de la IEEE, la cual especifica la capa física y la subcapa de control de acceso al medio (MAC). ZigBee Alliance define las capas superiores, englobándolas en tres capas: red (NWK), seguridad y aplicación (APL), como se puede apreciar en la **Figura A.9** (Ergen, 2004) (Alcatel, 2005a) (ZigBee, 2006).

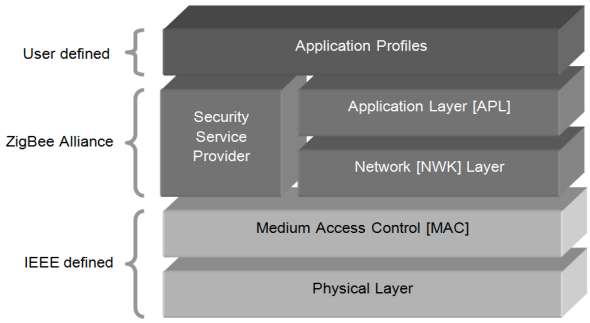

Figura A.9. Arquitectura en capas de ZigBee

Este estándar define dos tipos de dispositivos (ZigBee, 2006) (Kinney, 2003):

1. ***Full Function Device (FFD).*** Los dispositivos de funcionalidad completa implementan toda la funcionalidad de ZigBee y cuentan con memoria y cierto grado de capacidad de cómputo. Pueden implementarse sobre cualquier topología, ya que son capaces de comunicarse con cualquier otro dispositivo (FFD o RFD), así como de detectar otros dispositivos y servicios en una determinada red. Debido a las capacidades extras que implementan, sus requerimientos de energía son mayores que los RFD, por lo que generalmente se conectan a las líneas de energía convencionales. Los FFD pueden ser configurados para actuar como:
    - ***Coordinator.*** Los coordinadores inicializan y gestionan la red así como los nodos bajo su control. Asignan direcciones y almacenan la información de cada nodo. Necesitan más memoria y recursos que los demás tipos de FFD (Router o End Device).
    - ***PAN (Personal Area Network) Coordinator.*** Los coordinadores de red funcionan igual que los coordinadores, pero tienen la jerarquía más alta dentro de la red. Sólo puede haber un PAN Coordinator por cada red.

- ***Router.*** Tienen la capacidad de transmitir información entre coordinadores, dispositivos finales y routers.
- ***End Device.*** Los dispositivos finales son los extremos de la red, actuando como sensores/actuadores.

2. ***Reduced Function Device (RFD).*** Los dispositivos de funcionalidad reducida son dispositivos sencillos y de bajo coste, con capacidad, memoria y funcionalidad limitadas. No requieren demasiada energía, por lo que generalmente se alimentan de baterías. Se emplean como dispositivos finales (End Devices), destinados a transmitir y recibir información. Sólo pueden comunicarse con los FFD.

Como se aprecia en la **Figura A.10**, ZigBee soporta las topologías de estrella, árbol y malla (Kinney, 2003) (ZigBee, 2006) (Ergen, 2004).

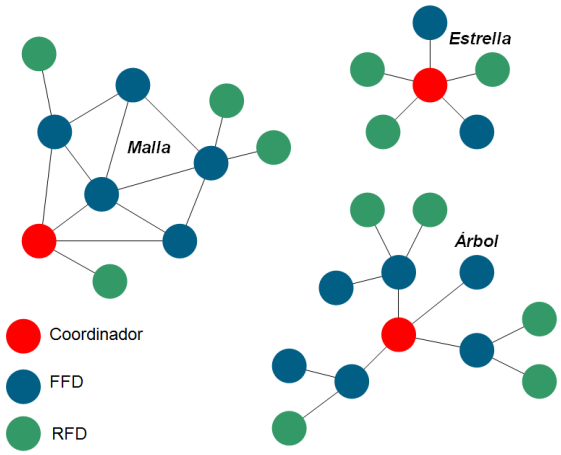

Figura A.10. Topologías soportadas por ZigBee

Estas topologías se describen a continuación:

- **Malla (*Mesh*).** Permite altos niveles de fiabilidad y escalabilidad, proporcionando múltiples rutas para la transmisión de datos con comunicación redundante y con capacidad de auto-organización y auto-curación. En esta topología, la mayoría de los dispositivos son FFD que

actúan como routers o coordinadores y existe sólo un PAN Coordinator. Se utiliza principalmente en control y monitorización industrial.

- **Estrella (*Star*).** En esta topología, la red está controlada por el PAN Coordinator que inicializa y administra a todos los dispositivos en la red, los cuales se comunican entre sí sólo a través del coordinador. Las redes en estrella se pueden comunicar entre ellas a través de sus respectivos PAN Coordinator. Esta configuración consume un mínimo energía, ya que generalmente se emplean RDFs, por lo que se utiliza principalmente en domótica, periféricos de ordenador, juguetes y juegos.
- **Árbol (*Cluster-Tree*).** En la topología de árbol, excepto los dispositivos finales (End Device) todos los dispositivos son FFD (routers o coordinadores). Los routers transportan información utilizando una estrategia de enrutamiento jerárquica. Cualquier FFD puede actuar como coordinador, sincronizando los servicios entre el resto de dispositivos. Sólo puede haber un PAN Coordinator, el cual decide qué dispositivos entran o salen de la red así como su jerarquía. Esta configuración proporciona altos niveles de fiabilidad, bajo consumo de energía y mayor área de cobertura.

Es posible conectar hasta 240 dispositivos finales a un solo coordinador, el cual asigna una dirección de red (entre 1 y 240) a cada dispositivo, pudiendo crear redes con más de 60,000 nodos (Poole, 2004) (Kinney, 2003) (Ergen, 2004). Las direcciones 0, 255 y el intervalo entre 241-254 están reservadas para identificar la interfaz de datos, para broadcast y para futuros desarrollos, respectivamente (ZigBee, 2006).

Los dispositivos ZigBee pasan la mayor parte del tiempo en modo inactivo o "dormidos", esperando para transmitir información, con un tiempo aproximado de 15msec para volver al estado activo y recibir un paquete, por lo que su consumo de energía es bajo, lo que permite que las baterías alcancen un mayor tiempo de vida (Ergen, 2004) (Freescale, 2007) (Alcatel, 2005a).

Apéndice

B

Tecnologías Inalámbricas

B.1. Wi-Fi

Las redes Wi-Fi™ (*Wireless Fidelity*) se utilizan principalmente como una extensión, o para reemplazar las tradicionales redes de cable. Estas redes proporcionan los mismos servicios de cable en cuanto a comunicaciones y aportan notables ventajas en cuanto a movilidad. Además, su coste de implementación es mucho menor, ya que presentan ahorro tanto en los componentes, como en la instalación de la infraestructura (Hewlett-Packard, 2002).

Es necesario que los dispositivos Wi-Fi cuenten con ciertos requisitos de interoperabilidad y deben ser certificados por la organización Wi-Fi Alliance, garantizando su compatibilidad con cualquier otro dispositivo certificado sin importar el fabricante (Hewlett-Packard, 2002) (U.S. Robotics, 2003). Todos ellos funcionan bajo el estándar 802.11, publicado en 1999 por el comité de estándares LAN/MAN (*Local Area Network / Metropolitan Area Network*) de la IEEE (*Institute of Electrical and Electronics Engineers*). Este estándar especifica las características que deben cumplir la capa física y la capa MAC (Media Access Control) del modelo de referencia OSI (*Open Systems Interconection*) de la ISO (*International Organization for Standarization*), descrito en la **Tabla B.1**, para redes inalámbricas de área local (WLAN). El principal objetivo de Wi-Fi consiste en proporcionar conectividad inalámbrica a dispositivos que requieren cierta movilidad dentro de una LAN (IEEE, 2003a) (King, 2001) (Hewlett-Packard, 2002).

Para la capa física, se define una tasa de transferencia de datos de hasta 2Mbps, utilizando RF (Radio Frecuencia) en la banda de los 2.4GHz, ya sea mediante el método FHSS (*Frequency Hopping Spread Spectrum*) o DSSS (*Direct Sequence Spread Spectrum*). La especificación para la capa MAC del nivel de

enlace es similar a la proporcionada por el estándar 802.3 (Ethernet) y es independiente de la capa de control lógico del enlace (LLC) definido en el 802.2. La diferencia con respecto al estándar 802.3 en la capa MAC radica en que se utiliza el protocolo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) en lugar del protocolo CSMA/CD (*Carrier Sense Multiple Access with Collision Detect*) ya que en medios inalámbricos no hay manera práctica de transmitir y recibir datos al mismo tiempo (IEEE, 2003a) (King, 2001).

Tabla B.1. Modelo de referencia OSI

Capa	Descripción
7) Aplicación (*Application*)	Identificación del receptor y tipo de comunicación con el usuario. Aplicaciones de alto nivel.
6) Presentación (*Presentation*)	Encriptación y conversión de datos.
5) Sesión (*Session*)	Inicio y finalización de sesiones.
4) Transporte (*Transport*)	Aseguramiento del envío y recepción de mensajes.
3) Red (*Network*)	Enrutamiento de datos entre redes utilizando la dirección de red.
2) Enlace de datos (*Data Link*)	Codificación, decodificación y transmisión de paquetes (*frames*) entre nodos basándose en la dirección física del dispositivo.
1) Física (*Physical*)	Envío de señales eléctricas sobre determinado medio físico, voltaje y velocidad.

A partir del estándar Wi-Fi se han derivado otras especificaciones, algunas de las principales descritas en la **Tabla B.2**. Estas mejoran diversos aspectos del estándar original, como el rendimiento, la seguridad, la confiabilidad de las transmisiones, entre otras (Hewlett-Packard, 2002) (King, 2001) (Sun Microsystems, 2000).

El 802.11a define la capa física para tasas de transferencia entre 6 y 54Mbps a 5GHz (IEEE, 2003a) (King, 2001), utilizando OFDM (*Orthogonal Frequency-Division Multiplexing*). Permite transmitir en un ambiente menos congestionado y con un mayor espectro en las bandas de los 5GHz, sin embargo, a tales

frecuencias, la absorción de la energía por objetos sólidos es mayor, dando como resultado un incremento en las pérdidas de la señal (Hewlett-Packard, 2002). Este estándar no es compatible con el 802.11b y los dispositivos son más costosos que los 802.11b y 802.11g (Sun Microsystems, 2000).

Tabla B.2. Características comparativas de los estándares 802.11a, 802.11b y 802.11g

Estándar	Banda de operación	Tipo de Modulación	Cobertura máxima	Máxima tasa de transmisión	Máximo número de canales	Coste
802.11a	5GHz	OFDM	50m	54Mbps	12	Alto
802.11b	2.4GHz	DSSS	100m	11Mbps	3	Bajo
802.11g	2.4GHz	OFDM	100m	54Mbps	3	Medio

El 802.11b es el estándar más utilizado en redes inalámbricas, con una tasa de transferencia de 11Mbps a 2.4GHz utilizando DSSS (IEEE, 2003b) (King, 2001). Sus principales desventajas son su baja tasa de transmisión y los problemas de interferencia, ya que opera en la misma banda que los dispositivos Bluetooth y los teléfonos inalámbricos, entre otros (Sun Microsystems, 2000).

El 802.11g se ha propuesto como reemplazo al 802.11b, con tasas de transferencia de hasta 54Mbps a 2.4GHz utilizando OFDM (*Orthogonal Frequency-Division Multiplexing*) (IEEE, 2003c). Ofrece un menor coste y menor pérdida de trayectoria que el 802.11a, mayor seguridad, y es compatible con el 802.11b (U.S. Robotics, 2003) (IEEE, 2003c), aunque presenta los mismos problemas de interferencia que el 802.11b (Sun Microsystems, 2000).

Los componentes principales de una WLAN son las estaciones (clientes inalámbricos) y los puntos de acceso, con un alcance de comunicación de hasta 100m. Una configuración BSS (*Basic Service Set*) está formada cuando dos o más estaciones se reconocen entre ellas y establecen una red, la cual puede configurarse de dos maneras (Hewlett-Packard, 2002):

- **Modo *Peer-to-peer* o *Ad Hoc*.** Conocido también como IBSS (*Independent Basic Service Set*), funciona exactamente igual que una LAN cableada, en donde dos o más estaciones se conectan entre ellas sin necesidad de un punto de acceso.
- **Modo Cliente/Servidor o de infraestructura.** Se establece cuando múltiples estaciones se conectan a un punto de acceso, que actúa además como un puente con la red cableada, que generalmente proporciona acceso a Internet.

Aunque las señales RF son capaces de penetrar objetos sólidos, excepto los de metal, éstas pueden verse reducidas debido a diversos factores (U.S. Robotics, 2003) (Sun Microsystems, 2000), entre ellos:

- Distancia entre los dispositivos.
- Potencia de la transmisión.
- Materiales en los edificios.
- Interferencias con otros dispositivos.
- Propagación de la señal (reflexión, absorción, pérdida de trayectoria, etc.).
- Tipo y localización de las antenas.

A frecuencias bajas, se produce una menor pérdida de energía a causa de obstrucciones y las señales pueden penetrar objetos sólidos más fácilmente, pero se requieren antenas más grandes y potentes. A frecuencias más altas, se pierde más energía pero es posible utilizar antenas más pequeñas (Hewlett-Packard, 2002) (Sun Microsystems, 2000). En la **Figura B.1** se muestran ejemplos de patrones de propagación de la señal en diferentes tipos de antenas.

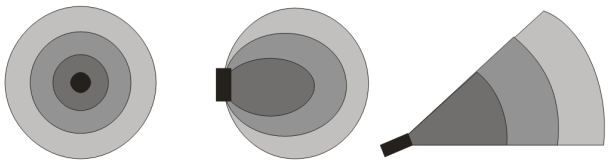

Figura B.1. Tipos de antenas. De izquierda a derecha: omnidireccional, “patch", y direccional “yagi”

El crecimiento en la implementación de redes inalámbricas presenta nuevos retos para los administradores de redes, entre ellos la seguridad (Cisco Systems, 2005) (King, 2001). Una red inalámbrica debe ser tratada como insegura, por lo que es necesario implementar mecanismos de protección de información y de autenticación de usuarios (King, 2001).

B.2. Identificación por Radiofrecuencia

La identificación por radiofrecuencia, (RFID: *Radio Frequency IDentification*) es un método utilizado para la captura automática de datos e identificación electrónica de productos, artículos, componentes, animales, incluso personas, mediante el uso de dispositivos llamados etiquetas (*tags*) (USDC, 2005). RFID proporciona una individualización a través de un único número ID (*ID number*) (Garfinkel, *et al*., 2005). Su principal uso es en la industria manufacturera, así como en el almacenamiento y distribución de productos, pero existen otros sectores en crecimiento, entre ellos los enfocados al cuidado médico, por lo que se estima una rápida expansión de RFID en los próximos años (ITAA, 2004) (SDSU, 2005).

Esta tecnología surgió en la segunda guerra mundial, sin embargo el desarrollo de sistemas RFID tal y como los conocemos en la actualidad, empezó a principios de los años setenta. En 1973, Mario Cardullo patentó en Estados Unidos la primera aproximación a la tecnología RFID pasiva, en la que los chips receptores solamente reaccionan ante la estimulación que reciben por parte de los lectores (ITAA, 2004) (USDC, 2005). En 1979, Michael Beigel había diseñado la que se considera la primera aplicación RFID de pequeño tamaño (Troyk, 1999). La primera patente norteamericana en llevar la nomenclatura “RFID” (U.S.

Patent 4.384.288) fue otorgada en 1983 a Charles Walton (Walton, 1983). Desde entonces, RFID ha ido adquiriendo una gran importancia, y se espera que tenga un papel determinante en la construcción de entornos inteligentes, especialmente basados en la Inteligencia Ambiental (Richter, *et al*., 2004).

Un sistema RFID se compone principalmente de cuatro elementos, tal y como se muestra en la **Figura B.2** (Garfinkel, *et al*., 2005) (Want, 2006):

1. **Etiquetas (*tags*).** Consisten básicamente en una antena, un pequeño chip de silicio que contiene un receptor y un transmisor de ondas de radio, un modulador para enviar señales de respuesta, lógica de control, memoria interna, y algunas de ellas un sistema de energía.
2. **Lectores.** Transmiten continuamente pulsos de energía mediante ondas de radio, los cuales son recibidos por las etiquetas. Las etiquetas detectan la energía y devuelven una señal de respuesta, que es recogida por el lector. La señal de respuesta contiene la información almacenada en el chip de las etiquetas, generalmente un número de serie.
3. **Antenas y Radios.** Conforman la capa física de RFID y se utilizan para transferir información entre los lectores y las etiquetas. El diseño de las antenas afecta en gran medida al rendimiento y comportamiento de un sistema RFID.
4. **Hardware de procesamiento.** Por lo general, es un repositorio de datos que se utiliza junto con software especializado para realizar el procesamiento de la información obtenida por los lectores.

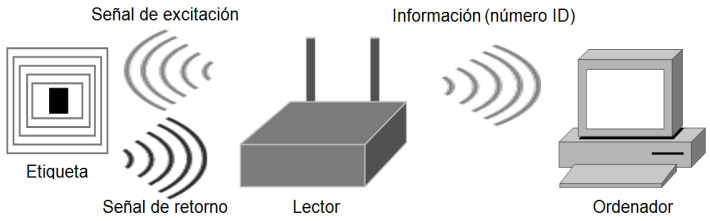

Figura B.2. Funcionamiento de RFID

Las etiquetas RFID pueden clasificarse dependiendo de varios aspectos, por ejemplo por su forma o sus características físicas (adhesivas, tarjetas, discos, pulseras, implantable, etc.), sin embargo, la forma más común de clasificarlas es por el sistema de alimentación que poseen, dividiéndose básicamente en pasivas y activas:

- **Etiquetas pasivas.** También conocidas como "*transponders*", no tienen integrado un sistema de energía, por lo que la absorben del campo electromagnético generado por los lectores RFID. La señal transmitida por los lectores proporciona la energía suficiente para alimentar el chip de la etiqueta y enviar una señal de respuesta (Lalley, 2004). Las antenas para este tipo de etiquetas deben ser diseñadas tanto para absorber la energía, como para transmitir la señal de respuesta. Dicha señal no solo es capaz de enviar un número de identificación ID, sino también información almacenada en una pequeña memoria que puede integrarse en la etiqueta. La estructura de una etiqueta pasiva se muestra en la **Figura B.3**, identificando la antena y el chip RFID. Las etiquetas pasivas tienen un corto alcance de lectura, que va desde centímetros hasta pocos metros, dependiendo de las frecuencias de transmisión y el diseño de las antenas. Este tipo de etiquetas se utilizan principalmente en la industria manufacturera y en implantes y, debido a que no cuentan con sistema de alimentación, pueden llegar a ser muy pequeñas, pudiéndose integrar en una gran cantidad de productos y dispositivos, como pegatinas, brazaletes, botones, juguetes, etc. De hecho, en febrero de 2007 la empresa Hitachi desarrolló un *transponder* llamado μ-Chip, que mide tan solo 0.05×0.05mm (sin antena) y menos de 7.5μm de grosor, con un alcance de lectura de hasta 30cm y que opera a 2.45GHz. Entre los distintos tipos de *transponders* se encuentran los implantables, en los cuales la estructura interna, mostrada en la **Figura B.3**, está compuesta por un cilindro de vidrio que contiene un microchip, una antena, y un condensador de energía. La empresa VeriChip ha desarrollado la

primera etiqueta RFID implantable para uso humano, que ha sido aprobada recientemente por la FDA (*Food and Drug Administration*) de Estados Unidos para uso médico (Verichip, 2006). La etiqueta se compone de un pequeño *transponder* cubierto por una cápsula de vidrio de 11mm de largo y 2.1mm de diámetro. Transmite en la frecuencia de los 125KHz (±6kHz) y almacena un número ID único de 15 dígitos, el cual se graba desde el proceso de manufactura y es muy difícil de alterar, ya que tan solo el proceso de de codificación utiliza 38 bits de información, lo que permite un total de 490 mil millones de posibles números ID (Lalley, 2004). Las etiquetas pasivas son más pequeñas, baratas, y tienen un mayor tiempo de uso, pero la desventaja es que el rango de lectura es mucho menor que el de las activas las cuales se describen a continuación.

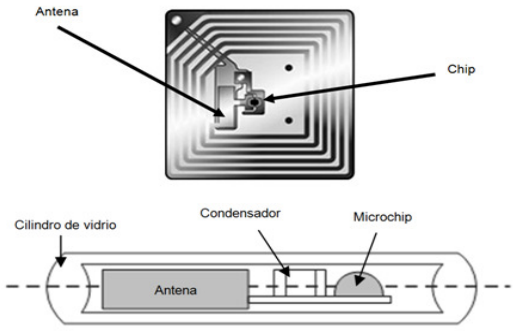

Figura B.3. Arriba: Etiqueta RFID; Abajo: *Transponder* implantable

- **Etiquetas activas.** Las etiquetas activas tienen integrado su propio sistema de alimentación, transmitiendo continuamente una señal, la cual es recogida por los lectores. Este hecho incrementa significativamente los rangos de lectura hasta en cientos de metros respecto de las etiquetas pasivas. Además, las etiquetas activas cuentan con una autonomía de las baterías de varios años. El rendimiento y fiabilidad también es mayor que los de las etiquetas pasivas, siendo más efectivas

en entornos con altos niveles de interferencias (Ramakrishnan, *et al*., 2006). Debido a la autonomía de las etiquetas activas, es posible integrar sensores (temperatura, humedad, luz, vibración, etc.) y sistemas de almacenamiento con mayores capacidades y funcionalidades, aunque se incrementa sensiblemente el tamaño de las etiquetas. El coste y tamaño de las etiquetas activas es mayor que las etiquetas pasivas (Garfinkel, *et al*., 2005), por lo que es necesario estudiar detenidamente el ámbito de aplicación para elegir la mejor opción a ser implementada. Las etiquetas activas se utilizan principalmente cuando es necesario un mayor alcance de lectura, un mayor ancho de banda. La problemática se desarrolla en entornos sensibles a interferencias, o cuando es necesaria la utilización de sensores integrados en las etiquetas.

Actualmente no existe ningún organismo internacional que regule oficialmente las frecuencias en las que operan los dispositivos RFID, por lo que cada país es responsable de normalizarlas. Aún así, existen organismos que trabajan en la estandarización y regulación de las frecuencias, entre ellos, los más importantes son: En Estados Unidos, la FCC (*Federal Communications Comisión*); en Canadá, el DOC (*Department of Communication*); en Japón, el SOUMU (*Ministry of Internal Affairs and Communications*); en China, el MII (*Ministry of Information Industry*); y en Europa, los organismos ERO (*European Radiocommunications Office*), CEPT (*Conférence Européenne des administrations des Postes et des Télécommunications*), ETSI (European Telecommunications Standards Institute), así como las administraciones propias de cada país. Además de la estandarización para el uso de las frecuencias, existen organismos que regulan el funcionamiento de RFID, entre ellos la ISO (*International Organization for Standardization*) y la EPCglobal (Duc, *et al*., 2006). Dependiendo de la frecuencia en la que operan, los sistemas RFID se clasifican en cuatro tipos: Baja frecuencia (LF: *Low Frequency*), alta frecuencia (HF: *High Frequency*), ultra alta frecuencia (UHF: *Ultra High Frequency*), y microondas. Esta clasificación se muestra en la **Tabla B.3**. Los sistemas LF y HF pueden utilizarse en todo el mundo sin requerir licencia, mientras que los sistemas UHF requieren autorización y certificación de

cada país para poder ser utilizados (Want, 2006) (Garfinkel, *et al*., 2005) (USDC, 2005). En Estados Unidos es posible utilizar sistemas UHF sin licencia, aunque con ciertas restricciones, mientras que en Europa es necesario cumplir las regulaciones EN-300-220 y EN-302-208 del ETSI y la 70-03 del ERO, además de contar con la licencia correspondiente otorgada por cada país.

Tabla B.3. Clasificación y rango de frecuencias

País/Región	LF	HF	UHF	Microondas
USA	125-134 KHz	13.56 MHz	902-928 MHz	2400-2483.5 MHz 5725-5850 MHz
Europa	125-134 KHz	13.56 MHz	865-868 MHz	2.45 GHz
Japón	125-134 KHz	13.56 MHz	No permitida	2.45 GHz
China	125-134 KHz	13.56 MHz	No permitida	2446-2454 MHz

Los sistemas de localización son uno de los usos más recientes que se han dado a RFID. Se basan en la utilización de múltiples antenas y lectores ubicados estratégicamente, con mayor o menor número de etiquetas o lectores, dependiendo del tipo de etiquetas (activas o pasivas) y del tipo de lectores (fijos o móviles). Los sistemas de localización mediante RFID generalmente utilizan etiquetas activas, debido a que el alcance de lectura es mucho mayor. Las principales configuraciones son:

- **Lectores fijos y etiquetas móviles.** Los lectores se colocan en posiciones determinas, mientras que las etiquetas se ubican en los objetos que se desean localizar. Esta configuración es la más utilizada, incluso para localizar personas.
- **Etiquetas fijas y lectores móviles.** Se coloca una serie de etiquetas en un área determinada, mientras que el lector va detectando las antenas a su paso, siguiendo las etiquetas o realizando una triangulación para saber su posición exacta. Esta configuración, se utiliza principalmente para la orientación de robots autónomos.

La localización mediante RFID se utiliza principalmente en interiores, cubriendo el hueco dejado por tecnologías como GPS (*Global Positioning System*) que se encuentran con problemas debidos a su poca precisión en entornos cerrados (Tapia, *et al*., 2007f). Un sistema de localización RFID requiere la ubicación de marcadores (etiquetas o lectores) en posiciones fijas. Mediante la medida de la intensidad de la señal enviada por los marcadores o del tiempo que tarda la señal en transmitirse, se puede determinar la distancia aproximada a los marcadores. De esta forma es posible obtener una representación, ya que en un plano, basta tener dos distancias de referencia para determinar una posición por métodos de triangulación.

Apéndice

C

Herramientas para la Ingeniería del Software Orientada a Agentes

C.1. Gaia

Gais establece una estructura de modelos relacionados entre sí y organizados en dos fases, la de análisis y la de diseño. En la fase de análisis se obtienen los modelos de roles y de interacción (entre los roles), mientras que la fase de diseño, los modelos de agentes, de servicios y de conocidos. Esta estructura de modelos se muestra en la **Figura C.1**.

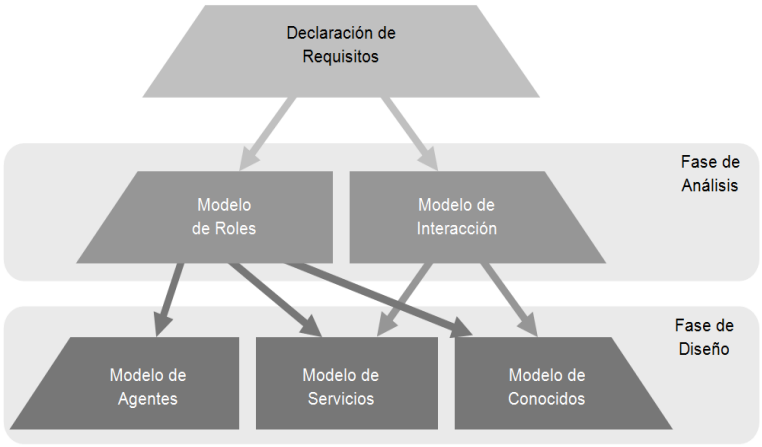

Figura C.1. Estructura de modelos en Gaia

Existen algunos desarrollos que buscan mejorar la utilidad de esta metodología, añadiendo características que extienden las capacidades básicas de Gaia, con la finalidad de obtener modelos más detallados y con mayores funcionalidades. Las principales extensiones a Gaia son las siguientes:

- **Gaia v.2.** Se basa principalmente en el concepto de organización, añadiendo tres conceptos básicos a Gaia (Zambonelli, et al., 2003):

  - Descripción del entorno. Se especifican las entidades y recursos que forman parte del entorno y que interactúan con el sistema multiagente.
  - Descripción de reglas organizacionales. Se especifican las restricciones que deben aplicarse en la organización, definiendo las condiciones en las que un agente puede participar dentro de ésta.
  - Descripción de estructuras organizacionales. Permite obtener una arquitectura global del sistema.
  - Para incluir estos conceptos, Gaia v.2 añade en la fase de análisis la identificación de las metas de la organización, un modelo del entorno, un modelo preliminar de roles, un modelo preliminar de interacción y reglas organizacionales. Además, en la fase de diseño, Gaia v.2 agrega la definición de la arquitectura global del sistema mediante una estructura organizacional.

- **ROADMAP.** Proporciona una notación específica y un meta-modelo que permite modelar sistemas multiagente abiertos y complejos (Juan, et al., 2002b) (Juan, et al., 2003b). Algunas de las principales características de esta extensión, y de los cuales carece Gaia, son: Fase de requisitos; representación de la información del entorno; representación del conocimiento del dominio; organización de los roles en jerarquías; modelo de una estructura organizacional; y mecanismos para manejo de los agentes en tiempo de ejecución. ROADMAP añade a la fase de análisis los modelos de casos de uso, del entorno y de conocimiento, que se complementan con el modelo de roles y el modelo de interacción (modelo de protocolos en ROADMAP). Estos modelos se generan de forma secuencial, siendo las salidas de cada uno, las entradas del posterior, como se muestra en la **Figura C.2**.

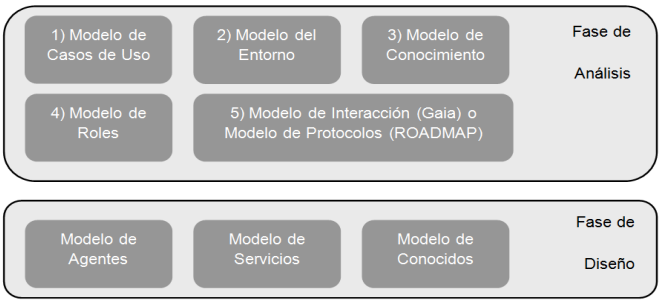

Figura C.2. Etapas en la metodología Gaia/ROADMAP

A continuación, se describen en detalle las fases de análisis y diseño de la metodología Gaia, junto con su extensión ROADMAP.

C.1.1. Análisis Gaia/ROADMAP

El objetivo de esta etapa es obtener una comprensión general del sistema y de su estructura, a través de una organización definida por un conjunto de roles y las relaciones entre ellos. En la metodología Gaia, se comienza por identificar directamente los roles a partir de una serie de requisitos. Sin embargo, al utilizar ROADMAP el proceso se inicia identificando los casos de uso y modelando el entorno del sistema multiagente, para así modelar el conocimiento utilizado en dicho sistema. Una vez obtenidos estos modelos, se genera el modelo de roles Gaia, el cual incluye conceptos de los modelos anteriores, como conocimientos y metas de los roles. Finalmente, se obtienen los modelos de protocolos y de interacciones del sistema multiagente.

Los modelos obtenidos en esta fase de la metodología Gaia/ROADMAP se describen a continuación (Wooldridge, et al., 2000) (Juan, et al., 2002b) (Juan, et al., 2003b):

- **Modelo de Casos de Uso.** Es un método adaptado de las metodologías de análisis y diseño orientadas a objetos, que incorpora diagramas generalizados y escenarios especializados, además de conceptos como zonas y conocimiento. Este modelo resulta útil para definir los requisitos del sistema.
- **Modelo del Entorno.** Este modelo permite definir las distintas zonas en el entorno y su jerarquía dentro del sistema.
- **Modelo de Conocidos.** Se basa en los modelos de casos de uso y del entorno para proporcionar una representación del dominio de conocimiento del sistema, a través de una jerarquía y descripción de componentes de conocimiento. En este modelo se intenta identificar el conocimiento del que debe disponer cada agente para llevar a cabo sus comportamientos en las zonas adecuadas y de acuerdo a cada caso de uso. Esto se logra analizando los ciclos de vida de los componentes de conocimiento, centrándose en cómo son creados, consumidos y almacenados.
- **Modelo de Roles.** En este modelo se identifican los roles claves en el sistema. En Gaia, el concepto de rol se define como una descripción abstracta de las funciones que debe realizar una entidad (Wooldridge, et al., 2000). Cada rol consta de siete atributos, los últimos tres exclusivos de ROADMAP:
    1. **Responsabilidades.** Es el principal atributo para definir un rol, y determina la funcionalidad del rol. Existen dos tipos de responsabilidades:
        - **Propiedades "*Liveness*".** Describen la funcionalidad del agente, especificando los estados que debe desarrollar cuando se den ciertas condiciones del entorno.
        - **Propiedades "*Safety*".** Son las condiciones o restricciones que el agente debe satisfacer mientras se esté ejecutando.
    2. **Permisos.** Muestran los recursos (lectura, escritura, ejecución, etc.) a los que tiene acceso el rol y la manera de acceder a ellos por parte

del agente, ya sea, para leer, modificar o generar información. En la notación para los permisos se utiliza: "*reads*", "*changes*" o "*generates*", y el nombre del recurso (Coleman, et al., 1994).

3. **Actividades.** Se describen las tareas que el agente, a través de los roles, puede realizar sin la necesidad de interactuar con otros agentes.
4. **Protocolos.** Describen la forma en la que interactúan entre sí los roles, y las actividades necesarias para ello.
5. **Sub-roles.** Define una estructura jerárquica de roles mediante esquemas que definen el lugar que ocupa cada rol dentro de la jerarquía. Esta jerarquía se representa mediante árboles, con el nodo raíz representando al sistema completo y el resto de nodos son roles atómicos y roles compuestos. El árbol de roles se genera mediante agregación y no se permite herencia.
6. **Metas.** Representan las metas que el rol debe cumplir, generalmente asociadas a protocolos o actividades. Mediante este atributo se puede evaluar el rendimiento de un agente, ya que permite observar si se cumplen las metas fijadas.
7. **Conocimiento.** Especifica los componentes de conocimiento y su jerarquía dentro del rol.

El esquema del modelo de roles, junto con los atributos descritos anteriormente, se muestra en la **Figura C.3**.

Role: NOMBRE DEL ROL
Sub roles: Lista de sub-roles
Knowledge: Conocimiento del entorno Goals: Metas a conseguir
Description: Descripción general de las actividades del rol
Protocols and Activities: Lista de protocolos y actividades que desarrolla
Permissions: Lista de recursos a los que puede acceder
Responsibilities: Liveness: Lista de funcionalidades del rol Safety: Lista de restricciones del rol

Figura C.3. Plantilla para el modelo de roles en Gaia/ROADMAP

- **Modelo de Interacción en Gaia (de Protocolos en ROADMAP).** En este modelo se representan las dependencias y relaciones de comunicación entre los roles. Se definen los protocolos (patrones de interacción) para cada una de las interacciones entre ellos y se especifica su propósito, intentando establecer un cierto orden en el intercambio de mensajes. Es necesario describir los protocolos a través de los siguientes atributos:
  - Propósito: Descripción de la interacción.
  - Iniciador: Rol o roles que comienzan la interacción.
  - Respondedor: Rol o roles con los que interactúa el iniciador.
  - Entradas: Información que proporciona el rol iniciador.
  - Salidas: Información que proporciona (o se le suministra a) el rol respondedor.
  - Procesamiento: Descripción de las acciones que se realizan durante la interacción.

En un sistema multiagente se producen una gran cantidad de interacciones entre los roles. Por lo tanto, es necesario describir los protocolos para cada una de ellas, utilizando los atributos anteriores con el esquema de la **Figura C.4**.

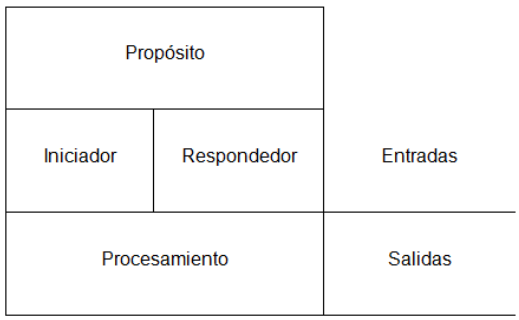

Figura C.4. Modelo de interacción Gaia, Modelo de protocolos en ROADMAP

C.1.2. Diseño Gaia

El objetivo de esta fase es reducir el nivel de abstracción obtenido en la etapa de análisis, lo suficiente para poder aplicar metodologías tradicionales, como las orientadas a objetos, y así pasar directamente al desarrollo y posterior implementación de los agentes y el sistema. Además, en esta fase se muestra en detalle la cooperación entre los agentes, así como sus necesidades, para poder alcanzar los objetivos del sistema. Como se menciona anteriormente, en el proceso de diseño se deben crear los modelos de agentes, de servicios y de conocidos, los cuales se describen a continuación.

A. **Modelo de agentes.** Se identifican los tipos de agentes y sus jerarquías en el sistema, así como las instancias definidas en la **Tabla C.1**, que aparecerán en tiempo de ejecución para cada uno de ellos dentro del sistema. El tipo de agente se define a partir de un conjunto de roles. Aunque un agente puede realizar más de un rol, por motivos de

optimización en el diseño es más común que exista una correspondencia uno a uno entre roles y tipos de agentes (Wooldridge, et al., 2000). El modelo de agentes, mostrado en la **Figura C.5**, se crea utilizando la estructura de árbol, en el cual, los nodos terminales corresponden a los roles, y el resto de los nodos con tipos de agentes presentes en el sistema.

Tabla C.1. Operadores utilizados para definir el número de instancias

Operador	Significado
n	Exactamente *n* instancias
m..n	Entre *m* y *n* instancias
*	0 o más instancias
+	1 o más instancias

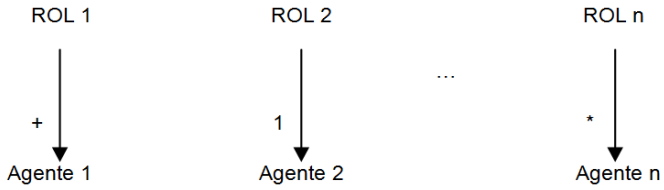

Figura C.5. Modelo de agentes en Gaia

B. Modelo de servicios. Se definen los principales servicios (funciones o tareas) que puede realizar cada agente de forma exclusiva, partiendo de las actividades, protocolos y propiedades (*liveness* y *safety*) de los roles. En general, existe un servicio asociado con cada protocolo, incluso suelen tener el mismo nombre. Para cada servicio se deben definir las propiedades (*Inputs* y *Outputs*), que representan las entradas y salidas del servicio, obtenidas del modelo de interacción, así como las condiciones (*Pre-conditions* y *Post-conditions*) que se deben cumplir al iniciar y al finalizar el servicio, derivadas de las propiedades safety del

modelo de roles. Este modelo no es clave en la metodología, por lo que generalmente se omite.

C. **Modelo de Conocidos.** Como se muestra en la **Figura C.6**, se define la comunicación entre los agentes, a partir del modelo de agentes y del modelo de interacción, descritos anteriormente, esto para prevenir congestiones en las comunicaciones que puedan causar problemas al implementar el sistema.

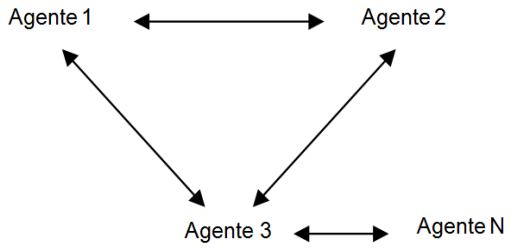

Figura C.6. Modelo de conocidos en Gaia

Los modelos obtenidos con Gaia/ROADMAP permiten obtener un esquema del sistema multiagente con un alto nivel de abstracción e independiente de cualquier plataforma de desarrollo para este tipo de sistemas. Desafortunadamente, los resultados que se obtienen quedan alejados de la implementación, por lo que es necesario detallarlos utilizando herramientas más especializadas que permitan reducir el nivel de abstracción y así acercar el modelo a la implementación del sistema. Una de estas herramientas es el lenguaje SysML, el cual se describe a continuación junto con los pasos necesarios para poder adaptar los modelos Gaia/ROADMAP y obtener los modelos propios de SysML.

C.2. SysML

SysML (*System Modeling Language*) es un robusto lenguaje de modelado para la ingeniería de sistemas basado en UML (*Unified Modeling Language*) (SysML.org, 2007) (OMG, 2005). Este lenguaje resuelve gran parte de los problemas derivados del diseño y desarrollo de sistemas multiagente, entre ellas la dependencia de las herramientas orientadas a objetos (Bauer, et al., 2000a) (AUML.org, 2007). Las capacidades de SysML lo hacen adecuado para el diseño y representación de sistemas complejos, entre ellos los sistemas multiagente.

Al igual que UML, SysML hace uso de diagramas para diseñar y representar los sistemas. Para el desarrollo de sistemas es conveniente hacer uso de todos los diagramas especificados. En la práctica resulta difícil obtener tal nivel de detalle, por lo que generalmente se utilizan solamente algunos de ellos para representar el modelo general del sistema y sus funcionalidades. A continuación se describen los principales modelos de diagramas en SysML, con los cuales es posible modelar, de forma básica, prácticamente cualquier sistema multiagente:

- **Diagramas de Definición de Bloques.** Se emplean para representar los agentes del sistema y cada uno de sus roles.
- **Diagramas de Secuencia.** Representan las interacciones entre los agentes del sistema, en particular la transmisión de mensajes. Se utilizan para definir los roles de los agentes, utilizando líneas de vida de los objetos.
- **Diagramas de Estados.** Representan los posibles estados del agente a través de grafos.
- **Diagramas de Interacción.** Al igual que los diagramas de secuencia, representan interacciones entre agentes, pero hacen énfasis en las asociaciones entre ellos.

C.2.1. Diseño SysML

En esta fase se deben definen los estados internos, los roles que desempeña y los servicios que ofrece cada agente dentro del sistema multiagente, representándolos a través de diagramas SysML.

En la fase de diseño se obtienen, los diagramas de Definición de Bloques para cada agente, los de Secuencia, los de Estados, y los de Interacción. A diferencia de Gaia, los Diagramas de Secuencia SysML presentan mayor nivel de detalle y agregan mejoras como intercambio de mensajes entre agentes y el orden en que estos se producen. Los Diagramas de Estados se utilizan para detallar más aún el modelo del sistema multiagente, representando los estados de los agentes y las actividades que ocurren en el sistema. Los diagramas que se obtienen del diseño con SysML se describen a continuación:

A. **Diagramas de Definición de Bloques.** Se utilizan para representar los agentes y sus roles dentro del sistema. SysML extiende los diagramas de clase UML (OMG, 2005), haciendo una diferencia entre agente y objeto, como se muestra en la **Figura C.7**, logrando representar los agentes junto con sus características y arquitectura (SysML.org, 2007) (AUML.org, 2007). En SysML, sea cual sea el tipo de agente (reactivo, deliberativo o híbrido), todos se definen a través de una estructura que se basa en tres características básicas:
    1. **Identificador.** Es el nombre o clave única que se le da a cada agente dentro del sistema.
    2. **Rol.** Un agente puede tener varios roles que definen su comportamiento dentro del sistema, siendo capaz de cambiar de rol en tiempo de ejecución.

3. **Organización.** Un agente puede pertenecer a una o varias organizaciones, las cuales, definen los roles de los agentes y las relaciones entre esos roles.

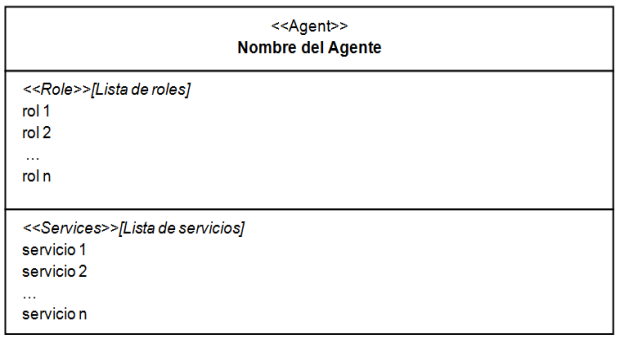

Figura C.7. Diagrama de Definición de Bloques para un agente en SysML

A esta estructura básica se le añaden clases, entre ellas, las capacidades, los servicios y los protocolos de interacción que se definieron anteriormente, utilizando la misma estructura que el diagrama anterior. Las capacidades contienen: entradas, que son los objetos que recibe el agente para realizar la capacidad; salidas, expresadas como objetos resultado de la capacidad; y restricciones de entrada y de salida, que reflejan las limitaciones a cumplirse, antes y después de realizar la capacidad. Los servicios son las actividades que puede desarrollar un agente y que puede proporcionar a otros agentes, conteniendo el nombre del servicio, la descripción en lenguaje natural del servicio, el tipo de servicio, los protocolos de interacción que soporta, los lenguajes de comunicación de agentes que utiliza, las ontologías soportadas, los lenguajes de contenido con los que es capaz de trabajar y las propiedades que lo caracterizan. Los protocolos de interacción permiten a los agentes comunicarse entre ellos, mostrando si alguno de los agentes cambia de rol o roles.

B. Diagramas de Secuencia. Muestran los patrones de comunicación de los mensajes entre agentes, así como las limitaciones en el contenido de esos mensajes. Generalmente se emplea el lenguaje ACL (*Agent Communication Language*), desarrollado por la FIPA (FIPA, 2005), para mantener una semántica consistente. Gracias a estos diagramas, se puede apreciar el comportamiento y la interacción de los agentes en el sistema, utilizando todos los diagramas anteriores distribuidos en niveles. La **Figura C.8** muestra la estructura general de un diagrama de secuencia SysML.

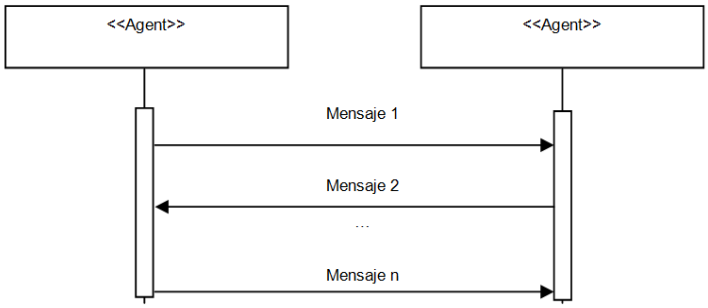

Figura C.8. Diagrama de Secuencia en SysML

C. Diagramas de Estados. Tal y como se muestra en la **Figura C.9**, se utilizan para definir el comportamiento de cada agente, mostrando los estados por los que pasan los agentes y los eventos que generan transiciones de un estado a otro en tiempo de ejecución.

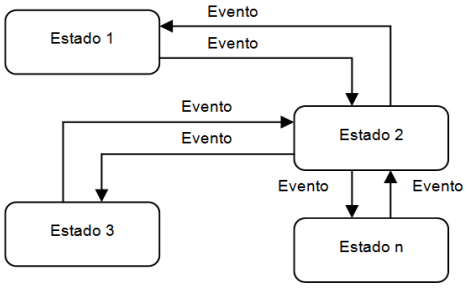

Figura C.9. Diagrama de estados en SysML

D. **Diagramas de Interacción.** Muestran las interacciones que existen entre los agentes, los roles que pueden tomar los agentes, y las interacciones entre los roles, basándose en los mensajes que intercambian los agentes y los eventos que generan a esos mensajes. Para ello, se utilizan diagramas de colaboración o diagramas de secuencia, empleando líneas continuas para mostrar las interacciones entre los agentes, y líneas discontinuas para mostrar las interacciones internas cuando un agente cambia de rol o roles. El sentido de la interacción puede ser hacia uno o ambos sentidos y se representa con flechas, agregando comentarios acerca del evento que la genera y el mensaje transmitido. La **Figura C.10** muestra la estructura general de un diagrama de interacción en SysML.

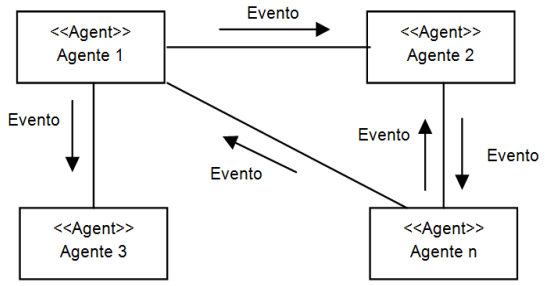

Figura C.10. Diagrama de interacción en SysML

Apéndice

D

Modelos del Análisis y Diseño Gaia

Esquema del Rol: Paciente (P)

Descripción:
Gestiona la información médica/sanitaria de los pacientes y su localización

Protocolos y Actividades:
Localización, Actualización Atributos, Actualización valores de atributos, Respuesta Consultar Atributos Paciente, Respuesta Modificar Atributos Paciente, Guardar Atributos Paciente, Solicitar cita, Respuesta Consultar Localización Paciente, Desconexión

Permisos:

Actualización	localización
	atributos
	valores de atributos

Responsabilidades:
Paciente = (Localización | Actualización Atributos | Actualización valores de atributos | Respuesta Consultar Atributos Paciente | Respuesta Modificar Atributos Paciente | Guardar Atributos Paciente | Solicitar cita | Respuesta Consultar Localización Paciente)* · Desconexión

Seguridad
Valor de atributo < su valor máximo
Valor de atributo > su valor mínimo

Figura D.1. Rol Paciente

Esquema del Rol: Enfermero (E)

Descripción:
Gestiona el plan de tareas diarias de los enfermeros del centro geriátrico y actualiza los atributos médicos de los pacientes

Protocolos y Actividades:
Conexión, Iniciar Tarea, Finalizar Tarea, Rechazar Tarea, Consultar Atributos Paciente, Modificar Atributos Paciente, Consultar Localización Paciente, Obtener Lista Pacientes, Planificar Horario, Desconexión.

Permisos:

Lectura	Localización Paciente
Actualización	Tareas
	Atributos Paciente

Responsabilidades:
Enfermero = Conexión · ((Iniciar Tarea · Finalizar Tarea) | Rechazar Tarea | Consultar Atributos Paciente · (Modificar Atributos Paciente)* | Consultar Localización Paciente | Obtener Lista Pacientes | Planificar Horario)* · Desconexión

Figura D.2. Rol Enfermero

Esquema del Rol: Médico (ME)

Descripción:
Gestiona los tratamientos de los pacientes y organiza su agenda de citas con ellos

Protocolos y Actividades:
Conexión, Crear Tratamiento, Actualizar Tratamiento, Borrar Tratamiento, Crear Cita, Actualizar Cita, Borrar Cita, Consultar Localización Paciente, Obtener Lista Pacientes, Desconexión.

Permisos:

Lectura	Localización Paciente
Actualización	Tratamientos
	Citas

Responsabilidades:
Médico = Conexión · (Crear Tratamiento | Actualizar Tratamiento | Borrar Tratamiento | Crear Cita | Actualizar Cita | Borrar Cita | Consultar Localización Paciente | Obtener Lista Pacientes)* · Desconexión

Figura D.3. Rol Médico

Esquema del Rol: Seguridad (S)

Descripción:
Monitoriza la localización de enfermeros y pacientes, y gestiona las alarmas

Protocolos y Actividades:
Localizar Paciente, Localizar Enfermero, Enviar Alerta

Permisos:

Actualización	Localización Paciente
	Localización Enfermero

Responsabilidades:
Seguridad = (Localizar Paciente · Enviar Alerta | Localizar Enfermero)^{w}

Seguridad
Paciente dentro del Centro Geriátrico

Figura D.4. Rol Seguridad

Esquema del Rol: Manager (MA)

Descripción:
Administra la base de datos y realiza el reparto de tareas entre los enfermeros

Protocolos y Actividades:
Consulta BBDD, Actualización BBDD, Reparto Tareas, Enviar Planificación Tareas, Respuesta Conexión, Respuesta Desconexión, Respuesta Crear Tratamiento, Respuesta Actualizar Tratamiento, Respuesta Borrar Tratamiento, Respuesta Crear Cita, Respuesta Actualizar Cita, Respuesta Borrar Cita, Respuesta Solicitar Cita, Respuesta Enviar Alerta, Respuesta Obtener Lista Pacientes, Enviar Lista Atributos, Enviar Lista Pacientes, Enviar Lista Citas

Permisos:

Actualización	Médicos
	Enfermeros
	Pacientes
	Atributos de Pacientes
	Localización de Pacientes
	Citas
	Tratamientos
	...

Responsabilidades:
Manager = (Consulta BBDD | Actualización BBDD | (Reparto Tareas · Enviar Planificación Tareas) | Respuesta Conexión | Respuesta Desconexión | Respuesta Crear Tratamiento | Respuesta Actualizar Tratamiento | Respuesta Borrar Tratamiento | Respuesta Crear Cita | Respuesta Actualizar Cita | Respuesta Borrar Cita | Respuesta Solicitar Cita | Respuesta Enviar Alerta | Respuesta Obtener Lista Pacientes | Enviar Lista Atributos | Enviar Lista Pacientes | Enviar Lista Citas)^{w}

Seguridad
Accesos de Personal

Figura D.5. Rol Manager

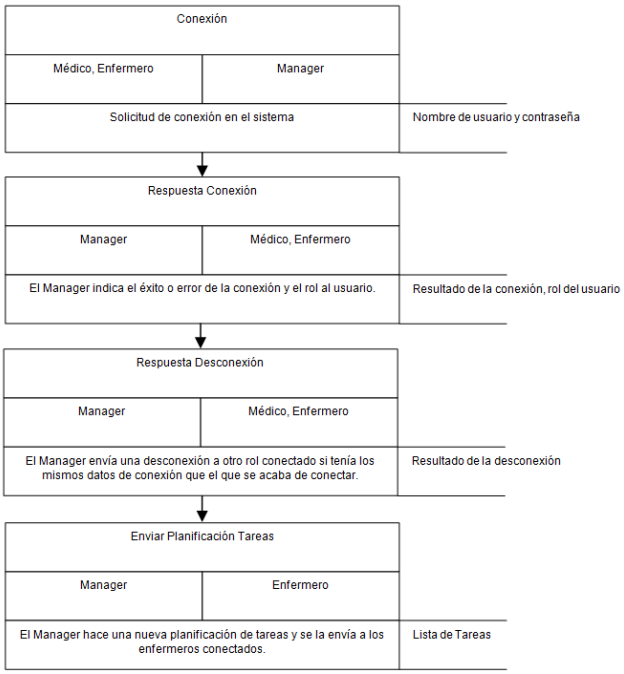

Figura D.6. Conexión en el Sistema de un Médico o Enfermero

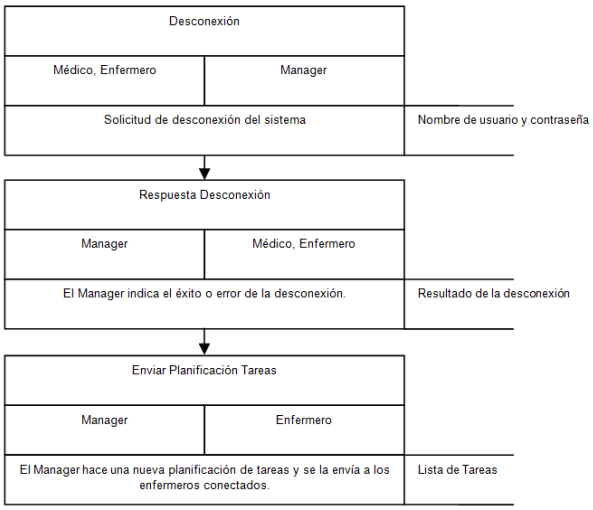

Figura D.7. Desconexión del Sistema de un Médico o Enfermero

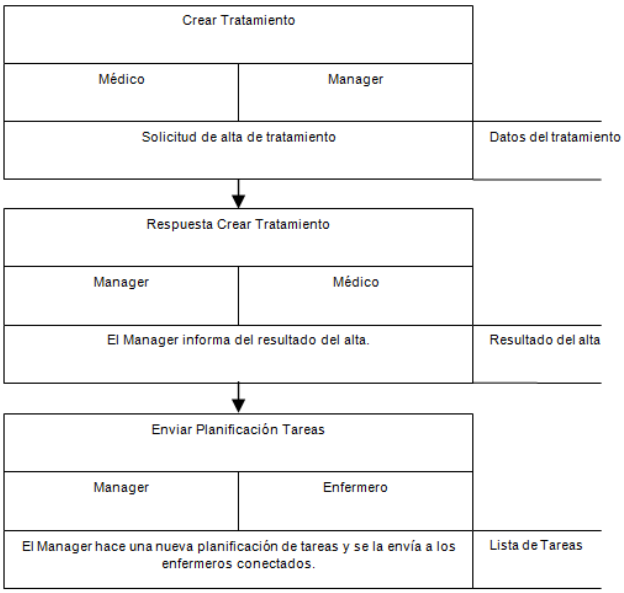

Figura D.8. Nuevo Tratamiento

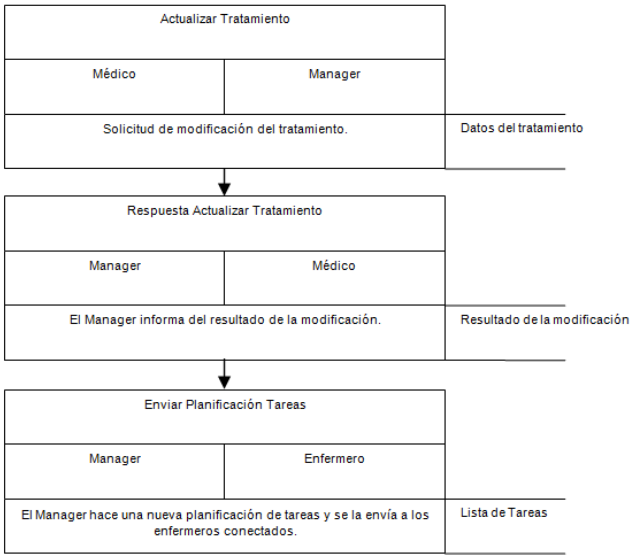

Figura D.9. Modificar Tratamiento

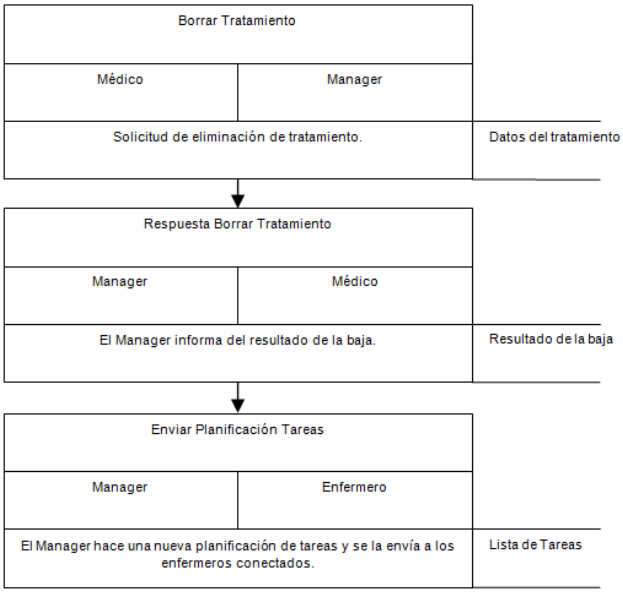

Figura D.10. Eliminar Tratamiento

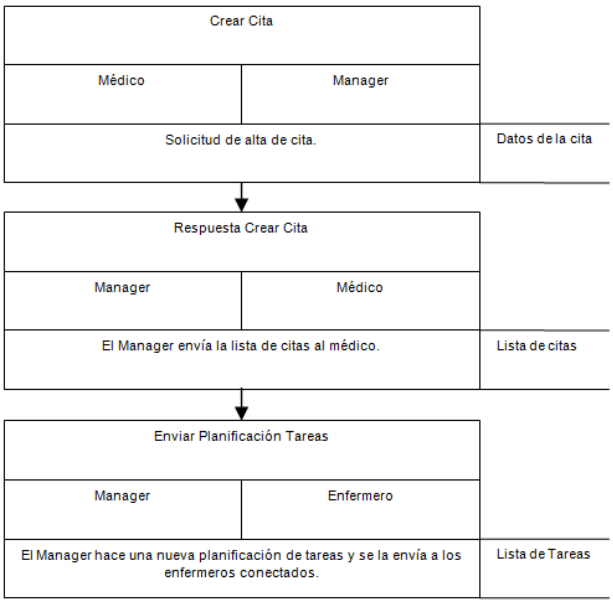

Figura D.11. Nueva Cita

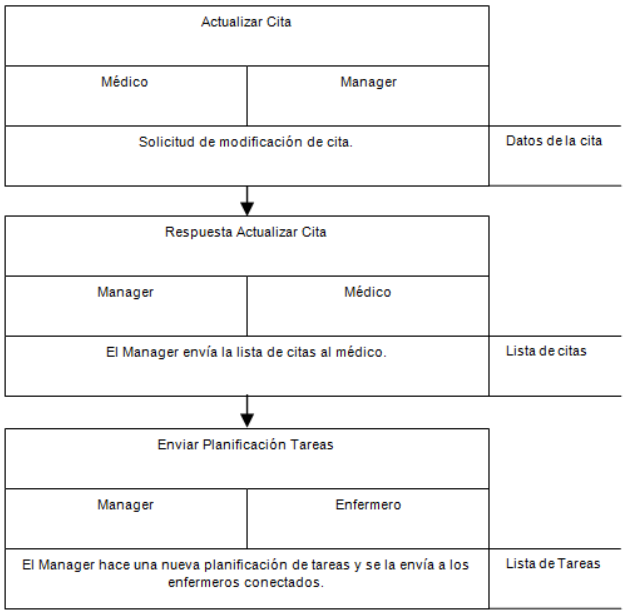

Figura D.12. Modificar Cita

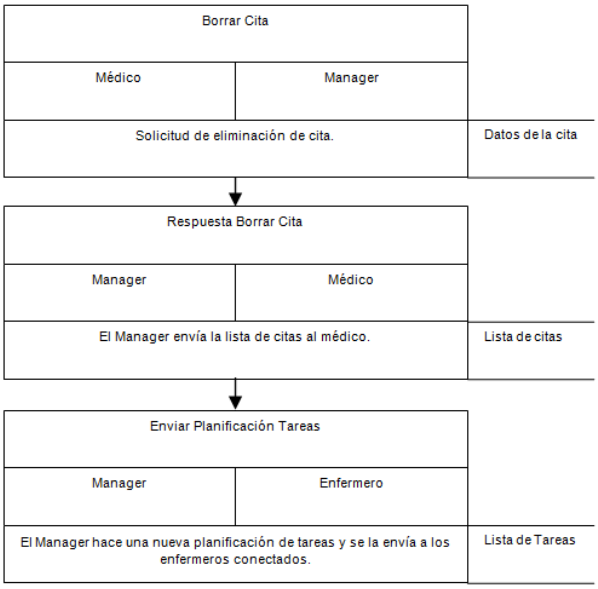

Figura D.13. Eliminar Cita

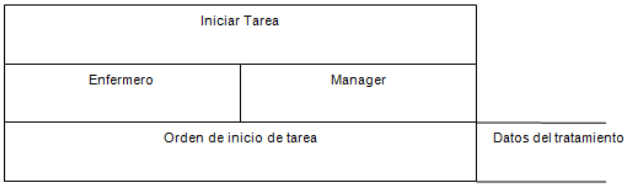

Figura D.14. Iniciar Tarea

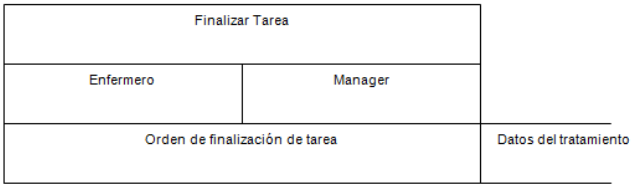

Figura D.15. Finalizar Tarea

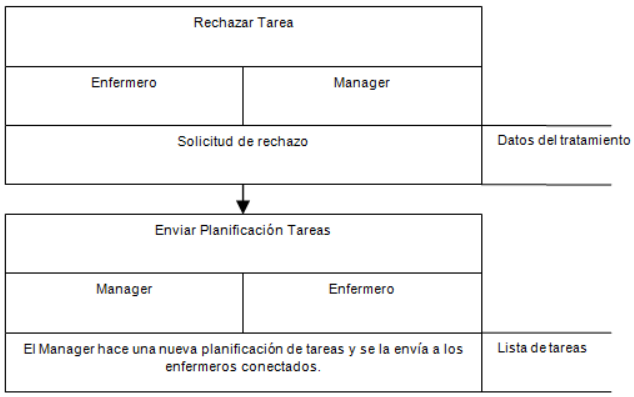

Figura D.16. Rechazar Tarea

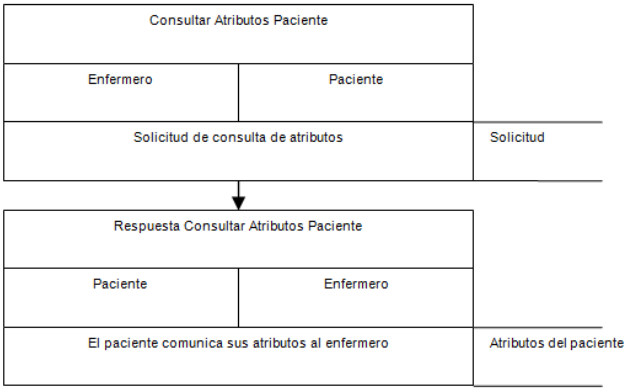

Figura D.17. Consultar Atributos de Paciente

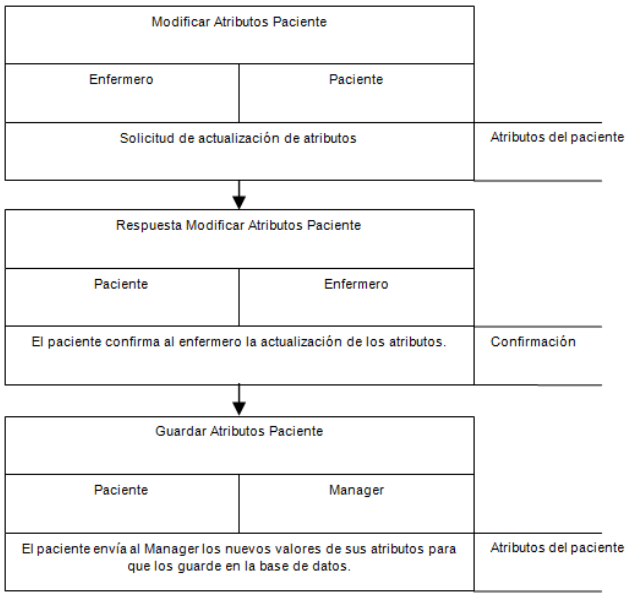

Figura D.18. Modificar Atributos de Paciente

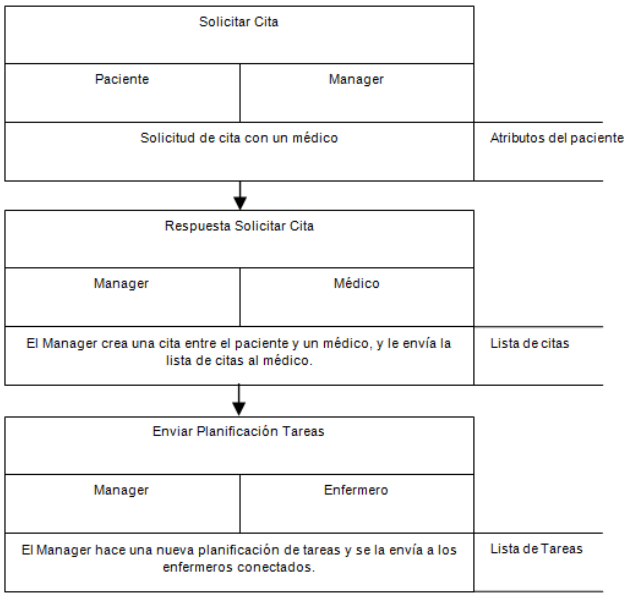

Figura D.19. Solicitar Cita

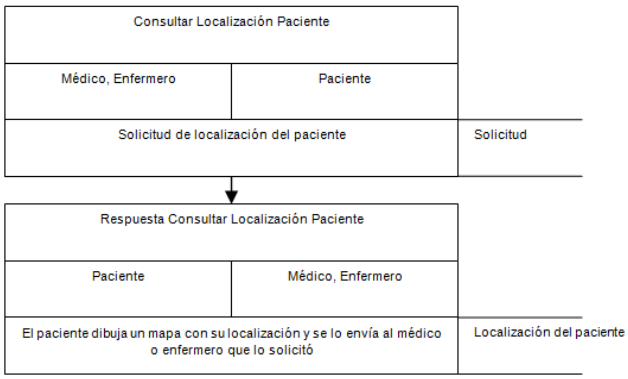

Figura D.20. Consultar Localización de Paciente

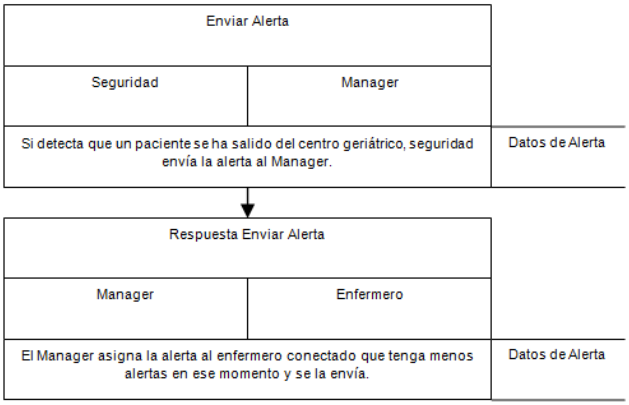

Figura D.21. Enviar Alertas

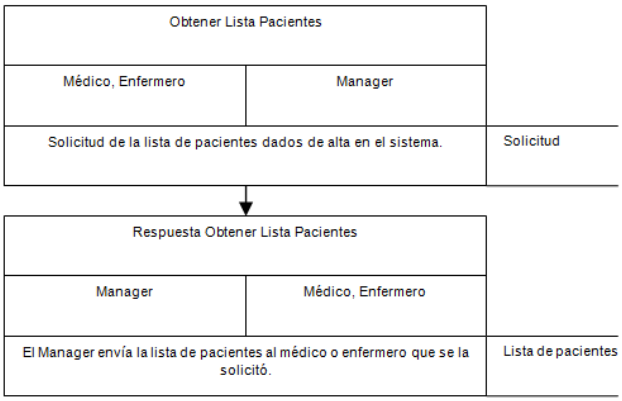

Figura D.22. Obtener Lista de Pacientes

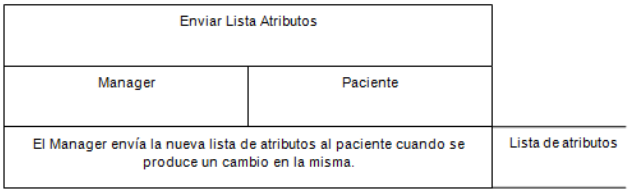

Figura D.23. Enviar Lista de Atributos de Paciente

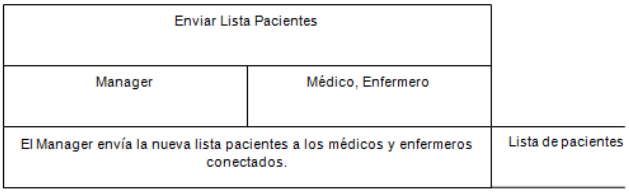

Figura D.24. Enviar Lista de Pacientes

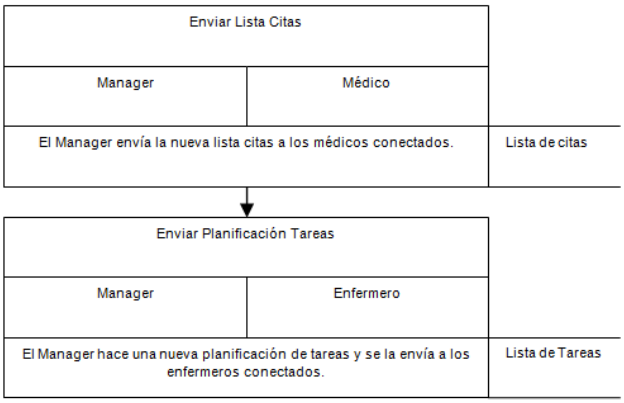

Figura D.25. Enviar Lista de Citas

Apéndice

E

Modelos del Diseño SysML

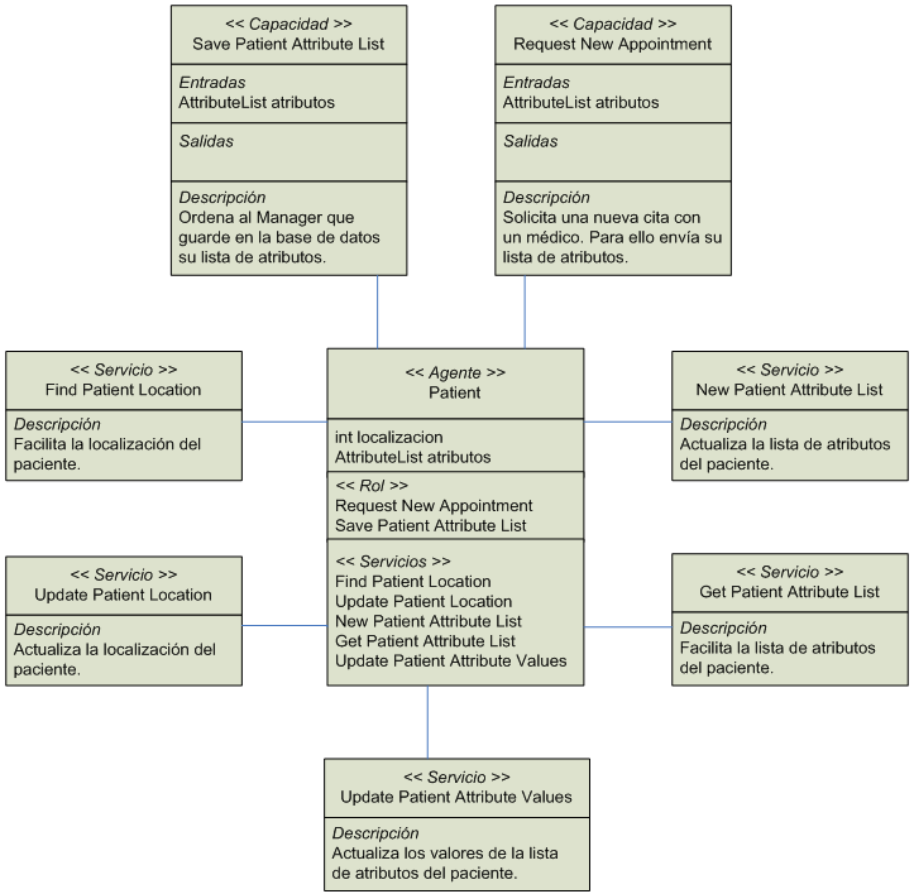

Figura E.1. Diagrama de Definición de Bloques: Patient Agent

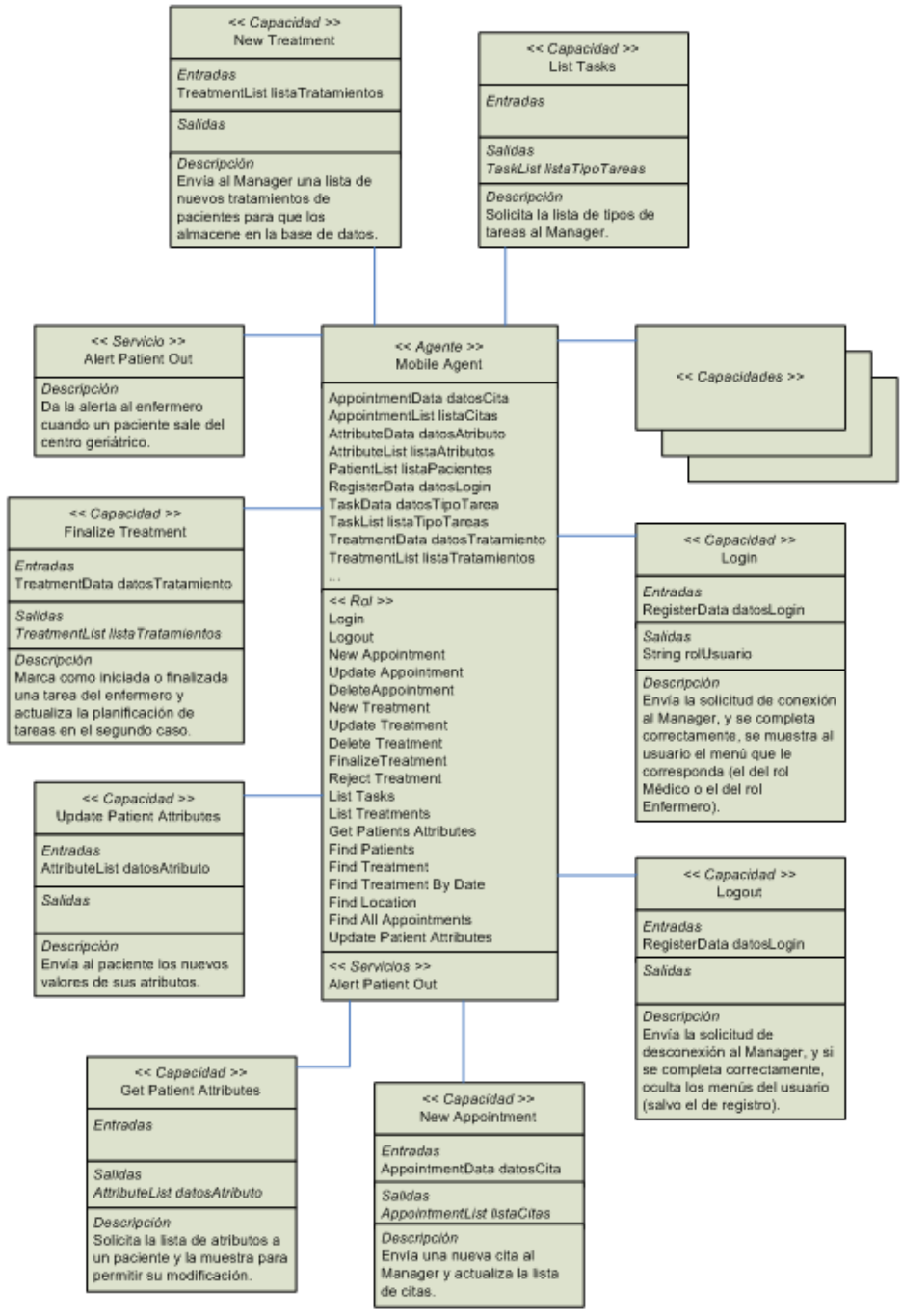

Figura E.2. Diagrama de Definición de Bloques: Mobile Agent

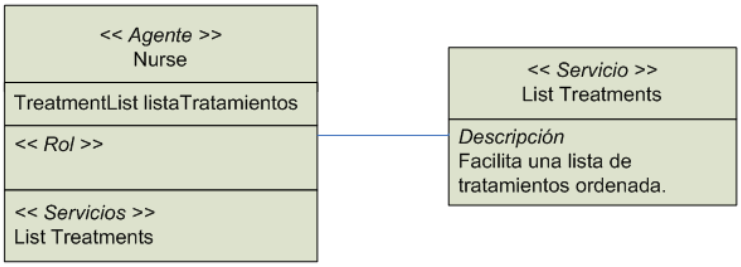

Figura E.3. Diagrama de Definición de Bloques: Nurse Agent

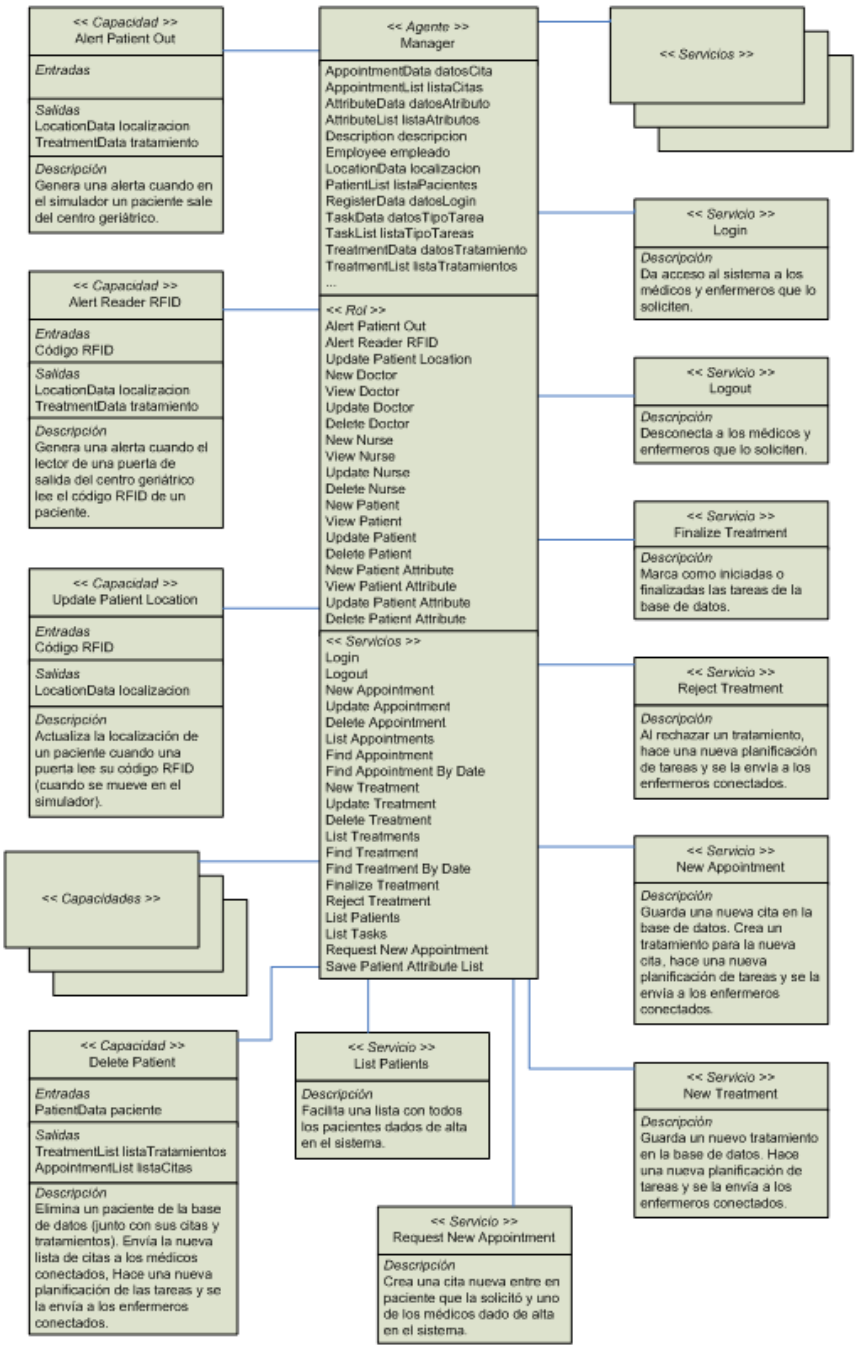

Figura E.4. Diagrama de Definición de Bloques: Manager Agent

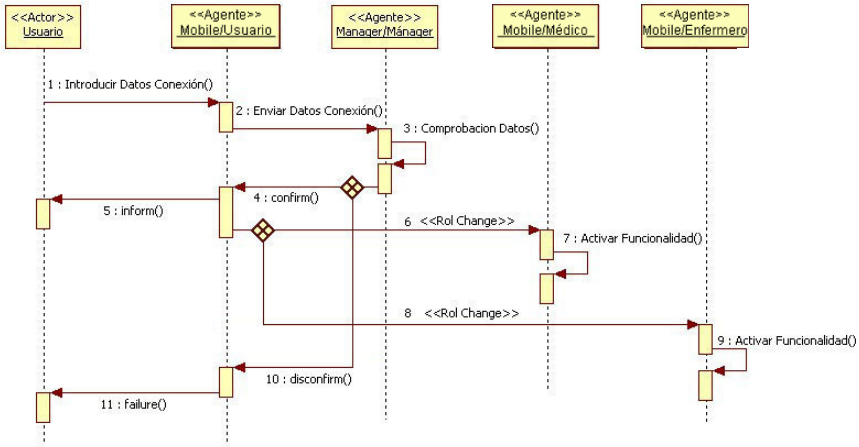

Figura E.5. Diagrama de Secuencia: Conexión

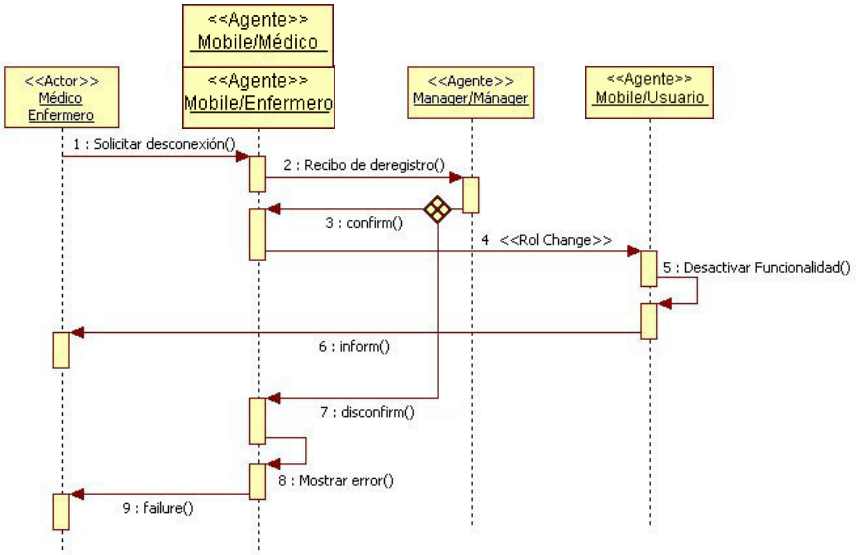

Figura E.6. Diagrama de Secuencia: Desconexión

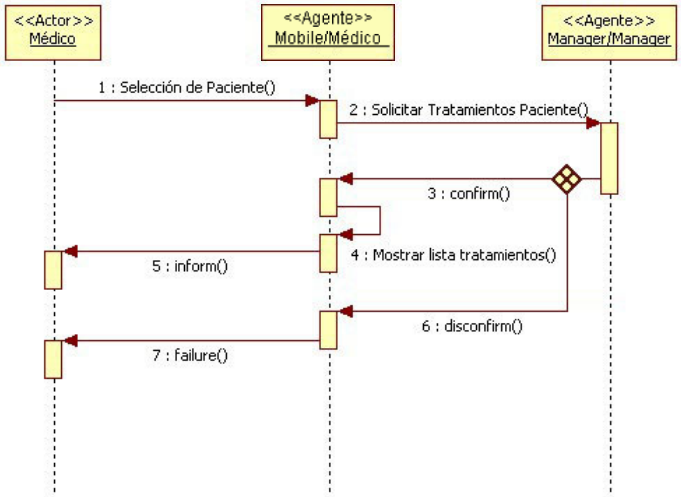

Figura E.7. Diagrama de Secuencia: Consulta de Tratamientos

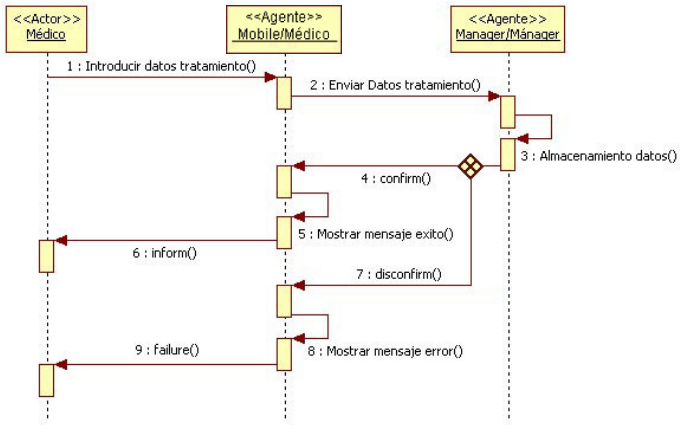

Figura E.8. Diagrama de Secuencia: Nuevo Tratamiento

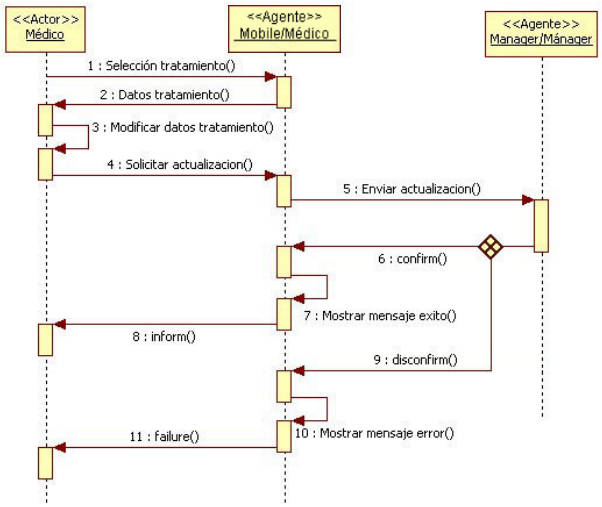

Figura E.9. Diagrama de Secuencia: Actualización de Tratamiento

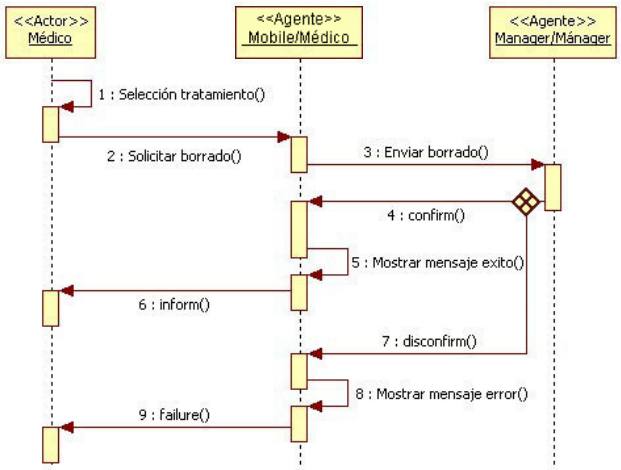

Figura E.10. Diagrama de Secuencia: Borrado de Tratamiento

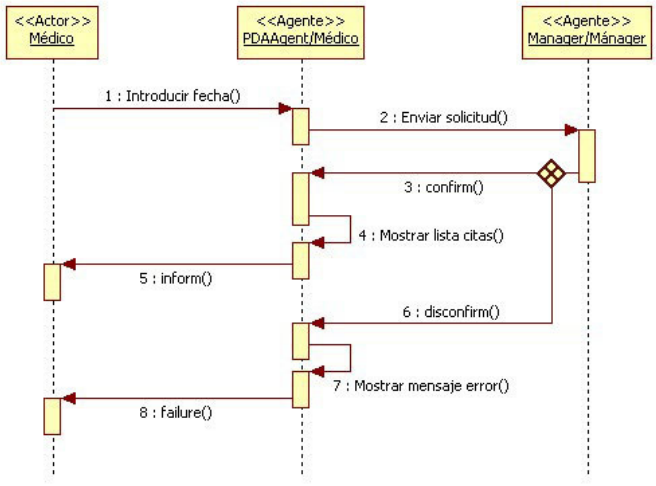

Figura E.11. Diagrama de Secuencia: Selección de Citas

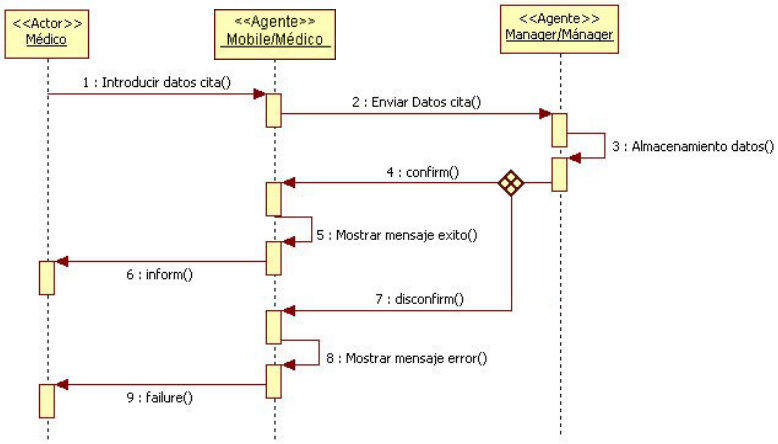

Figura E.12. Diagrama de Secuencia: Nueva Cita

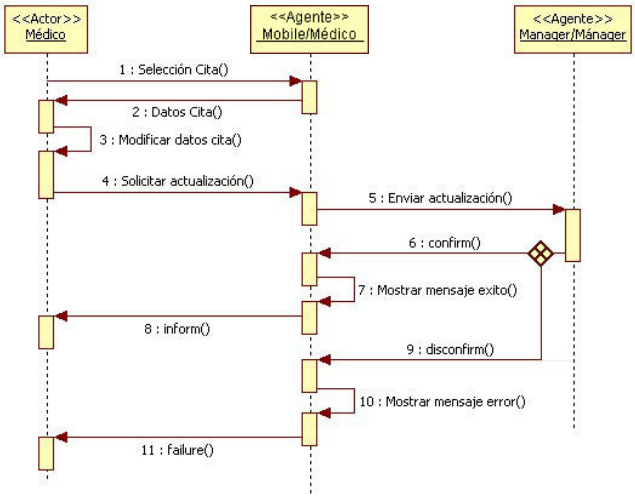

Figura E.13. Diagrama de Secuencia: Actualización de Cita

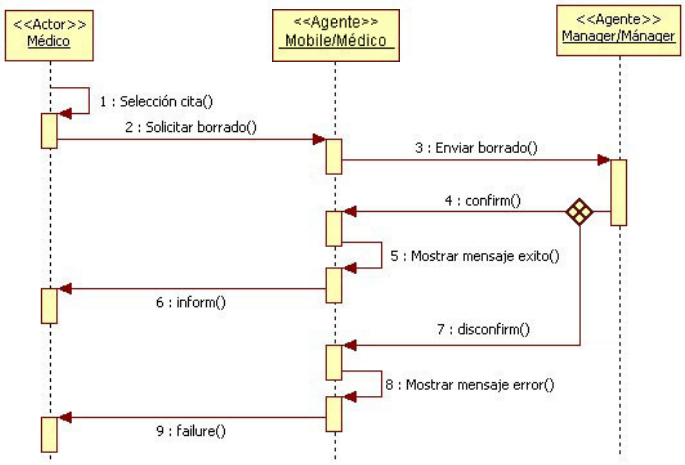

Figura E.14. Diagrama de Secuencia: Borrado de Cita

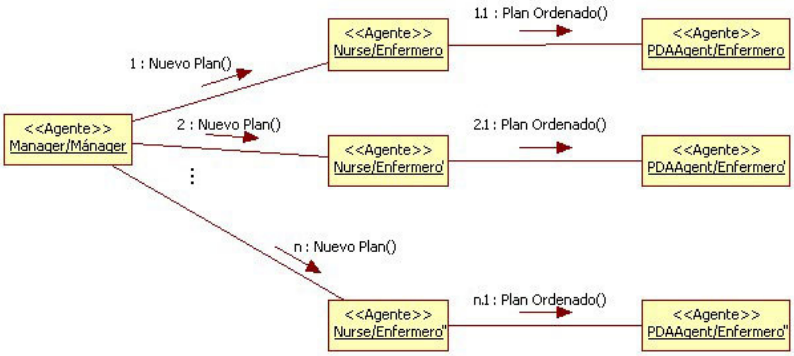

Figura E.15. Diagrama de Interacción: Planificación

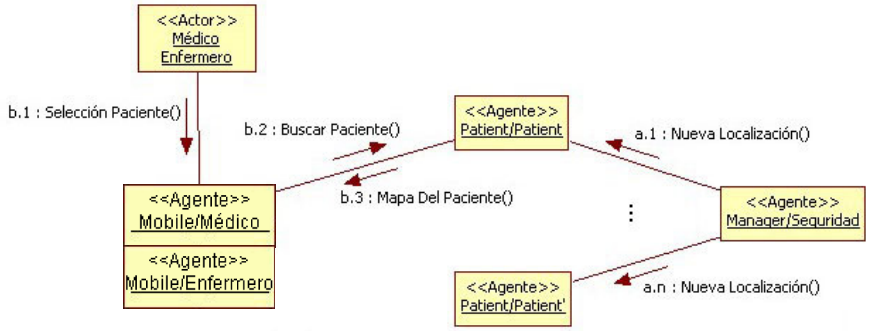

Figura E.16. Diagrama de Interacción: Localización de Usuario

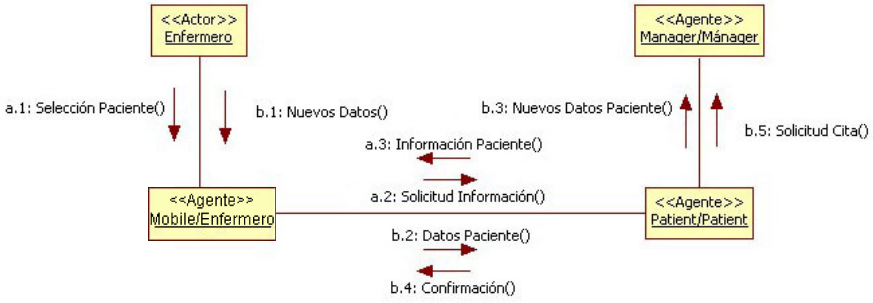

Figura E.17. Diagrama de Interacción: Información de Pacientes

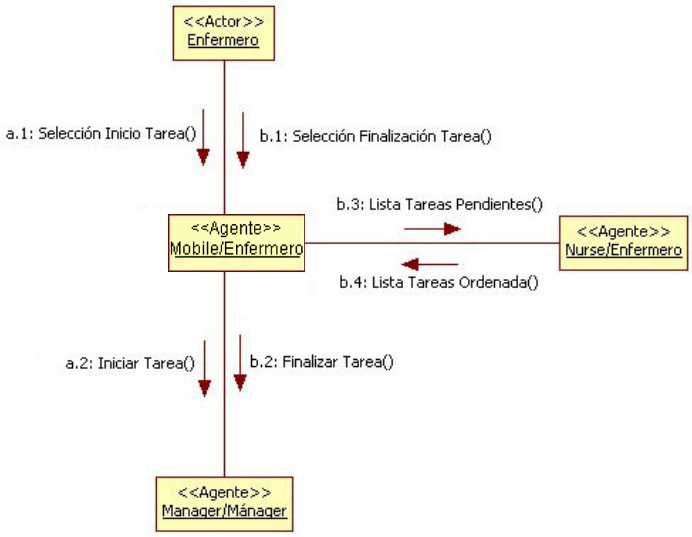

Figura E.18. Diagrama de Interacción: Inicio y Finalización de Tareas

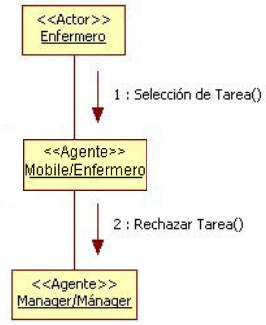

Figura E.19. Diagrama de Interacción: Rechazo de Tareas

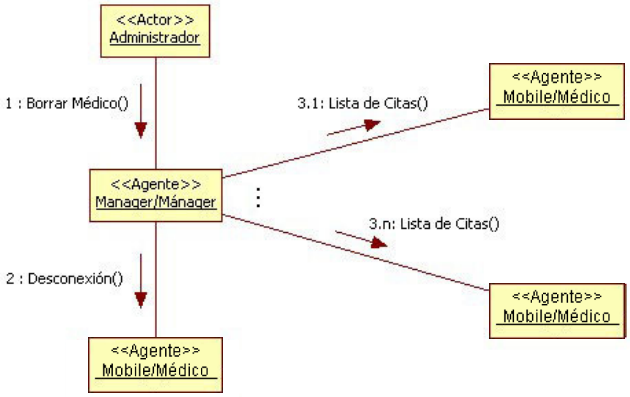

Figura E.20. Diagrama de Interacción: Borrado de Médico

Figura E.21. Diagrama de Interacción: Borrado de Enfermero

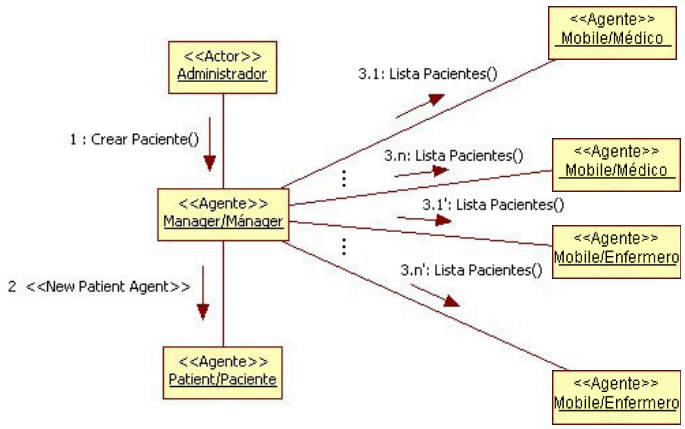

Figura E.22. Diagrama de Interacción: Alta de Paciente

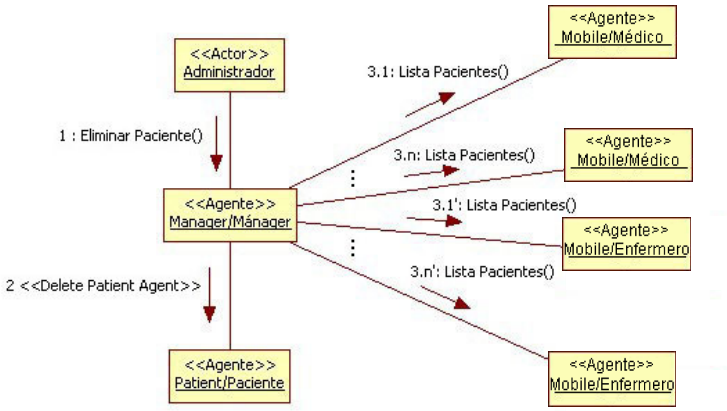

Figura E.23. Diagrama de Interacción: Borrado de Paciente

Figura E.24. Diagrama de Interacción: Alta de Atributo de Paciente

Figura E.25. Diagrama de Interacción: Actualización de Atributo de Paciente

Figura E.26. Diagrama de Interacción: Borrado de Atributo de Paciente

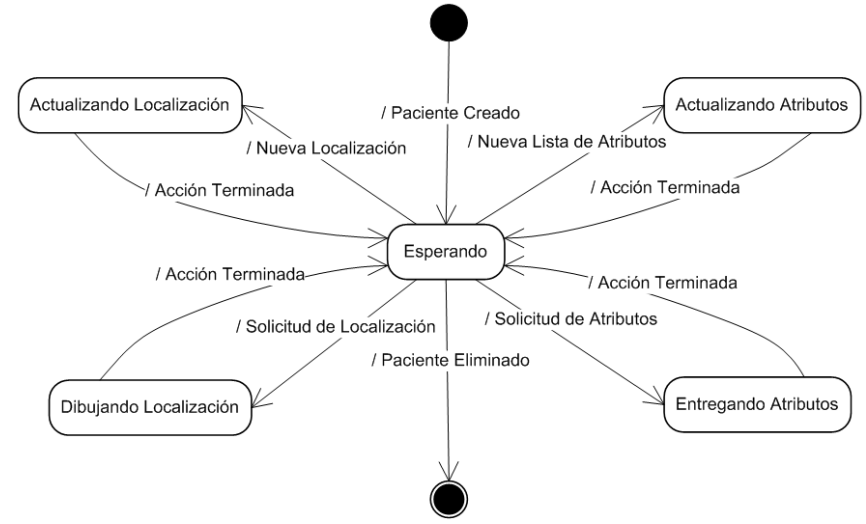

Figura E.27. Diagrama de Estados: Patient Agent

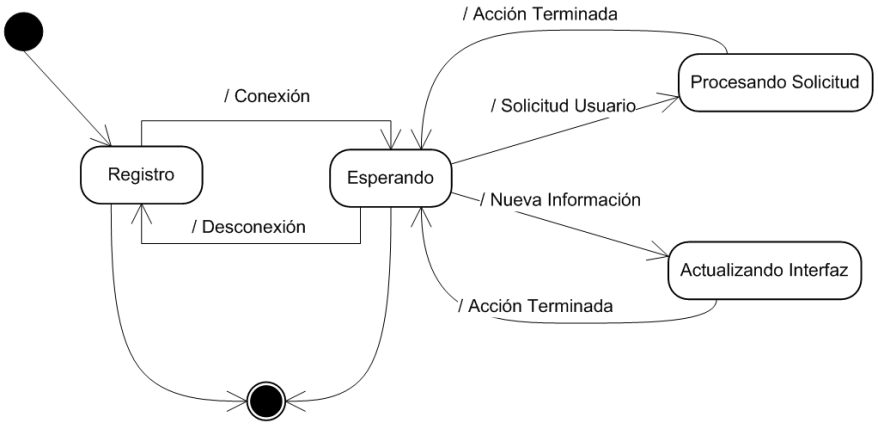

Figura E.28. Diagrama de Estados: Mobile Agent

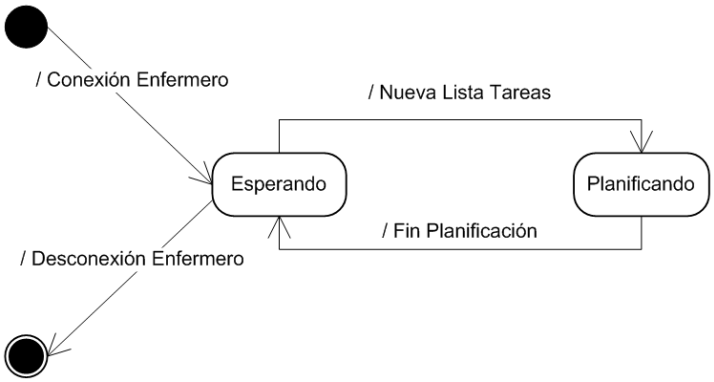

Figura E.29. Diagrama de Estados: Nurse Agent

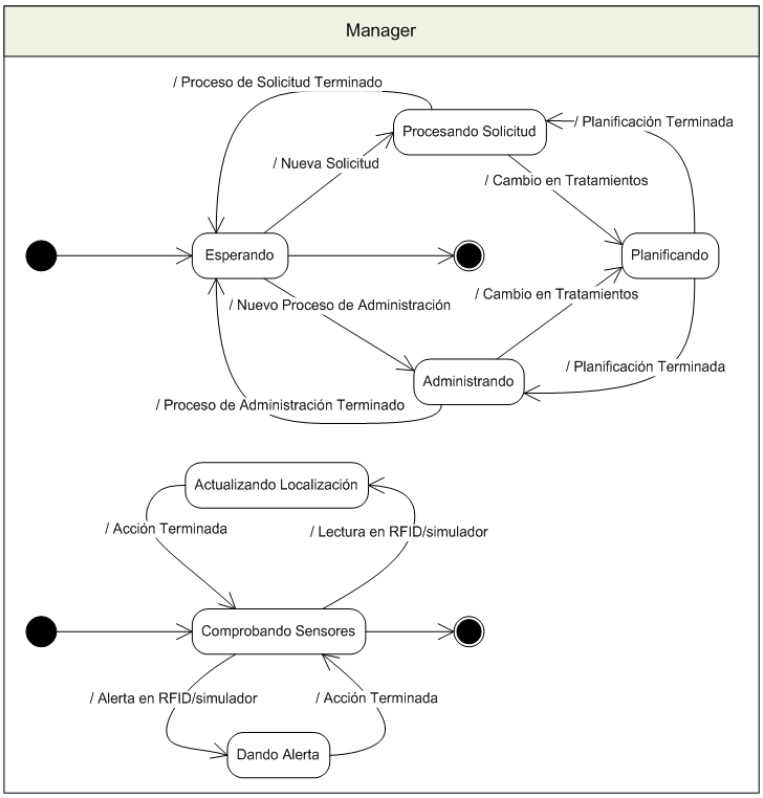

Figura E.30. Diagrama de Estados: Manager Agent