DEPARTAMENTO DE INFORMÁTICA Y AUTOMÁTICA

FACULTAD DE CIENCIAS

VNiVERSiDAD
D SALAMANCA

TESIS DOCTORAL

SISTEMA MULTIAGENTE SENSIBLE AL CONTEXTO

AUTOR

D. Juan Agustín Fraile Nieto

DIRECTORES

Dra. Dña. Belén Pérez Lancho

Dr. D. Javier Bajo Pérez

Septiembre de 2011

La memoria titulada “SISTEMA MULTIAGENTE SENSIBLE AL CONTEXTO” que presenta D. Juan Agustín Fraile Nieto para optar al Grado de Doctor por la Universidad de Salamanca ha sido realizada bajo la dirección de la profesora Dra. Dña. Belén Pérez Lancho, Profesora Titular del Departamento de Informática y Automática de la Universidad de Salamanca, y por el profesor Dr. D. Javier Bajo Pérez, Profesor Encargado de Cátedra de la Facultad de Informática de la Universidad Pontificia de Salamanca.

Salamanca, Septiembre de 2011.

Los Directores

El Graduado

Fdo.: Dra. Dña. Belén Pérez Lancho
Profesora Titular de Universidad
Informática y Automática
Universidad de Salamanca

Fdo.: D. Juan Agustín Fraile Nieto

Fdo.: Dr. D. Javier Bajo Pérez
Profesor Encargado de Cátedra
Facultad de Informática
Universidad Pontificia de Salamanca

Resumen

En una sociedad claramente influenciada por las nuevas tecnologías, los nuevos avances en computación sensible al contexto han supuesto un cambio importante en los hábitos de interacción de los usuarios con los sistemas informáticos y la información contextual. Es necesario tener en cuenta que cualquier contexto puede proporcionar un escenario para la realización de tareas automatizadas. Al mismo tiempo, los sistemas tienden a ofrecer un soporte colaborativo a los usuarios en su entorno. Tener en cuenta el contexto, introduce la necesidad de aportar soluciones tecnológicas que ayuden al usuario en función de su perfil y su situación, así como de los cambios que se producen en su entorno. Aunque las técnicas desarrolladas hasta ahora dentro de la computación sensible al contexto proporcionan una base sólida para modelar e interpretar el contexto, hay aspectos que no han sido cubiertos de manera plenamente satisfactoria en los sistemas actuales. Un sistema sensible al contexto utiliza su entorno para modificar su comportamiento y así satisfacer mejor las necesidades del usuario.

Durante los últimos años los sistemas multiagente han sido estudiados como una alternativa para modelar sistemas sensibles al contexto, aunque no siempre cubren todas las necesidades reales de los sistemas distribuidos. Los sistemas multiagente facilitan la reutilización de funcionalidades y optimización de la compatibilidad entre distintas plataformas. Este trabajo de investigación presenta la arquitectura *Home Care Context-Aware Computing* (HoCCAC). Se trata de una arquitectura multiagente especialmente desarrollada para satisfacer las necesidades que requieren los sistemas sensibles al contexto. HoCCAC define un conjunto de agentes que optimizan la gestión de información contextual y mejoran el desarrollo de sistemas distribuidos, adaptables, ubicuos, y con cierto grado de inteligencia. Esto lo logra HoCCAC con la utilización de agentes proveedores de información y la integración del mecanismo de

planificación *Critical Path Method* (CPM) y razonamiento *Case Based Reasoning* (CBR) en el principal agente de la arquitectura, el agente Intérprete.

Aunque los sistemas sensibles al contexto tienen numerosas aplicaciones para todo tipo de usuarios, son muy útiles especialmente en problemas asociados con las personas mayores o discapacitados, una realidad presente en los países más desarrollados del mundo. Mejorar las consecuencias sociales y económicas del envejecimiento poblacional es un reto para la concepción actual de los sistemas nacionales de salud y bienestar, los cuales han de garantizar la mejor asistencia a este colectivo de personas dependientes, de forma sostenible. Por este motivo para probar la utilidad de la arquitectura HoCCAC, se define un caso de estudio que consiste en el desarrollo de un prototipo de sistema multiagente orientado a mejorar los servicios de asistencia y cuidados médicos en el hogar a personas dependientes, especialmente con la Enfermedad Pulmonar Obstructiva Crónica (EPOC). Las conclusiones obtenidas han sido satisfactorias, demostrando que la arquitectura multiagente HoCCAC permite modelar e implementar sistemas basados en la sensibilidad contextual de forma eficiente.

Abstract

It is important to note that any context can set the stage for performing automated tasks. At the same time, systems tend to offer collaborative support for users in their context. Keeping the context in mind raises the issue of the need to provide technological solutions that help users with regards to their profile and location, as well as changes that can occur in their environment. Although the techniques developed to date for context-aware systems provide a solid base for modeling and interpreting the context, there are aspects that have been sufficiently well covered in current systems. A context-aware system uses the surrounding environment to modify its behavior and thus better satisfy user needs.

For the last several years, multiagent systems have been studied as an alternative for modeling context-aware systems, although they do not always cover all of the real needs for distributed systems. Multiagent systems facilitate the ability to reuse functionalities and to optimize the compatibility between different platforms. The present research study will introduce the *Home Care Context-Aware Computing* (HoCCAC) architecture, a multiagent architecture specially designed to meet the needs required by context-aware systems. HoCCAC defines a set of agents that optimizes the management of contextual information and improves the development of distributed, adaptable and ubiquitous systems with a certain degree of intelligence. HoCCAC achieves this by using information provider agents and incorporating the *Critical Path Method* (CPM) planning method and *Case Based Reasoning* (CBR) in the Interpreter Agent, the primary agent of the architecture.

Although context-aware systems have numerous applications for different kinds of users, they may be very useful in studying the problems associated with an aging population, a situation of particular concern in many of the most developed countries in the world. Improving the social and economic consequences associated with an aging population is one of the goals of current national health care systems which must provide the best possible assistance to this group of dependent individuals, while

ensuring its sustainability. Consequently, in order to test the HoCCAC architecture, a case study was defined, consisting of the development of a multiagent system prototype specially designed to improve services in the assistance and medical care at home for dependent persons, particularly those suffering from Chronic Obstructive Pulmonary Disease (COPD). The results obtained are quite satisfactory, demonstrating that the HoCCAC multiagent architecture can model and implement context aware systems very efficiently.

Agradecimientos

"Ama y haz lo que quieras. Si callas, callarás con amor; si gritas, gritarás con amor; si corriges, corregirás con amor, si perdonas, perdonarás con amor."
(San Agustín)

Este trabajo de investigación constituye un esfuerzo en el que de manera directa o indirecta han participado muchas personas. El grado de participación de cada uno de ellos y ellas es muy variado. Algunos han leído y han corregido, otros han colaborado y opinado, todos han tenido mucha paciencia y me han animado. Por este motivo todo el mérito de este trabajo no me corresponde a mí, y quiero expresar mi más sincero agradecimiento a compañeros, amigos y familia.

En especial dar las gracias a Juan Manuel Corchado y Belén Pérez por darme la oportunidad de entrar en este mundo, guiarme con sabios consejos y ofrecerme toda su ayuda. A Javier Bajo, al que admiro y aprecio por ser un trabajador incansable, mejor persona y por tener una paciencia infinita para resolver mis dudas. Este trabajo se ideó y materializó bajo la atenta mirada de los tres. Los tres son grandes personas y me han ofrecido la posibilidad de conocer unos modelos de carrera académica ejemplares.

Quiero dar las gracias a todos los miembros del grupo BISITE por ofrecerme siempre su apoyo incondicional. Con ellos he pasado muy buenos momentos y a todos les tengo que agradecer el haber llegado hasta aquí.

También doy las gracias a todos mis compañeros de trabajo en la Universidad Pontificia de Salamanca, especialmente a los profesores de la Facultad de Informática y a lo compañeros del área de desarrollo del CPD por su cariñosa acogida, su apoyo y por ese día a día tan entretenido que tenemos.

Gracias a mis padres Carmen y Ventura y a mi hermana Rogi, sin ellos habría sido imposible desarrollar este trabajo de investigación. Ellos han puesto su esfuerzo y empeño en poner todos los medios para que yo hoy sea lo soy *"seguid siempre así"*.

El agradecimiento más especial para mis abuelos, hace algunos años con su presencia, los últimos con su ausencia cercana y siempre con sus consejos, me ayudaron a llegar hasta aquí y estoy seguro que serían las personas más felices de poder leer las hipótesis, descripciones, conclusiones, opiniones, etc. que contine este trabajo. Abuelo Alfonso, *"este trabajo ha sido duro, pero hay cosas más duras en la vida…"*.

No me puedo olvidar de la familia y los abuelos de Sandra. Leónides y Bienvenida han participado activamente en el caso de estudio descrito y en ellos se basa parte de este trabajo. Gracias por vuestro cariño y paciencia.

Por último y siempre al principio, este trabajo se lo dedico a Sandra por estar a mi lado, tener paciencia, animarme mucho y en definitiva ser como es, *"no cambies"*. Y a Javier, que en unos meses verá la luz y espero que algún día pueda realizar un trabajo de este tipo, si así él lo desea.

Gracias a todos.

Índice

Índice de Figuras

Índice de Tablas

1

Introducción

La tecnología de la información está avanzando a un ritmo muy alto acercándose a nuevos segmentos de población y facilitando mayor número de tareas en un amplio número de dominios de aplicación. Con la penetración de Internet y las nuevas formas de comunicación, la información y los servicios dejan de estar accesibles a segmentos de población concretos y comienzan a estar al alcance de una gran mayoría de usuarios. La nueva era tecnológica está produciendo cambios profundos en el modo de proceder de nuestras actividades diarias. De esta manera aparece la necesidad de crear nuevos sistemas que se adapten cada vez mejor a las demandas de los usuarios y su entorno, conduciendo esto, durante los últimos años, hacia los sistemas sensibles al contexto. Estos sistemas almacenan y analizan toda la información relevante que los rodea y forma parte del entorno de los usuarios. La ubicación de un usuario, las actividades que realiza, la temperatura y las condiciones de iluminación del entorno donde vive, son parte de la información de contexto.

La computación sensible al contexto es un área emergente desarrollada a partir de la computación ubicua [Costa de, *et al.,* 2008] [Henricksen, *et al.,* 2006] y que influye

en el diseño de protocolos, comunicaciones e integración de sistemas y dispositivos. Los sistemas sensibles al contexto proponen nuevas formas de interacción entre las personas y la tecnología, haciendo que la tecnología se adapte a las necesidades de los individuos y al entorno que los rodea. Los sistemas sensibles al contexto proporcionan mecanismos para el desarrollo de aplicaciones que conocen sus contextos y son capaces de adaptarse a posibles cambios [Abowd, *et al*., 2007]. En esta memoria se desarrollará una arquitectura multiagente orientada a trabajar con información de contexto y que simplificará el desarrollo de sistemas sensibles al contexto. La tendencia actual para mostrar información a los usuarios de los sistemas informáticos, es la distribución a través de sistemas heterogéneos y redes de información variadas que tienen en cuenta los dispositivos pequeños y portátiles existentes en la actualidad. Se trata de sistemas que se insertan en la vida cotidiana del usuario de forma no intrusiva y facilitan el que los usuarios se centren en las tareas que deben hacer, y no se preocupen por las herramientas que tienen que utilizar. La arquitectura definida en este trabajo de investigación presentará un método innovador basado en la cooperación y comunicación entre agentes, que facilitará la incorporación de nuevos dispositivos que proporcionan información de contexto. Estos dispositivos se podrán acoplar al sistema de forma automatizada, sin que el usuario perciba ninguna variación a la hora de realizar sus tareas diarias. El rápido avance de las tecnologías para el control y la automatización hará posible incorporar dispositivos contextuales que proporcionarán a los agentes la capacidad para percibir el estado del entorno y reaccionar físicamente sobre éste de acuerdo a las necesidades de los usuarios.

Por otro lado los escenarios de ayuda a personas dependientes son un claro ejemplo de entornos contextuales donde se puede mejorar en gran medida la calidad de vida de una persona. El alto crecimiento del número de personas dependientes y el avanzado estado de desarrollo de las tecnologías hacen plantear la necesidad de generar nuevas soluciones para entornos de cuidados en el hogar (*Home Care*) [Bajo *et al.,* 2007] [Corchado *et al.,* 2008a]. En *Home Care* se requiere la utilización de sensores, actuadores, dispositivos y sistemas inteligentes para construir un entorno distribuido en el que las funciones del hogar estén automatizadas. En este sentido los sistemas multiagente facilitan el desarrollo de entornos *Home Care*, ya que pueden proporcionar un soporte continuo en la planificación de actividades de personas

dependientes, la predicción de situaciones potencialmente peligrosas y el manejo de soportes físicos y cognitivos de la persona asistida. La arquitectura multiagente desarrollada en este trabajo de investigación incorporará en el principal agente de dicha arquitectura un elemento novedoso para simplificar y optimizar la planificación de actividades y poder ofrecer soluciones anticipadas a situaciones de riesgo. Este componente novedoso combina el método de planificación *Critical Path Method* (CPM) con un sistema de *Case-Based Reasoning* (CBR) incorporado a un agente inteligente. La combinación del método CPM y el sistema de razonamiento CBR permite desarrollar y optimizar la realización de planes de tareas cotidianas asociadas a un usuario. El razonamiento CBR no es una forma de razonamiento en la que se utilicen reglas generalizadas, sino que se razona a partir del conocimiento almacenado en una memoria de casos que contiene experiencias previas. De esta forma, cuando el agente de la arquitectura se encuentra con un nuevo problema, utiliza los casos almacenados en la memoria, más similares al caso del problema, teniendo en cuenta tanto las condiciones bajo las que se solucionaron estos casos como las condiciones actuales. En los últimos años los sistemas CBR se han aplicado a una gran variedad de entornos de diferentes características [Bajo *et al.,* 2006] [Ma *et al.,* 2005] [Bai *et al.,* 2008], pero nunca en combinación con el método de planificación CPM, como se propone en la arquitectura que se describe en el capítulo 4. El método CPM identifica las actividades que limitan la duración de un plan de tareas y así logra que este se realice lo más pronto posible. La combinación del método de planificación CPM y el sistema de razonamiento CBR en un agente inteligente permite dotar a dicho agente de una mejor capacidad de adaptación a entornos dinámicos y cambiantes, además de mejorar el desarrollo de planes de tareas óptimos.

La base sobre la que se cimienta este trabajo de investigación son los agentes y los sistemas multiagente, que han sido aplicados con éxito en diversos desarrollos relacionados con entornos dependientes en el grupo de investigación de Biomedicina, Sistemas Inteligentes y Tecnología Educativa BISITE [Tapia *et al.,* 2009] [Fraile *et al.,* 2009a]. El grupo de investigación BISITE 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, este trabajo de investigación sigue una línea continuista en la investigación y aplicación de estos sistemas, extendiendo sus

posibilidades hacia otras áreas, en especial la computación sensible al contexto y el valor añadido que ofrecen los agentes y sistemas multiagente a esta área.

Un agente puede describirse como un sistema computacional que se sitúa en el entorno y es capaz de actuar de forma autónoma en dicho entorno para alcanzar sus objetivos [Wooldridge, 1995]. Ampliando un poco más esta definición, tenemos que, un agente es cualquier programa 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]. Los agentes poseen características tales como autonomía, razonamiento [Wooldridge, 1995] [Russell *et al*., 1995] [Georgeff *et al.,* 1998], reactividad, habilidades sociales, pro-actividad y movilidad, entre otras. Estas características, sumadas a todos los elementos diferenciadores, enumerados en esta introducción y que se presentarán en este trabajo de investigación, hacen que la tecnología de agentes resulte adecuada para ser utilizada en la construcción de sistemas sensibles al contexto.

Una vez identificados los retos de este trabajo de investigación se formula la hipótesis de partida, se enumeran los objetivos y se describe la motivación que conduce al desarrollo de esta memoria. Finalmente, en este capítulo se describe la metodología seguida y la estructura de la memoria.

1.1 Hipótesis y objetivos

El objetivo final de este trabajo es investigar en la obtención de una arquitectura multiagente, que incorpora elementos con capacidades de razonamiento, planificación y aplicable al desarrollo de entornos sensibles al contexto. La arquitectura debe proponer funcionalidades con capacidades de razonamiento y planificación de tareas capaces de complementarse entre sí. Estas funcionalidades deben poder ser ejecutadas en entornos inteligentes, distribuidos y dinámicos, y tienen que facilitar la interacción con el usuario. Además la arquitectura debe ser capaz de soportar agentes con mecanismos que permitan capturar información contextual diversa. Para ello la arquitectura podrá añadir dinámicamente nuevos agentes Proveedores de información, cuando detecte que el contexto proporciona otro tipo de información valiosa para el

sistema. De este modo surge la arquitectura multiagente *Home Care Context Aware Computing* (HoCCAC). HoCCAC, también, proporciona mecanismos que permiten modelar problemas relacionados con la computación sensible al contexto en términos de agentes y sistemas multiagente. La Teoría de Agentes [Wooldridge y Jennings, 1995] considera a los agentes y sistemas multiagente fundamentales en el desarrollo de contextos inteligentes.

La hipótesis defendida en esta memoria es que *la arquitectura HoCCAC es un modelo adecuado y adaptado al diseño y desarrollo de sistemas sensibles al contexto.* Esto se debe a que la arquitectura propuesta incluye agentes inteligentes con capacidades de razonamiento y planificación. La combinación de estas capacidades permite a los agentes inteligentes desarrollar planes óptimos de tareas, de duración mínima, basados en conocimiento almacenado a partir de experiencias pasadas. Esta combinación de capacidades proporciona a la arquitectura una mayor flexibilidad para adaptar su comportamiento en tiempo de ejecución. También en tiempo de ejecución, los agentes de la arquitectura facilitan la asociación de dispositivos que proporcionen nueva información contextual y así el sistema la puede gestionar de forma eficiente. En definitiva, con este trabajo de investigación se trata de demostrar que si se utiliza la arquitectura HoCCAC en el desarrollo de sistemas sensibles al contexto, se simplifica el desarrollo de este tipo de sistemas, se optimiza el rendimiento en la generación de planes de tareas dinámicos para resolver tareas cotidianas y se facilita la inclusión de nuevas fuentes de información contextuales.

Para poder alcanzar el objetivo final es necesario estudiar el estado del arte de los sistemas sensibles al contexto y los sistemas multiagente, así como de los distintos enfoques de sistemas multiagente existentes que están aplicados sobre entornos dependientes y entornos sensibles al contexto. En el marco de esta tesis se ha trabajado en:

- Revisar el estado del arte de conceptos y sistemas relacionados con la sensibilidad al contexto, entornos dependientes y de cuidados en el hogar, todas ellas áreas relacionadas con la investigación.
    - Examinar los problemas asociados con las áreas relacionadas con la investigación.

  - Analizar los beneficios proporcionados por los sistemas sensibles al contexto desarrollados en diferentes ámbitos de dominio.
  - Revisar y analizar otras áreas relacionadas con la investigación como son la computación ubicua, la computación pervasiva, la Inteligencia Ambiental, la domótica y los dispositivos inalámbricos.

- Proponer una arquitectura multiagente eficiente que facilite la captura de información contextual a partir de sensores distribuidos en el entorno. Los sensores pueden recoger distintos tipos de información como por ejemplo luminosidad, temperatura, humedad, etc. y la arquitectura propuesta debe poder permitir agregar cualquier nuevo sensor en tiempo de ejecución.
- Buscar mecanismos formales útiles para resolver problemas de planificación de tareas cotidianas de una persona. La arquitectura propuesta debe incorporar estos mecanismos de planificación y ser capaz de optimizar el desarrollo de las tareas en entornos dinámicos. La arquitectura propuesta necesita mecanismos de planificación sencillos y no mecanismos de planificación complejos cuyo tiempo de respuesta sea elevado.
- Crear un modelo de sistema multiagente, cercano a la implementación. Para ello se utilizará la metodología *Systems Modeling Language* (SysML) [SysML.org, 2010], descrita en detalle en el Apéndice B de este trabajo. SysML es un subconjunto ampliado de UML 2.0. La principal ventaja de utilizar la metodología SysML es que permite especificar sistemas que combinan elementos *hardware* como sensores con aplicaciones o componentes *software*.
- Aplicar el modelo obtenido a un caso de estudio concreto en un entorno dependiente. Los resultados obtenidos en el prototipo desarrollado se analizarán y evaluarán. Así se conseguirá información útil para poder llevar a cabo futuras mejoras sobre el modelo inicial.
- Incorporar al caso de estudio nuevas formas de interactuar con el entorno a través de la combinación entre agentes y nuevas tecnologías móviles como *Near Field Communication* (NFC).

Es importante destacar que los objetivos planteados en este trabajo de investigación no están enfocados a crear únicamente servicios para contextos tecnológicamente muy

avanzados, sino que aspiran a lograr espacios inteligentes ubicuos, capaces de cubrir gran parte de las necesidades que plantean las acciones cotidianas de los usuarios.

1.2 Motivación

Los sistemas sensibles al contexto ofrecen un gran potencial para mejorar la calidad de vida de las personas y simplificar el uso de la tecnológica a través de servicios personalizados. Asimismo, la tecnología proporciona a las personas formas nuevas de comunicación e interacción con su entorno. El desarrollo de software basado en la computación sensible al contexto 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. Estos modelos de arquitectura proporcionan el soporte elemental para la implementación de sistemas sensibles al contexto que manejan información del contexto relacionada con: la localización del usuario, las tareas que este realiza o las condiciones atmosféricas del entorno entre otras. Esto implica que el sistema desarrollado tendrá un mayor grado de conocimiento del contexto de usuario para ofrecer soluciones proactivas y eficientes. La finalidad de estos sistemas es asistir a los usuarios en sus tareas cotidianas, proporcionando la información apropiada, a la persona adecuada y en el momento justo.

Actualmente, hay pocos sistemas que utilizan información de diferentes atributos contextuales y relacionan esta información para interactuar sobre usuarios. Son varios los sistemas sensibles al contexto centrados en resolver un problema determinado, por ejemplo ofrecer servicios a partir de un atributo contextual como es la localización [Griswold *et al.,* 2004], incluir parámetros ambientales en aplicaciones contextuales [Ermi y Mäyrä, 2005] [Koskinen y Suomela, 2006], utilizar las preferencias de usuario para ofrecer servicios **[**Kurkovsky *et al.,* 2005], incorporar redes inalámbricas y tecnología sin contacto en sistemas médicos de apoyo en desastres [Brown *et al.,* 2006], detectar eventos críticos en el hogar [Bricon-Souf y Newman, 2007] o monitorizar la actividad física en espacios abiertos exteriores [Buttussi y Chittaro, 2008]. Por este motivo es importante el desarrollo de nuevas arquitecturas, como la que se presentará en esta investigación, que faciliten el diseño y construcción de

sistemas sensibles al contexto de manera estandarizada. Por otro lado son pocos los sistemas sensibles al contexto que incluyen el concepto de razonamiento sobre la información contextual. Este concepto implica transformación del contexto mediante utilización del conocimiento adquirido. Estos procesos normalmente requieren de motores de razonamiento que utilizan bases de conocimiento para realizar el trabajo. En todos los sistemas de este tipo una vez obtenido y almacenado el contexto, ha de enviarse la información o datos procesados a las aplicaciones y componentes que lo soliciten. La arquitectura que se presentará a lo largo de este trabajo de investigación, está basada en el estudio y propuesta de integración, de nuevos mecanismos de razonamiento y planificación, sobre agentes inteligentes en entornos sensibles al contexto. La arquitectura que se desarrollará combina como novedad el método de razonamiento CBR [Bou *et al.,* 2008] y el método de planificación CPM [Castro-Lacouture *et al.,* 2009] y los incorpora a un agente inteligente para generar y optimizar planes de tareas sencillos relacionados con tareas cotidianas. Estos nuevos mecanismos permitirán un funcionamiento proactivo y una re-planificación dinámica y autónoma del propio sistema para ayudar al usuario en sus tareas diarias. El propio sistema se adaptará a situaciones novedosas para ir mejorando sus propios resultados y evolucionará con el paso del tiempo, para combinar la información disponible y poder resolver satisfactoriamente un mayor número de casos. Estas situaciones son muy habituales en entornos sensibles al contexto donde se recoge gran variedad de información del entorno. Debido a la gran variedad de información que puede llegar a proporcionar un contexto es necesario que la arquitectura sea capaz de incorporar dinámicamente nuevos agentes proveedores de información. La cooperación y comunicación entre agentes permite simplificar las labores de gestión de las fuentes de información contextual. Por este motivo, en este trabajo de investigación, también como novedad, se diseñará y construirá una plataforma multiagente que es capaz de procesar información contextual variada y generar planes de tareas cotidianas para un usuario en tiempo de ejecución. Estos planes de tareas desarrollados por el sistema multiagente deben mejorar el rendimiento del usuario al que se dirigen.

El problema de la integración de fuentes de información ha dado lugar a la aparición de nuevos modelos de representación y arquitecturas distribuidas de adquisición de información y tratamiento de dicha información, dado que las técnicas

clásicas no facilitan esta tarea. El paradigma multiagente ha sido utilizado para implementar este tipo de sistemas distribuidos satisfactoriamente [Corchado *et al.,* 2006] [Bajo *et al.,* 2007]. La arquitectura multiagente HoCCAC que se presentará en este trabajo de investigación se basa en un modelo distribuido de agentes, que en este caso, trata de resolver de forma novedosa la integración de fuentes de información variadas como se ha explicado en el párrafo anterior. En dicha arquitectura, el agente intérprete, principal agente de la arquitectura, interactúa con los sensores contextuales a través de los agentes Proveedores y el agente Interface, tanto para mantener actualizada la información contextual como para poder interactuar con el entorno. De esta manera, el agente Intérprete determina un curso de acciones óptimas o planes para que el usuario alcance un objetivo en el periodo de tiempo más corto posible. Además el agente Intérprete evalúa el cumplimiento del plan con la información que recibe del resto de agentes de la plataforma.

Como ya se ha comentado anteriormente, son varias las fuentes de información que puede proporcionar un contexto, entonces, estas fuentes de información proporcionan una estimación sobre la situación del contexto de calidad y fiabilidad determinadas. Una estimación más precisa se podrá calcular, en ocasiones, mediante la combinación de información procedente de diferentes fuentes. Las estrategias cooperativas de adquisición de información contextual podrán adaptarse a un mayor número de casos aumentando la disponibilidad de la estimación. La arquitectura HoCCAC facilita la integración y comunicación entre agentes y dispositivos siendo un único agente Proveedor el encargado de supervisar y controlar el funcionamiento de un dispositivo. Todos estos agentes Proveedores se comunican con el agente Intérprete que es el que trata y combina la información recibida para generar soluciones eficientes. Para generar estas soluciones eficientes el agente intérprete integra el método de planificación CPM y el sistema de razonamiento CBR.

Con el fin de evaluar la adaptación de la plataforma a un entorno real, dicha plataforma se validará sobre un caso de estudio orientado a mejorar la calidad de vida de un paciente con Enfermedad Pulmonar Obstructiva Crónica (EPOC) en su hogar. Los pacientes EPOC son un colectivo que necesita de soluciones tecnológicamente avanzadas que traten de mejorar su calidad de vida. Aunque los pacientes con EPOC son enfermos crónicos, sí existen una serie de acciones cotidianas muy concretas, que

pueden ayudar a demorar el avance de la enfermedad. Este conjunto de acciones se pueden agrupar en planes de tareas, cuyo desarrollo puede mejorar con el paso del tiempo y su cumplimiento es fácilmente evaluable. En este escenario es esencial la evaluación del comportamiento de los agentes inteligentes que se describirán en la arquitectura HoCCAC, las capacidades de razonamiento utilizadas por los agentes y los métodos de planificación integrados en los agentes.

1.3 Metodología de la investigación

El método de trabajo seguido en el desarrollo de este trabajo de tesis se estructura en una serie de pasos que se describen a continuación. Todos estos pasos se enlazan y complementan unos con otros hasta conseguir lograr los objetivos planteados anteriormente.

Primero se analiza la problemática y se definen sus características. Una vez analizado el problema se propone una hipótesis para solucionarlo y se plantean los objetivos a cumplir para lograrlo. Además se describe la motivación que lleva a plantear los objetivos planteados y la metodología que se sigue. En paralelo, se hace un estudio del estado del arte. Para ello se realiza una revisión y descripción del estado actual de todos los conceptos relacionados con la computación sensible al contexto, la teoría de agentes y el resto de áreas, tecnologías y desarrollos relacionados con la presente investigación, para obtener un marco teórico que permita enriquecer el conocimiento y mejorar el proceso de desarrollo.

Seguidamente se define una arquitectura multiagente orientada a la construcción de sistemas inteligentes en entornos sensibles al contexto. Partiendo de la información obtenida de las actividades anteriores, se diseña y desarrolla un modelo que integra los componentes necesarios para proponer una solución útil y novedosa a la problemática definida, siguiendo los objetivos planteados.

A continuación se evalúa la propuesta. Se implementa un sistema sensible al contexto inteligente en el hogar de una persona dependiente, utilizando la arquitectura multiagente propuesta.

Finalmente se lleva a cabo el análisis y la evaluación de resultados y obtención de conclusiones. Se valora la incorporación de los agentes y sistemas multiagente al

desarrollo de sistemas sensibles al contexto, y se extraen conclusiones de los resultados obtenidos tras aplicar la arquitectura multiagente propuesta a un caso de estudio real.

1.4 Estructura de la memoria

Para probar la hipótesis de partida y alcanzar los objetivos establecidos, se ha estructurado esta memoria en ocho capítulos y dos apéndices, que se organizan en cinco partes.

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 sensibles al contexto, así como de la necesidad de obtener una mayor flexibilidad de estos sistemas al reutilizar funcionalidades y distribuir los recursos. Se presentan los objetivos, la hipótesis y la motivación que han llevado al desarrollo de la arquitectura HoCCAC. Finalmente, se detalla la metodología de investigación aplicada y se realiza una breve descripción de esta memoria de investigación.

La segunda parte presenta el estado del arte e integra los capítulos 2 y 3 de esta memoria. En el capítulo 2, se analiza el estado del arte de la computación sensible al contexto. Se describen los aspectos y conceptos más significativos y se presentan algunas de las áreas relacionadas con esta nueva área multidisciplinar. Se analiza la problemática en torno a las personas dependientes, al continuo envejecimiento de la población y las soluciones aportadas por sistemas sensibles al contexto desarrollados hasta ahora. Finalmente, se analizan las tecnologías *Wireless Fidelity* (WiFi), ZigBee, *Radio Frecuency IDentification* (RFID) y NFC todas ellas con el potencial para aportar a HoCCAC 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 y planificación capaces de integrarse con dichos agentes. Asimismo, se estudian los trabajos y tendencias existentes en el campo de la integración de sistemas multiagente y entornos sensibles al contexto, describiendo en este capítulo algunos de

los desarrollos que incorporan sistemas multiagente en entornos sensibles al contexto y de dependencia.

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

La cuarta parte, correspondiente al capítulo 5, presenta el caso de estudio de un sistema multiagente que hace uso de la arquitectura HoCCAC. El caso de estudio trata de mejorar los cuidados y la asistencia a pacientes EPOC en su hogar. Se hace uso de de la metodología orientada a agentes SysML [SysML.org, 2010] para definir los principales componentes que integran el sistema.

La última parte de la memoria se encuentra formada por las conclusiones y el trabajo futuro, un resumen de todo el trabajo de investigación en inglés, la bibliografía y los apéndices. En el capítulo 6, se analizan los resultados obtenidos durante todo el proceso de investigación que han llevado al desarrollo e implementación de la arquitectura HoCCAC. A partir de los resultados obtenidos, se presentan las conclusiones y el trabajo futuro, junto con algunas de las posibles líneas de desarrollo a seguir basándose en esta investigación. La memoria finaliza con el listado de las reseñas bibliográficas que han servido como referencia para el desarrollo de este trabajo de investigación, además de los apéndices que permiten detallar algunas tecnologías reseñadas a lo largo del trabajo de investigación y la metodología de análisis y diseño utilizada para desarrollar el caso de estudio propuesto.

2

Computación sensible al contexto

En este capítulo se analizan el concepto de contexto y algunos aspectos importantes de la computación sensible al contexto, como la captura de información de atributos de contexto y la forma de interactuar con el usuario en su entorno. La búsqueda de entornos de software que se adapten cada vez mejor a las demandas de los usuarios y su entorno nos conducen hacia los sistemas sensibles al contexto. Por este motivo en este capítulo se hace una revisión de sistemas sensibles al contexto. Estos sistemas almacenan y analizan toda la información relevante que los rodea y forma parte del entorno de los usuarios. Las preferencias del usuario, sus gustos, su ubicación, su estado de ánimo, su actividad, su entorno, la temperatura de la estancia en la que se encuentra, las condiciones de iluminación etc., componen la información que puede ser clasificada de inicio como información de contexto. El problema al que se enfrentan estos sistemas es el manejo de la información del entorno para proporcionar respuestas adecuadas en función del estado contextual. Los sistemas sensibles al contexto proporcionan mecanismos para el desarrollo de aplicaciones que

conocen sus contextos y son capaces de adaptarse a posibles cambios. Una aplicación sensible al contexto (*context-aware*) utiliza su entorno para modificar su comportamiento y así satisfacer mejor las necesidades del usuario o entidad. La información se adquiere generalmente por medio de sensores. La tendencia actual para mostrar información a los usuarios del sistema, dado el gran número de dispositivos pequeños y portátiles, es la distribución a través de sistemas heterogéneos y redes de información variadas. Estos sistemas se mezclan en la vida cotidiana del usuario hasta llegar a hacerse indistinguibles de tal forma que los usuarios se puedan centrar en las tareas que deben hacer y no se preocupen por las herramientas que tienen que utilizar. Todos los elementos u objetos integrados en el contexto no deben interferir en las actividades para las que son usados y deben simplificar dichas actividades. Así pues, en esta introducción al capítulo, se va a comenzar definiendo contexto.

Este capítulo está estructurado de la siguiente manera: En primer lugar, se hace una introducción a la computación sensible al contexto, describiendo conceptos básicos y los elementos más importantes que componen esta área, así como sus principales aportaciones a la sociedad. A continuación se hace una revisión de las distintas áreas para las que se han desarrollado aplicaciones *context-aware*. Posteriormente, se presenta un análisis sobre el rol de la computación sensible al contexto en entornos de dependencia. Después, se describen algunas de las principales áreas relacionadas con la computación sensible al contexto, entre las que destaca la domótica, un área que aporta la infraestructura tecnológica necesaria para que la computación sensible al contexto desarrolle gran parte de su potencial. Luego, se presenta un breve análisis sobre la importancia de las tecnologías inalámbricas en desarrollos basados en la computación sensible al contexto y la relación de la computación sensible al contexto y otras áreas como la inteligencia ambiental, la computación ubicua y la computación pervasiva. Finalmente, se presentan las conclusiones correspondientes a este capítulo.

2.1 La computación sensible al contexto. Conceptos Básicos.

Las tecnologías de la información y comunicación están integradas en prácticamente todas las tareas de nuestras vidas, facilitando el desarrollo de las mismas y mejorando la calidad de vida [Miori, *et al*., 2005] [Aarts, *et al*., 2006]. No obstante, cada vez es mayor la necesidad de utilizar técnicas y conceptos contemplados en el ámbito de la computación sensible al contexto. Cualquier acción o entidad está ligada a un contexto. Una acción no puede ser descrita de manera detallada porque siempre hay información implícita [Benerecetti, *et al*., 2007]. Una acción depende de parámetros contextuales, como por ejemplo la temperatura o humedad del contexto, que forman parte del entorno, pero estos parámetros no se especifican en el desarrollo de la acción dentro de su contexto.

2.1.1 Definición de Contexto

El contexto se define como cualquier información que puede ser usada para caracterizar la situación de un usuario, lugar u objeto. Esta información se considera relevante en la interacción entre un usuario y una aplicación, incluyendo al usuario y a la aplicación [Dey *et al*. 2009]. La información del contexto cambia dinámicamente durante la ejecución de aplicaciones y se puede clasificar en:

- Información de contexto del usuario referida a la localización, preferencias y perfil del usuario.
- Información del contexto físico referida por ejemplo a las condiciones de iluminación, nivel de ruido o humedad del contexto.

De esta manera, para capturar la información de un contexto es necesario controlar los dispositivos físicos (sensores) instalados en el contexto. La información manejada debe estar debidamente estructurada y clasificada para facilitar su procesamiento. Las aplicaciones sensibles al contexto según Büscher *et al.* (2009), pueden obtener, interpretar y utilizar la información del contexto para adaptar sus funcionalidades. Las características más representativas presentes en las aplicaciones de contexto son [Hakkila y Mantyjarvi, 2005]:

- El usuario tiene que percibir de forma actualizada toda la información y servicios disponibles en el contexto.
- La ejecución automática de actividades para adaptar el contexto a las preferencias del usuario.
- El almacenamiento de información contextual para su posterior recuperación y uso.

Las aplicaciones sensibles al contexto actúan en entornos cambiantes. En estos entornos cambiantes la disponibilidad de recursos y de servicios puede cambiar de manera significativa durante la ejecución de las aplicaciones [Büscher *et al.,* 2009]. Por este motivo las aplicaciones sensibles al contexto tienen que ser capaces de adaptarse a los cambios. Así las aplicaciones se reconfiguran ellas mismas dinámicamente dependiendo del contexto y del usuario [Martínez-Ruiz *et al.,* 2008]. De esta forma se consigue un mayor nivel de satisfacción para el usuario [Sheng *et al*. 2009].

Varios autores han definido la palabra contexto dentro del ámbito de la Computación Ubicua, concepto que se trata posteriormente en este trabajo de investigación, y con el paso del tiempo esta definición ha ido evolucionando hasta formas más generales. Schilit *et al*., (1994) limita el concepto de contexto a la información de localización, ya que es una de las principales fuentes de información. De todas formas la información de localización es una de las principales fuentes de información asociadas a un contexto, pero no la única. Ryan *et al.,* (1998) y Brown *et al.,* (1997) incorporan al concepto de contexto los componentes de usuario, condiciones atmosféricas, estaciones o periodos del año, etc. Dey *et al.*, (2009) incorpora además al concepto de contexto, estados referidos al usuario como son su orientación, grado de atención o ánimo del usuario, además de las condiciones que lo rodean. Todas las fuentes de información que se van incorporando con el paso de los años al concepto de contexto van completando la definición, pero realmente se centran en entornos muy concretos de aplicación. Sistemas como los *stick-e notes* [Brown, *et al*., 2006] ya comienzan a describir el contexto de usuario y todos los componentes que lo rodean. Al mismo tiempo que cambia el concepto de Computación Ubicua, el concepto de contexto se completa aún más y autores como

Henricksen *et al.,* (2006) proponen descripciones más generales. Para Henricksen el contexto puede describirse respondiendo a las preguntas ¿Dónde? ¿Con quién? y ¿Qué? Otras definiciones [Prekop, *et al*., 2003] describen el contexto como un conjunto de componentes complejos que contienen información relativa al problema o dominio que se está tratando.

Una buena definición de contexto la dan Abowd *et al.,* (2007), quienes explican que el contexto es todo tipo de información que pueda ser utilizada para describir la situación de una entidad. Según Abowd *et al.,* (2007) una entidad es una persona, lugar u objeto que es importante para la interacción entre un usuario y una aplicación. Esta definición engloba cualquier tipo de información referente a factores climatológicos, localización, estado de ánimo, etc. y da pie a nuevas informaciones de contexto relevantes. Otra definición de contexto más general puede ser: *"El conjunto de información que describe los componentes que forman un entorno y su estado, siendo estos componentes personas, lugares u objetos considerados relevantes para el entorno."*

2.1.2 Tipos de contexto

Una vez definido el contexto de manera general, se han diferenciado diversas categorías de contexto para ayudar a estructurarlo de una manera organizada. Ryan *et al.,* (1998) clasifican la información de contexto en los elementos de:

- Localización
- Entorno
- Identidad
- Tiempo

Esta clasificación enumera componentes muy importantes en información, sin embargo no engloba todo el contexto al que se refieren las definiciones más completas. Por ejemplo, el estado de ánimo de un usuario es una información propia de contexto pero es difícil de encajar en esta clasificación. Además, la palabra entorno es demasiado ambigua y puede ser hasta un sinónimo de la palabra contexto. Abowd *et al.,* (2007) tienen en cuenta la clasificación anterior, pero sustituyen la palabra entorno por actividad, que da una información más concreta sobre lo que se hace en el entorno.

Por otro lado, para Henricksen *et al.,* (2006) el contexto se clasifica en diferentes tipos, dependiendo de sus características:

- Contexto de usuario: Tanto su localización, como estado de ánimo, situación social, etc.
- Contexto de recursos informáticos: Aplicaciones disponibles en los ordenadores, como procesos ejecutándose, espacio libre, memoria, dispositivos conectados, etc.
- Contexto del entorno: temperatura, nivel de ruido, etc.

Esta clasificación es más amplia y puede englobar mucha más información de contexto. Tanto esta versión como las anteriores, vistas hasta ahora, se refieren al origen del contexto, es decir, la información está clasificada dependiendo de a qué componente pertenece la información. Por ello son clasificaciones referidas a la estructura que compone la definición de cada tipo de contexto. Se pueden distinguir otro tipo de clasificaciones que estructuran la información de contexto según sea la información que se recoge. Algunos autores [Prekop, *et al*., 2003] proponen clasificar el contexto en lo que llaman interno y externo.

- El contexto externo es la información recogida por sensores, de una forma concreta, por ejemplo temperatura, localización, luz, sonido, movimiento, etc. En este caso el contexto está muy relacionado con el entorno que rodea al usuario.
- El contexto interno es la información que describe el estado del usuario, por ejemplo el estado de ánimo, objetivos e intenciones o preferencias.

Puede verse fácilmente que el contexto interno es más complicado de obtener y además en muchas ocasiones está influenciado por el contexto externo. Esta clasificación se complementa con los tipos vistos anteriormente, ya que un componente de contexto puede clasificarse de distintas maneras. Por ejemplo, una información puede ser externa y a su vez de localización. Hay otras clasificaciones como la propuesta por Hofer *et al.* (2003) que distinguen entre contexto lógico y físico:

- El contexto físico, se refiere a la información del contexto en general. Es información captada por sensores y normalmente se actualiza frecuentemente

para tener siempre un valor de la situación actual. Es muy similar al contexto externo.

- El contexto lógico está relacionado con el contexto físico. Se obtiene de razonar y procesar el contexto físico. Es más difícil de obtener, más complejo y con mucho más significado. Los estados de ánimo de un usuario o el grado de adecuación de un contexto a unas características concretas son ejemplos de contexto lógico.

Donghai *et al.,* (2007) proponen otra clasificación, en la que diferencian entre contexto de alto nivel y contexto de bajo nivel. Esta es una clasificación que abarca un ámbito más amplio:

- El contexto de bajo nivel es toda la información de contexto recogida directamente de los dispositivos instalados en el sistema. Los valores que se capturan vienen directamente como se desean y por tanto el nivel de complejidad es bajo. La información capturada no se procesa de ninguna manera.
- El contexto de alto nivel se obtiene a partir de razonar el contexto de bajo nivel y obtener informaciones relevantes. Por sus características, contiene más información y resulta más interesante para las aplicaciones.

La propuesta de Henricksen *et al.,* (2006) es una buena clasificación orientada al significado de la palabra y relacionada con la estructura de la propia palabra, mientras que Donghai *et al.,* (2007) es una forma general de estructurar el contexto dependiendo de su nivel de complejidad. Por otro lado, Gwizdka *et al.* (2007) hacen un análisis del contexto y describe cuatro usos principales que las aplicaciones hacen de él:

- Contexto como elemento que proporciona información: El contexto suministra información que la aplicación utiliza en su beneficio. Por ejemplo, se puede utilizar la temperatura de una habitación para ajustar las preferencias de confortabilidad de un usuario en su hogar.
- Contexto como elemento que interpreta la entrada de información: El contexto es información que sirve para distinguir entre distintas situaciones de usuario de forma correcta. Un ejemplo puede ser, a partir del grado de luminosidad en una habitación y la hora del día se puede distinguir el tipo de tareas que realiza el usuario.

Contexto como elemento que proporciona información al usuario para que este interactúe sobre el contexto: La información es para que el propio usuario la interprete y pueda interactuar sobre el sistema. Un ejemplo puede ser el aviso de mal funcionamiento de un dispositivo. Esa información se envía al usuario para que él la interpreta y decida qué acción tomar sobre el sistema.

- Contexto como lanzador de eventos: El sistema recibe la información de contexto, la analiza y si se cumplen una serie de condiciones se ejecuta una determinada acción. Un ejemplo puede ser cuando está activada la calefacción y el sistema detecta una alta temperatura en el exterior. El sistema lanza un evento que desactive la calefacción.

2.1.3. Aplicaciones sensibles al contexto

Una vez descritas las definiciones y clasificaciones realizadas por distintos autores acerca del contexto, es necesario definir lo que son las aplicaciones *context-aware*, es decir, aplicaciones sensibles del contexto. Del mismo modo que la definición de contexto ha evolucionado de una forma más concreta a una más general, la definición de aplicaciones *context-aware* también ha ido evolucionando.

Henricksen *et al.,* (2006) lo definen como un tipo de software que se adapta al entorno dependiendo de los objetos, los usuarios y los cambios de estado de esos objetos y usuarios en el tiempo. Conociendo ya en este punto varias definiciones de contexto es fácil determinar que esta definición de aplicaciones *context-aware* es muy concreta. En ella se tiene en cuenta poca información relativa al contexto, además de decir que únicamente se adaptan a un entorno.

Autores como Jbara *et al*., (2007) definen el software *context-aware* de manera más completa y eliminando la unión existente en definiciones anteriores entre contexto y usuario. Utilizan el sinónimo *environment-directed*, y definen el software *context-aware* como aplicaciones que monitorizan el estado del entorno y adaptan su funcionamiento a dicho estado, siguiendo las preferencias definidas por el usuario. Henricksen *et al.,* (2006) lo define como sistemas que recogen cambios en el entorno y reaccionan a ellos. Sin embargo, una aplicación *context-aware* no tiene porqué reaccionar a los cambios que suceden en el entorno, sino que puede ofrecer únicamente información de contexto que no cambia.

Kurkovsky *et al*., (2005) hablan de sistemas que modifican su comportamiento dependiendo de la información de contexto que recojan en cada momento. Abowd *et al.,* (2007) lo define como un tipo de software que utiliza el contexto para suministrar información relevante al usuario. La relevancia de la información depende de las preferencias del propio usuario. Así pues, todas las definiciones vistas hasta el momento se centran en la información de contexto.

2.1.4 Tipos de aplicaciones sensibles al contexto

Tras analizar las definiciones de aplicaciones *context-aware*, podemos ver que es un concepto muy relacionado con la computación ubicua y la computación pervasiva, ambos conceptos se tratan dentro de este capítulo en los puntos 2.6.2 y 2.6.3 respectivamente. Para entender mejor el concepto de aplicación *context-aware*, es útil recurrir a la clasificación. Henricksen *et al*., (2006) realiza dos clasificacioens. La primera de ellas divide a las aplicaciones en dos tipos las que tienen como objetivo capturar información y las que pretenden mostrar información. La segunda categorización las divide dependiendo de si las tareas son ejecutadas automáticamente o de forma manual. Dependiendo de la categorización descrita, Henricksen *et al.,* (2006) definen cuatro tipos principales de aplicaciones:

- Selección por cercanía: este tipo de aplicación muestra en una interfaz los elementos contextuales cercanos e interesantes que el usuario puede seleccionar de forma manual. Este tipo de aplicaciones resultan útiles en contextos donde los elementos dependen de la cercanía del usuario. Así, un ejemplo puede ser una aplicación móvil que muestra la localización de un usuario y cuando él cambia de posición, la aplicación actualiza los elementos más cercanos.
- Reconfiguración contextual automática: En estas aplicaciones los elementos que constituyen un contexto se reconfiguran dependiendo del estado contextual. Dependiendo la acción que se quiera realizar en el contexto los componentes contextuales pueden desarrollar diferentes acciones. Un posible ejemplo es una vivienda en la que automáticamente cuando llega la noche se cierran las ventanas y persianas, se bloquean las puertas y se apagan todas las conexiones sin actividad.

- Acciones contextuales: Son aplicaciones que, a petición del usuario, ejecutan acciones que dependen del estado contextual. De esta forma, si un usuario baja la temperatura a la que desea estar a través de un dispositivo móvil, la aplicación *context-aware* baja la temperatura de referencia del termostato de la habitación donde esté localizado el usuario.
- Acciones lanzadas por el contexto: En este caso las aplicaciones ejecutan acciones automáticas dependiendo del estado contextual. Estas acciones están basadas en reglas de funcionamiento y preferencias de usuario previamente introducidas. Se da en sistemas donde, por ejemplo, la luz se enciende automáticamente al llegar una persona a la habitación.

Por otro lado Pascoe *et al.,* (2007) establecen que las aplicaciones *context-aware* se dividen en distintos tipos dependiendo de las características del propio contexto. La clasificación de Pascoe considera los siguientes tipos:

- Captura contextual: La forma más simple de aplicación en la que la información contextual se recoge de los sensores y se muestra de forma adecuada. La información no se procesa de ninguna forma, sólo se muestra de forma adecuada.
- Adaptación contextual: La aplicación *context-aware* se adapta dependiendo el estado del contexto. La información contextual se utiliza para determinar el curso de acciones a seguir en la adaptación del contexto.
- Descubrimiento contextual de recursos: Este tipo de aplicaciones tienen en cuenta los recursos disponibles en el contexto. Incorporan nuevos componentes al sistema que pueden ser gestionados por este y que pueden proporcionar información valiosa. Un ejemplo es una aplicación que localiza nuevos sensores inalámbricos que proporcionan información sobre el grado de humedad en un determinado contexto.
- Aumento contextual: Este tipo de aplicaciones son bastante más complejas y no se limitan sólo a recoger información contextual, realizar adaptaciones contextuales o reaccionar ante la información suministrada. Estas aplicaciones mejoran y complementan los datos que obtienen con información añadida de la realidad contextual. Un ejemplo podría ser una aplicación móvil que automáticamente activa los sensores de una habitación cuando detecta la

cercanía de un usuario. Estos sensores están en un reposo relativo para ahorrar energía hasta que son activados.

Este último tipo de aplicaciones *context-aware* añade un nuevo concepto que la primera clasificación de Henricksen *et al.,* (2006) no contempla, el aumento del contexto. Hacer que la información contextual se complemente y que incluso el propio contexto proporcione un valor añadido a la aplicación *context-aware*.

Brown *et al.,* (2006) proponen otra clasificación que se basa en la forma en que se modifica el estado del contexto. Para Brown las aplicaciones *context-aware* se dividen en dos tipos:

- Continuas: Que están ejecutándose en un contexto que cambia muy a menudo y es necesario actualizarlo en intervalos cortos de tiempo, como por ejemplo aplicaciones basadas en localización de un elemento con capacidad de movimiento.
- Discretas: Aquellas que se ejecutan en contextos donde los cambios son esporádicos y muy concretos. Como las que se basan en el control de la localización de objetos con muy poca capacidad de movimiento.

Las aplicaciones *context-aware* continuas tienen en muchos casos características de aplicaciones en tiempo real y son más complicadas de desarrollar. Por otro lado las aplicaciones discretas no tienen restricciones de tiempo y esto las simplifica.

Abowd *et al*., (2007) presentan otra clasificación que se basa en el uso de la información contextual. En este caso se platean las siguientes clases:

- Aplicaciones que muestran la información contextual al usuario: El usuario percibe la información del contexto como la ha solicitado. Esta información de contexto puede ser mostrada de acuerdo a unas pautas para poder ofrecer diferentes servicios a los usuarios.
- Aplicaciones que ejecutan servicios de forma automática: Utilizan la información de contexto para ejecutar servicios que dependen de las condiciones contextuales.
- Aplicaciones que usan la información de contexto para etiquetar datos: Incorporan información añadida a los datos sobre objetos que tiene el sistema.

Esta categorización de Abowd *et al*., (2007) permite que una misma aplicación pueda considerarse de varios tipos a la vez. No incorpora la adaptación de la

aplicación *context-aware* al contexto. Aplicaciones que se adaptan al contexto dependiendo la información que recojan, no se incluyen en esta clasificación y sí son aplicaciones *context-aware*. El último tipo presentado aporta un factor nuevo muy importante, que es el de etiquetar la información contextual. Este mismo concepto lo define Pascoe *et al.,* (2007) en la clasificación que propone y es el aumento contextual.

Una vez vistas distintas definiciones y clasificaciones del concepto de contexto y de aplicaciones *context-aware*, se puede decir, que el principal impulsor de la computación sensible al contexto "*context-aware computing*" son los avances tecnológicos [Baldauf, *et al*., 2004]. Cada vez son más importantes los soportes tecnológicos para facilitar una vida independiente y garantizar un envejecimiento saludable, así como para la integración laboral y social.

Además, como es posible apreciar por todo lo descrito hasta ahora en este capítulo, definir el concepto de contexto y aplicaciones sensibles al contexto no es sencillo, pero sí es posible identificar las principales características que son exigibles para obtener aplicaciones sensibles al contexto. Inicialmente, un entorno sensible al contexto necesita establecer un modelo del entorno y necesita reconocer los cambios que se producen en él. En ese momento es posible aplicar diferentes técnicas de Inteligencia Artificial que permitan obtener capacidades de aprendizaje y de adaptación. Así pues, un sistema sensible al contexto utiliza la información contextual de un usuario (localización, intereses, preferencias, características del entorno, etc.) y trata de ser capaz de aprender y auto-adaptarse a ese contexto concreto a través de sistemas inteligentes. Los objetivos que debe cubrir un sistema sensible al contexto comienzan con la captación de información y la adaptación de las propiedades del sistema al contexto. Además, el sistema sensible al contexto tiene que ser capaz de reconocer e integrar nuevos recursos de forma automática y mejorar los servicios ofrecidos por el contexto de manera transparente para el usuario.

En la siguiente sección se presenta una revisión de las aplicaciones *context-aware* realizadas como guías móviles en zonas turísticas, aplicaciones para áreas de trabajo, aplicaciones en campus universitarios que facilitan el aprendizaje, aplicaciones de apoyo en la gestión de datos, aplicaciones para mejorar la calidad de vida en el hogar, aplicaciones asistenciales y de cuidado de salud, aplicaciones para casos de

emergencia, aplicaciones para gestión de transportes y logística, aplicaciones para comercio y ocio, aplicaciones para deporte y entrenamiento y aplicaciones como juegos ubicuos.

2.2Revisión de aplicaciones context-aware.

Los avances en las tecnologías de localización, el desarrollo de nuevas interfaces para representar información contextual o la definición de nuevas arquitecturas software han permitido utilizar una gran cantidad de posibles entornos de aplicación. De la misma manera que las tecnologías avanzan los sistemas sensibles al contexto han crecido y ofrecen novedosas y variadas formas de ser aplicados.

Turismo

Hay varios proyectos que se han centrado en el desarrollo de guías móviles, capaces de proporcionar al usuario información relevante dependiendo de su situación y objetivos. Por ejemplo Baus *et al.,* (2005) describe una revisión de guías móviles turísticas, entre las que están Cyberguide o GUIDE. Se han desarrollado también guías para museos [Raptis *et al.,* 2005] y [Jbara *et al.,* 2007], que contienen una revisión de trabajos relacionados. Entre este tipo de trabajos están los proyectos Sotto Voce o Exploratorium. Por último, para este tipo de contextos, resaltar que en la actualidad existe una nueva posibilidad en este ámbito que es la combinación de guías móviles como las descritas con las tecnologías de realidad aumentada [Lee *et al.,* 2006]. La realidad aumentada puede mejorar la experiencia de comunicación entre una persona y una aplicación *context-aware* y hacer que esta comunicación sea interactiva.

Áreas de trabajo

Otro ámbito de las aplicaciones *context-aware* son los contextos de recogida masiva de información. Este tipo de contextos también requiere que la aplicación sea capaz de estructurar y organizar la información para su posterior tratamiento. Por ejemplo, el objetivo del proyecto Labscape [Artikis *et al.,* 2009] fue supervisar y apoyar las acciones llevadas a cabo en laboratorios de bioquímica donde la cantidad de información que se gestiona es muy grande. Por otro lado, Pascoe *et al.,* (2007)

desarrollan un prototipo de aplicación móvil para facilitar la recogida de información ecológica en contextos naturales.

Otro tipo de entornos para los que se han desarrollado aplicaciones *context-aware* son las oficinas o centros de trabajo. Por ejemplo, el *Context-Aware Office Assistant* [Yan *et al.,* 2009] es un agente software que se comunica con los visitantes y se encarga de gestionar la información del usuario propietario. Johanson *et al.* (2002) describen el proyecto iRoom como una sala de reuniones que ofrece características de realidad aumentada a través de pantallas, redes de comunicación y otros dispositivos. Tähti *et al.,* (2004) describen varios servicios *context-aware* entre los que destaca un servicio que adapta automáticamente un dispositivo móvil dependiendo el estado de su propietario y la agenda del mismo.

Aprendizaje

Otros entornos de aplicación para los sistemas *context-aware* son los campus universitarios y sus aulas. Por ejemplo, el proyecto ActiveCampus [Griswold *et al.,* 2004] tiene una aplicación móvil basada en localización que incorpora mensajería instantánea, información dinámica o localización de otros compañeros. El proyecto presentado por *Abowd et al.,* (2007) tiene el objetivo de crear una pizarra basada en la realidad aumentada en un aula. Las pizarras virtuales son capaces de recoger eventos y proporcionar a los alumnos nueva información al finalizar la clase. Por otro lado, Sadeh *et al.,* (2005) definen un entorno web que proporciona servicios contextuales. Este entorno web gestiona una serie de agentes con la capacidad de recoger información contextual relevante para el alumno dentro del ámbito universitario. También el proyecto MOBIlearn [Syvänen *et al.,* 2005] mejora el desarrollo de sistemas de apoyo al aprendizaje en otras circunstancias como por ejemplo, en visitas a ciudades turísticas o cursos extraordinarios. Esta aplicación se adapta de acuerdo a las sugerencias y contenidos asociados al contexto del usuario.

Gestión de datos

El hogar es un contexto en el que es más sencillo el uso e implantación de sensores, actuadores o redes inalámbricas por sus limitaciones de espacio. Bettini *et al.,* (2009) hacen una revisión importante de aplicaciones que denominan *context-aware home*. En particular, hay un creciente interés por las aplicaciones *context-aware* orientadas a

facilitar el desarrollo de actividades cotidianas [Fraile *et al.,* 2009] y al mantenimiento del bienestar personal.

Asistenciales

Siguiendo con el tema del desarrollo de actividades cotidianas, también se detecta un contexto de actuación en los entornos asistenciales y de cuidado de la salud. Especialmente importantes son las aplicaciones *context-aware* que monitorizan y cuidan de personas dependientes en su hogar o en residencias geriátricas [Bajo *et al.,* 2007] [Bajo *et al.,* 2010]. Siguiendo con contextos relacionados, las aplicaciones *context-aware* son útiles en centros hospitalarios, donde se puede optimizar la gestión de recursos y profesionales. Por ejemplo Bricon-Souf y Newman, (2007) utilizan este tipo de aplicaciones con fines logísticos, para organizar las gestión de recursos y medicamentos. También se están utilizando aplicaciones *context-aware* en la asistencia quirúrgica. Agarwal *et al.,* (2007) desarrollan un prototipo de sistema de información contextual para un quirófano. En este prototipo los datos se recogen a través de sensores asociados a personas, material y medicación. Toda la información recogida se utiliza para construir un contexto quirúrgico y detectar eventos médicos significativos. Son muchos más los sistemas *context-aware* relacionados con este tipo de contextos, es en el siguiente punto de este capítulo donde se explicará en detalle la relación existente entre las aplicaciones *context-aware* y los entornos de dependencia. Esto se debe a que la arquitectura que se presenta en el capítulo 4 de este trabajo de investigación está principalmente orientada a entornos de dependencia y más concretamente a mejorar la calidad de vida de personas dependientes en su hogar.

Las aplicaciones *context-aware* también pueden resultar muy útiles en situaciones de peligro extremo que requieren de una logística compleja. Por ejemplo, el sistema *Siren* [Jiang *et al.,* 2004] es una aplicación *context-aware peer-to-peer* que recopila, integra y distribuye los datos de contexto. *Siren* está especialmente orientada a la extinción de incendios. *Siren* facilita la generación de respuestas de emergencia con la información contextual. También Luyten *et al.,* (2006) desarrollan un sistema para bomberos que suministra información contextual en tiempo real a los bomberos para mejorar el desarrollo de sus tareas. Este sistema incorpora un sistema de razonamiento basado en reglas y dispone de interfaces especiales para ser utilizadas en casos extremos por los bomberos. Otra aplicación *context-aware* es WIISARD [Brown *et*

al., 2006], un sistema que mejora la respuesta médica en desastres. Esta aplicación trata de eliminar posibles errores relacionados con la falta de información contextual. Los pacientes son identificados mediante etiquetas que el personal médico puede leer con dispositivos electrónicos. Cuando el paciente está debidamente registrado la información asociada al paciente se distribuye entre todo el personal médico. De esta forma todo el personal médico está debidamente informado de los pacientes tratados, facilitándose los diagnósticos, tratamientos y la localización del paciente.

Transporte y logística

En contextos relacionados con el transporte y la logística se han desarrollado aplicaciones *context-aware* que proporcionan información al usuario en función de su localización y otras aplicaciones que monitorizan el traslado de objetos. Un ejemplo es la aplicación que describen Meier *et al.,* (2006) para apoyo a los usuarios en lo que se refiere a infraestructuras de transportes, servicios y señales de tráfico. El objetivo es poder ofrecer al pasajero la mejor combinación posible para moverse entre dos puntos. Esta aplicación proporciona información precisa al pasajero en estaciones de tren, metro o autobús y aeropuertos. En estos contextos se requiere información fiable y precisa que ayude al pasajero a tomar una decisión rápida. Otro tipo de aplicaciones en estos contextos es por ejemplo la que presentan Anumba y Aziz (2006), que está orientada a proporcionar servicios a los trabajadores de estaciones de metro, tren o autobús y aeropuertos. Esta aplicación realiza un seguimiento de las tareas que deben realizar los trabajadores a partir de la información contextual que recoge. Otro tipo de sistemas *context-aware* están orientados a la logística como el proyecto AwareGoods[1] que incorpora mecanismos de control para conocer información a cerca de la ruta que ha seguido un artículo en su transporte y el estado del propio artículo.

Ocio y deporte

El conocimiento del contexto del usuario cobra una importancia especial en entornos en los que el objetivo es seducir al comprador, ya que dicho contexto proporciona posibilidades de adaptación de publicidad y promociones. Por ejemplo, Kurkovsky *et al.,* (2005) describen una aplicación móvil, que se adapta al usuario y le envía a este promociones y publicidad teniendo en cuenta sus preferencias. La

[1] Proyecto Awaregoods: <http://www.comp.lancs.ac.uk/%7Ealbrecht/awaregoods/>

aplicación se denomina SMMART y se adapta al contexto del usuario, registrando sus hábitos de compra para poder enviar promociones y publicidad selectiva. Otro ejemplo de aplicación *context-aware* para centros comerciales es *RClick*. Esta aplicación que es un prototipo de una operadora móvil japonesa está basado en la localización del usuario. El prototipo conecta teléfonos móviles con otros dispositivos como etiquetas RFID, tarjetas chip, *SmartPhones* y todo tipo de pantallas localizadas en centros comerciales. Los usuarios reciben información de la zona comercial donde se encuentran a través de su terminal móvil. Del mismo modo, Bajo *et al.,* (2006) definen un sistema multiagente que con la ayuda de dispositivos RFID proporciona orientación a los clientes, en sus dispositivos móviles, sobre las instalaciones de ocio y sugerencias para ir de compras en centros comerciales. La localización del usuario en centros comerciales o tiendas también activa la posibilidad de asociar al usuario la descarga de contenidos multimedia u oferta de servicios. Por ejemplo, durante un evento deportivo, el usuario puede tener disponibles ciertos contenidos en su terminal móvil, adecuados a sus intereses, o sugerencias de próximos partidos y la posibilidad de adquirir las entradas con algún descuento. Una vez que cambie de localización, esos servicios pueden limitarse o simplemente, no estar disponibles. La relación entre servicios contextuales y eventos deportivos lleva a desarrollar aplicaciones como FLAME [Weiβenberg y Gartmann, 2004]. Esta aplicación se diseñó para ofrecer servicios contextuales en los juegos olímpicos de Pekín en base a la localización del usuario.

Otro entorno donde las aplicaciones *context-aware* pueden resultar útiles es en entrenamientos deportivos. En estos entornos los servicios contextuales pueden resultar muy útiles para optimizar y motivar entrenamientos. Actualmente ya existen aplicaciones que monitorizan entrenamientos ciclistas a través de dispositivos móviles como por ejemplo la aplicación *runKeeper*[2]. Se ejecutan sobre dispositivos móviles con GPS y también realizan un tratamiento posterior de los datos cuando finaliza el entrenamiento. Otras aplicaciones como la presentada por Buttussi y Chittaro (2008), supervisan la realización de entrenamientos en espacios abiertos. Además también

[2] http://runkeeper.com/

combina la explotación de los datos capturados con una base de conocimiento, para construir modelos personalizados de preparación física para los deportistas.

Tabla 2.1 Características principales de sistemas sensibles al contexto.

	Localización	Modelado de Información	Bienestar-Cuidados Médicos	Logística	Publicidad y Compras	Deportes	Juegos Pervasivos
GUIDE [Baus et al., 2005]	SI	---	---	---	---	---	---
Raptis et al., 2005	SI	---	---	---	---	---	---
Jbara et al., 2007	SI	---	---	---	---	---	---
Lee et al., 2006	SI	SI	---	---	---	---	---
Artikis et al., 2009	SI	SI	SI	---	---	---	---
Pascoe et al., 2007	SI	SI	---	---	---	---	---
Yan et al., 2009	SI	SI	---	---	---	---	---
Johanson et al., 2002	SI	SI	---	---	---	---	---
Tähti et al., 2004	SI	SI	SI	---	---	---	---
ActiveCampus [Griswold et al., 2004]	SI	---	SI	---	---	---	---
Abowd et al., 2007	---	SI	SI	---	---	---	---
Sadeh et al., 2005	---	SI	SI	---	---	---	---
MOBIlearn [Syvänen et al., 2005]	SI	SI	SI	---	---	---	---
Bettini et al., 2009	SI	SI	SI	---	---	---	---
Fraile et al., 2009	SI	SI	SI	---	---	---	---
Bajo et al., 2010	SI	SI	SI	---	---	---	---
Brincon-Souf y Newman, 2007	SI	SI	---	SI	---	---	---
Agarwal et al., 2007	SI	SI	SI	SI	---	---	---
Siren [Jiang et al., 2004]	---	SI	---	SI	---	---	---
Luyten et al., 2006	SI	SI	---	SI	---	---	---
WIISARD [Brown et al., 2006]	---	SI	SI	SI	---	---	---
Meier et al., 2006	SI	SI	SI	SI	---	---	---
Anumba y Aziz, 2006	SI	SI	SI	SI	---	---	---
SMMART [Kurkovsky et al., 2005]	---	SI	SI	---	SI	---	---
Bajo et al., 2006	SI	SI	SI	---	SI	---	---
FLAME [Weißenberg y Gartmann, 2004]	SI	SI	SI	---	SI	---	---
Buttussi y Chittaro, 2008	SI	---	SI	---	---	SI	---
The Song of the North [Ermi y Mäyrä, 2005]	---	---	---	---	---	---	SI
Fernández et al., 2007	SI	---	---	---	---	---	SI

Las particularidades e interactividad que puede ofrecer la información contextual a las aplicaciones *context-aware* de ocio hace que este ámbito también sea considerado como un tipo de aplicaciones contextuales particular. Un ejemplo de juego pervasivo es *The Song of the North* [Ermi y Mäyrä, 2005]. También hay juegos pervasivos que recogen parámetros ambientales. Por ejemplo Fernández *et al.,* (2007) presentan un videojuego para móviles que consiste en el desarrollo de una mascota con habilidades de comunicación mediante tecnología Bluetooth. En este caso la mascota está influenciada por parámetros ambientales como la hora del día o la meteorología del día.

Una gran parte de las aplicaciones presentadas son prototipos o aplicaciones de pruebas en entornos limitados [Artikis *et al.,* 2009], [Pascoe, 2007], [Johanson et al., 2002], [Sadeh et al., 2005], [Meier *et al.,* 2006], [Brown *et al.,* 2006]. Pero también hay muchas otras que se están utilizando en entornos reales como las relacionadas con guías turísticas [Raptis et al., 2005] y [Jbara et al., 2007], los servicios de gestión de pacientes y recursos en hospitales [Bricon-Souf y Newman, 2007] y [Agarwal *et al.,* 2007], aplicaciones para gestionar la logística de transporte de mercancías [Kurkovsky *et al.,* 2005] y [Anumba y Aziz, 2006], aplicaciones móviles para el control de entrenamientos deportivos [Buttussi y Chittaro, 2008] y los juegos pervasivos [Fernández, *et al.,* 2007], [Ermi y Mäyrä, 2005].

Como se puede ver en los estudios enumerados en la Tabla 2.1 la mayoría de sistemas *context-aware* se basan en la localización y ubicación de personas, objetos o entidades. Por otro lado, también hay una gran mayoría de estudios que utilizan la información de contexto que recogen para tratarla y ofrecer soluciones o ayudar a los usuarios en sus tareas cotidianas. El bienestar y los cuidados a personas en el hogar, residencias o centros hospitalarios, es también un objetivo prioritario de muchos de los sistemas de la Tabla 2.1. Pocos sistemas son los que usan información de diferentes atributos de contexto y relacionan distintos tipos de datos para interactuar sobre usuarios o pacientes. Muchos de los sistemas *context-aware* analizados se centran en resolver un problema determinado. Por este motivo es importante el desarrollo de nuevas arquitecturas que faciliten el diseño y construcción de sistemas *context-aware* de manera estandarizada. En este trabajo de investigación se quiere dar un paso más y, además de tratar de utilizar el mayor número de atributos de contexto,

se quiere que toda la información recogida por el sistema propuesto sea almacenada y procesada para poder mejorar soluciones futuras. Así se consigue que el sistema sea eficaz y proactivo. De todas formas, aún son muchos los retos que presenta la generación de servicios contextuales viables comercialmente; no sólo se trata de aspectos técnicos, sino también de cuestiones sociales como la inquietud ante el acceso no debido a información personal o de actividad. La garantía de privacidad y seguridad en la adquisición, almacenamiento y manejo de información es básica para que estas aplicaciones se generalicen. En el siguiente punto de este capítulo se trata de profundizar más en uno de los objetivos que persigue este trabajo, que es el desarrollo de sistemas sensibles al contexto que mejoren la calidad de vida de las personas dependientes en su hogar.

2.3La computación sensible al contexto y los entornos dependientes

Es de sobra conocido que el envejecimiento de la población es una realidad en los países más desarrollados del mundo que, además, se prevé que evolucione al alza en los próximos años; sus consecuencias, sociales y económicas, son un reto para la concepción actual de los sistemas nacionales de salud y bienestar [Malanowski et al., 2008] los cuales tratan de garantizar la mejor asistencia a sus mayores y personas dependientes de forma sostenible.

Se están desarrollando diferentes iniciativas, con el objetivo de utilizar las tecnologías de la información y las comunicaciones para minimizar y manejar los efectos del cambio demográfico. Al mismo tiempo se trata de aprovechar las oportunidades que pueden nacer para generar propuestas de sistemas innovadores relacionados con estos cambios demográficos. Estos nuevos sistemas deben intentar que las personas dependientes puedan vivir en su entorno preferido, incrementando su autonomía, confianza y movilidad. También deben apoyar el mantenimiento de las capacidades físicas y funcionales de las personas dependientes, favorecer estilos de vida saludables, mejorar la seguridad, prevenir el aislamiento social y soportar el funcionamiento de una red multifuncional alrededor del individuo. Finalmente es

necesario que presten apoyo a los cuidadores, familias y organizaciones, y mejoren la eficiencia y la productividad del uso de recursos en la sociedad.

Los sistemas basados en el concepto de Inteligencia Ambiental (AmI: *Ambient Intelligence*) [ISTAG, 2005], se aproximan mucho a los sistemas innovadores descritos anteriormente. El concepto de AmI que se trata más en detalle en el punto 2.6.1 de este capítulo, presenta un entorno en el que sensores, actuadores, sistemas de comunicaciones y de razonamiento, se integran de forma mínimamente invasiva y transparente para el usuario, con el objetivo de proporcionarle servicios adaptados, usables y seguros. Idealmente, los sistemas de AmI permiten movilidad, soportan comunicaciones ad hoc entre nodos, son heterogéneos y jerárquicos, sensibles al contexto del usuario, tienen capacidad de anticipación, se adaptan a situaciones anómalas y facilitan la interacción natural con los usuarios a través de voz y nuevas interfaces [Weber *et al.,* 2005].

La aplicación de los conceptos anteriores al cuidado de la salud ha dado lugar a diversos sistemas y prototipos [Bricon-Souf y Newman, 2007], entre los que figuran aquellos destinados a proporcionar asistencia en el hogar para personas con necesidades especiales, como mayores y discapacitados. Estos servicios son ofrecidos por sistemas conocidos como *Ambient Home Care Systems* (AHCS). Cuando los AHCS sirven para coordinar una red de cuidadores se convierten en sistemas *Computer Supported Coordinated Care* o CSCC [Consolvo *et al.,* 2008]. Sus funcionalidades son:

- La asistencia en caso de emergencia, para detectar, notificar y actuar en eventos críticos como caídas, golpes, ataques cardíacos o de pánico.
- La mejora de la autonomía, a través de servicios domóticos y de asistencia a tareas cotidianas, que eviten o disminuyan la necesidad de asistencia por terceros.
- El mantenimiento del confort y bienestar, concepto en el que se incluyen servicios de seguridad física, mantenimiento del hogar, servicios de control financiero, y también, gestión de las relaciones sociales y el ocio [Kleinberger *et al.,* 2007].

Hoy en día, la tecnología que proporcionan estos servicios está lo suficientemente avanzada para poner en marcha pilotos y prototipos [Abowd *et al.,* 2007]. Por supuesto, la tecnología puede mejorar, y de hecho lo está haciendo, y los servicios no pueden presumir todavía de comportamientos predictivos perfectos. Sin embargo, las dificultades de generalización de este tipo de servicios están por encima de los aspectos tecnológicos. Por un lado, el modelo de negocio, por otro lado las barreras de adopción como la pérdida de privacidad o la preocupación por los efectos secundarios del uso de la tecnología, hacen que se cree un sentimiento de ofuscación entre los usuarios, porque se sienten obligados a adaptarse al uso de interfaces complejos. En definitiva, los usuarios se encuentran una serie de problemas causados por diferentes factores:

- Muchos sistemas dejan a un lado la usabilidad y al usuario.
- Las tecnologías utilizadas muchas veces superan al usuario. Este hecho genera en los usuarios ansiedad, preocupación o frustración asociada al uso de la tecnología [Hensel *et al.,* 2006].
- La poca utilidad que en ocasiones tienen los servicios para solucionar problemas sencillos.
- La falta de adecuación al contexto y sistemas de razonamiento que no incluyen aspectos personales del usuario.
- La escasa escalabilidad e integración con sistemas externos, ya que la mayor parte de los sistemas son cerrados y no ofrecen posibilidad de conexión.
- Las interfaz estáticas y poco naturales, que no se ajustan suficientemente a la situación cognitiva de las personas.

Como se observa en la revisión del estado del arte que se incluye a continuación, muchos estudios van dirigidos a eliminar las barreras de mercado existentes entre los usuarios y la tecnología. Estas barreras son provocadas por aspectos relacionados con la usabilidad, el coste de los sistemas, la privacidad y la calidad de servicio. De hecho, los proyectos actuales hacen un esfuerzo especial en la validación de sus resultados con usuarios reales, a los que en algunas ocasiones también implican en el diseño de los productos o servicios.

En la literatura se encuentran pocos estudios formales y extensos dirigidos a analizar el interés y la aceptación de servicios sensibles al contexto en el hogar, y aún

menos dentro del colectivo de personas mayores. Por ejemplo Wu *et al.,* (2007) realizan un análisis de experiencia de usuario sobre su prototipo de servicios contextuales para *smart homes*. La población elegida para el estudio fueron doce individuos con educación superior, relativamente jóvenes y habituados al uso de los ordenadores, teléfonos móviles y servicios web. Las conclusiones recogen un conjunto de lugares comunes. En estos espacios se destaca la importancia de las interfaces y de los métodos de posicionamiento en interiores para obtener una experiencia de usuario gratificante. Mientras que en otros entornos no sucede así, en el hogar los usuarios son en general reticentes a llevar cualquier tipo de dispositivo encima, lo cual condiciona tanto los métodos de posicionamiento como la recepción de información. Por otra parte, algunos usuarios no se muestran dispuestos a distribuir algunas pantallas táctiles en lugares como la cocina o la entrada de la casa. En cuanto al tipo de servicios, destaca el interés por los de seguridad y también por los asociados al control de la salud.

Los usuarios prefieren servicios con capacidad de predicción, que les proporcionan sensación de control, a servicios totalmente automáticos. El equilibrio entre la automatización y el control humano es uno de los aspectos que condiciona la frustración o satisfacción del usuario [Emiliani y Stephanidis, 2005]. También mencionan su escepticismo en lo relativo a la privacidad.

Niemelä *et al.,* (2007) recogen los resultados de un estudio de evaluación de servicios de inteligencia ambiental para personas mayores. El estudio se realiza con dos grupos formados por personas mayores de Finlandia y España. Entre sus conclusiones se destaca que independientemente de su nacionalidad y cultura tecnológica, los mayores apreciaron la propuesta de servicios de seguridad física. Por su parte, los servicios de monitorización no son aceptados, salvo en caso de emergencias, debido al sentimiento de intromisión en la privacidad que suponen. Los usuarios aprueban el teléfono móvil como plataforma para transferir datos y recibir alertas. Los individuos de la muestra resultan estar más familiarizados con la tecnología móvil que con internet, por ejemplo. No obstante, se ponen de manifiesto los problemas relacionados con la usabilidad para los usuarios menos familiarizados en el uso de los dispositivos móviles. Las personas mayores entrevistadas expresan su preocupación por un factor que habitualmente no se considera al diseñar servicios de

asistencia. Este factor es el impacto sobre las capacidades cognitivas del usuario, que al verse arropado por diferentes tipos de recordatorios, dejan de verse obligados a utilizar su memoria. Emiliani y Stephanidis, [Emiliani y Stephanidis 2005] recalcan la existencia de estas consecuencias colaterales y plantean la necesidad de estudiar cómo las funciones humanas se ven afectadas tanto en el espacio físico como en el cognitivo con el uso de estos sistemas. Desde otra perspectiva, el estudio también recoge las opiniones de algunos profesionales de la salud (médicos y enfermeros), que destacan que la monitorización remota no debe sustituir a las visitas clínicas. Del trabajo de Niemelä *et al.,* (2007) se extrae la importancia de la segmentación de los receptores de los servicios, dentro del mismo grupo de personas mayores. No sólo la cultura tecnológica y la tendencia a aceptar tecnología es importante, de algún modo, los mayores del futuro deben estar más acostumbrados al uso de la tecnología y así reducir la brecha tecnológica que existe. El colectivo de personas mayores es muy heterogéneo. Está compuesto por personas que viven independientes en sus casas y no tienen sus capacidades físicas o psíquicas mermadas, y también está compuesto por personas mayores con deterioro cognitivo iniciado, que pueden habitar en apartamentos tutelados, o con discapacitados motores, que pueden utilizar una silla de ruedas si la silla está dotada de los sistemas de seguridad pertinentes, por ejemplo.

Por tanto, es necesario discriminar entre los tipos de usuario a los que se dirigen los servicios y establecer un usuario receptor del servicio. Por ejemplo, se han propuesto soluciones asistenciales para usuarios con enfermedades concretas, como son los enfermos de Alzheimer [Corchado *et al.,* 2008a]. Estos enfermos tienen unas necesidades peculiares relacionadas con la pérdida súbita de memoria y la desorientación. Para mejorar su calidad de vida, Fernández *et al.,* (2007b) han desarrollado un sistema específico que ayuda a superar las crisis de desorientación indicando al usuario dónde se encuentra y mostrando fotografías asociadas a los posibles lugares de destino. Intel, a través de su *Proactive Health Team*, en los últimos años ha desarrollado soluciones probadas en familias reales para facilitar la convivencia con esta enfermedad en el hogar. La tecnología también puede ayudar a enfermos de Parkinson: Yoerger *et al.,* (2005) presentan un sistema diagnóstico basado en acelerómetros que, colocados en lugares determinados del cuerpo, pueden llevar a ajustar la medicación en función de la situación motriz del paciente.

Como describen Aaløkke *et al.,* (2006), idealmente la tecnología y los sistemas deben evolucionar con las necesidades del usuario y adaptarse de forma sencilla sin tener en cuenta el tipo de usuario. De esta forma estos sistemas aprovechan el habitual proceso costoso de familiarización y recorren así más rápidamente la curva de aprendizaje. De esta forma hay autores [Pecora, *et al*., 2007] [Corchado, *et al*., 2006] que 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 pueden proveer apoyo constante en su vida diaria [Cesta, *et al*., 2007], prediciendo situaciones potencialmente peligrosas y entregando un soporte cognitivo y físico para la persona asistida [Bahadori, *et al*., 2007]. Existen desarrollos que hacen uso de agentes inteligentes en sistemas orientados a la automatización de servicios o para el cuidado y monitorización de personas. *RoboCare*, apoyado por el Ministerio Italiano de Educación, se centra en proporcionar asistencia a ancianos a través de un entorno multiagente, utilizando software, robots, sensores y humanos [RoboCare, 2005] [Bahadori, *et al*., 2007]. Una parte importante de este proyecto es el desarrollo de software y herramientas de visualización para sistemas con múltiples robots. El proyecto *RoboCare* apoya directamente el desarrollo de diversas líneas de investigación, entre ellas: sistemas multiagente, interacción humano-computadora, monitorización y diagnóstico, supervisión frente a autonomía. 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*., 2008]. *Smart Home Technology* [Angulo, *et al*., 2004] es una propuesta centrada en la automatización de servicios del hogar mediante el desarrollo de 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. A pesar de los avances en la implementación de agentes en entornos automatizados, los sistemas existentes tienen muchas carencias, aún hace falta mucha investigación y proponer nuevas soluciones.

La Tabla 2.2 presenta un resumen de los sistemas sensibles al contexto enumerados en este apartado con sus principales características. Como se muestra la mayoría de trabajos analizados están enfocados a los cuidados generales de los pacientes, tanto en entornos relacionados con el hogar como en residencias geriátricas u hospitales. De los sistemas descritos son solamente tres los que están enfocados a cuidados específicos, dos orientados a mejorar la calidad de vida de enfermos de Alzehimer [Corchado *et al.,* 2008a] [Fernández *et al.,* 2007b] y uno orientado a los enfermos de parkinson [Yoerger *et al.,* 2005]. El resto de trabajos está orientado a los cuidados generales de los usuarios. Por otro lado, se puede ver que casi todos los sistemas analizados apuestan por las tecnologías móviles como elemento práctico y útil para el cuidado y atención de pacientes. También comienza a ser significativo el número de sistemas que se basan en la tecnología de agentes.

Tabla 2.2. Sistemas sensibles al contexto en entornos dependientes.

	Cuidados Generales	Cuidados Específicos	Home Care	Cuidados en Resisdencias u Hospitales	Tecnología Móvil	Tecnología de Agentes
Bricon-Souf y Newman, 2005	SI	---	SI	---	---	---
Abowd et al., 2007	SI	---	---	---	SI	---
Wu et al., 2007	SI	---	SI	---	SI	---
Niemalä et al., 2007	SI	---	---	---	SI	---
Corchado et al., 2008 a	---	SI	---	SI	SI	SI
Fernández et al., 2007b	---	SI	---	---	SI	---
Yoerger et al., 2005	---	SI	---	SI	SI	---
Bahadori et al., 2007	SI	---	---	SI	---	SI
TeleCARE [Camarinha-Matos, et al., 2008]	SI	---	SI	SI	SI	SI
Smart Home Technology [Angulo, et al., 2004]	SI	---	SI	---	SI	SI

La arquitectura multiagente que se describe en el capítulo 4 de esta memoria ofrece servicios *context-aware* a usuarios en contextos particulares, e incluye un conjunto de servicios independientes, que realizan colecta e interpretación de datos de contexto. La característica fundamental del sistema es la de procesar y razonar los datos que proporciona el contexto para mejorar la calidad de vida de personas dependientes en su hogar a través de planes de tareas óptimos. Aún así, las personas en general se oponen a la idea de que un ordenador o un sistema tome decisiones por ellos. Todo esto, unido a un mayor coste económico, la complejidad y el tiempo requerido para su desarrollo, son los principales obstáculos para la aceptación social y la implementación de este tipo de propuestas. Sin embargo, con la aparición de nuevas tecnologías no intrusivas, el uso de sistemas multiagente aplicados en entornos cotidianos va en aumento. Ya no sólo se plantea la domótica como conjunto de sistemas que tratan de mejorar la calidad de vida de los usuarios. Se buscan sistemas inteligentes, como el propuesto en el capítulo 4 de esta memoria, que se adelanten en la detección y propongan soluciones efectivas a cualquier problema en el desarrollo de las actividades contidianas. Por tanto, se puede decir que la computación sensible al contexto, a través del concepto de computación ubicua, facilita el planteamiento de sistemas inteligentes que permiten la temprana detección y prevención de problemas en entornos automatizados y de dependencia. De todas formas es importante tener presentes los sistemas domóticos, como sistemas que mejoran la calidad de vida de las personas, y por eso en el siguiente punto de este capítulo se analiza la estrecha relación existente entre la computación sensible al contexto y la domótica.

2.4 La computación sensible al contexto y la domótica

En la actualidad, el nivel de confort deseado en el hogar se está incrementado considerablemente con el descubrimiento de nuevos dispositivos tecnológicamente muy avanzados. Los sistemas de calefacción y refrigeración se consideran básicos actualmente en cualquier hogar. A estos sistemas se añaden robots de aspiración centralizada, sistemas de ventilación mecánica, sistemas automáticos para persianas y puertas de garaje o aplicaciones móviles para interactuar con los dispositivos instalados en el hogar, por poner algunos ejemplos.

Por otro lado la energía, cada vez más escasa y con mayor coste, junto al elevado índice de contaminación ambiental plantea la necesidad de sistemas de gestión energética, optimización de recursos, racionalización de consumos, etc. Así mismo, las formas de vida actuales; trabajo, ocio, etc., están dejando, durante horas las casas vacías, siendo la seguridad uno de los factores más tenidos en cuenta hoy en día, como ya se ha descrito en puntos anteriores de este capítulo. Debido a todo lo anterior, los fabricantes de materiales eléctricos llevan algunos años fabricando productos que permiten activar la calefacción en base a parámetros contextuales para gestionar la energía y racionalizarla aun mejor [Lui, 2009]. Estos parámetros son la temperatura exterior o la ocupación de los locales, por ejemplo. A todas estas funciones se han sumado nuevas instalaciones singulares de vigilancia y comunicación, alarmas técnicas, alarmas anti intrusión, tele vigilancia, etc. Por lo tanto, las instalaciones se han vuelto más complejas, y cualquier modificación asociada a la instalación se convierte en una compleja intervención. Las viviendas así se convierten en hogares inteligentes con capacidades de toma de decisiones. De todas formas, el usuario no pierde nunca el control de la vivienda.

Muchas veces se utiliza el término inteligente sin justificar su uso y su significado. Para que un sistema se considere inteligente tiene que disponer de sistemas basados en las nuevas tecnologías de la información, como sistemas de tratamiento y razonamiento de la información. El uso de las nuevas tecnologías de la información en hogares hace surgir nuevas aplicaciones. Estas aplicaciones poseen capacidad de proceso de la información y comunicación entre los equipos y las instalaciones [Molina *et al.,* 2004]. Una vivienda domótica es aquella que tiene instalados sistemas de medida, orden, regulación y control de las funciones que intervienen en dicha vivienda o contexto. De esta forma se obtiene, información de las variables del entorno como temperatura o luz por ejemplo. Así una vez tratada toda la información recogida, se dan una serie de órdenes para modificar dichas variables.

Al mismo tiempo la domótica juega un papel muy importante dentro de los sistemas sensibles al contexto, 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.

Se define domótica como el conjunto de sistemas que automatizan las instalaciones del hogar [Salsbury, 2009]. La domótica se centra en cuatro objetivos básicos:

- Energía: El ahorro de energía es un objetivo importante y por ello una vivienda domótica tiene control energético de los sistemas que la forman.
- Confort: La automatización de una vivienda ofrece una mejor calidad de vida a los usuarios de la instalación, a través de una reducción del trabajo doméstico, un aumento de bienestar y tiempo libre para las personas que viven en la casa, una mayor limpieza, formación cultural y entretenimiento. Además la domótica automatiza el mantenimiento de las instalaciones.
- Seguridad: La seguridad de una vivienda siempre ha sido una preocupación para sus dueños, un sistema domótico es una gran ayuda para lograr este objetivo.
- Comunicaciones: La comunicación entre los distintos dispositivos en una vivienda domótica permite una comunicación interactiva con el usuario. De esta forma la vivienda ofrece servicios a distancia sin necesidad de que el usuario vaya directamente al interruptor general.

Así pues, 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 [Lui, 2009]. 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.

Por otro lado, como ya se ha visto, la computación sensible al contexto está orientada fundamentalmente a los servicios al usuario como la domótica. Por ello el enfoque tecnológico de procesamiento independiente del contexto es aplicable en cualquier lugar y en cualquier instante, permitiendo desacoplar los usuarios de los dispositivos, y modelando las aplicaciones como entidades que realizan tareas en beneficio de los usuarios y el contexto. De este modo el usuario puede estar ubicado en un entorno privado, en un lugar público, o en un entorno colectivo. La capacidad de interactuar con el sistema depende de su perfil de usuario. La prestación de los

servicios sensibles al contexto se produce a instancia de cualquier agente autorizado del sistema o del propio usuario y la información que recibe el usuario está particularizada y adaptada a su contexto.

2.5 La computación sensible al contexto y los dispositivos inalámbricos

Para lograr el suficiente dinamismo y distribución de recursos en los desarrollos basados en la computación sensible al contexto, 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 *General Packet Radio Service* (GPRS), *Universal Mobile Telecommunications System* (UMTS), Bluetooth, Wi-Fi, NFC, etc., implementadas en dispositivos cada día más comunes, como dispositivos móviles. Estas tecnologías amplían las posibilidades de comunicación para los sistemas distribuidos, permitiendo una mayor flexibilidad para el desarrollo de aplicaciones sobre entornos inalámbricos.

Una red es un sistema de transmisión de datos que permite compartir recursos e información [Cisco Systems, 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. Hoy en día es mucho más frecuente el uso de redes inalámbricas [Cisco Systems, 2005], en especial bajo la tecnología WiFi. Una infraestructura WiFi incrementa la movilidad, la flexibilidad y la eficiencia de los usuarios, ya que permite 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. Una de las principales ventajas de las redes inalámbricas es su menor coste de implementación, presentando ahorro tanto en los componentes como en la instalación de la infraestructura.

Uno de los fundamentos en los que se basa la computación sensible al contexto es la necesidad de que el contexto tenga en cuenta la presencia de las personas, localizándolas en un medio, tanto geográfico, como de actividad. Por este 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), 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*., 2007] [Corchado, *et al*., 2008a]. 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. Para leer las etiquetas (*tags*) RFID se utilizan lectores sin contacto y dispositivos móviles con tecnología NFC [Vergara, *et al.,* 2010]. NFC es un protocolo basado en una interfaz inalámbrica de corto alcance. La comunicación se realiza entre dos entidades. El protocolo establece conexión inalámbrica entre las aplicaciones de la red y los dispositivos electrónicos. NFC trabaja en la banda de los 13,56 MHz, lo que permite que no se requiera de ninguna licencia para su uso. En NFC siempre hay un dispositivo que inicia la conversación y es este el que monitoriza la misma. La tecnología NFC fusiona la usabilidad de la tecnología chip RFID con la portabilidad e inmersión de mercado de la telefonía móvil. Consigue, a través de un campo electromagnético, comunicar de forma segura un teléfono móvil o portátil con una etiqueta inteligente (RFID) o con otro dispositivo para realizar micro pagos, intercambio de información y control de acceso.

Con respecto a los agentes y a los sistemas multiagente, las tecnologías inalámbricas 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*., 2006]. 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 [Fraile, *et al*., 2008]. Asimismo, RFID aporta una de las piezas más importantes en la computación sensible al contexto: 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*., 2008a].

2.6 Otras áreas relacionadas con la computación sensible al contexto

2.6.1 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, 2005] [Haya *et al*., 2005]. 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]. La AmI influye en el diseño de protocolos, comunicaciones, integración de sistemas, dispositivos, etc. [Reynolds, 2006], 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 [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 [Emiliani *et al*., 2005]. Las principales características que debe tener un sistema basado en la AmI son:

- 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], proponiendo una visión de los individuos rodeados de dispositivos e interfaces inteligentes asociados a objetos de la vida cotidiana [Emiliani *et al*., 2005] [Haya *et al*., 2005]. 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 necesidad de realizar un gran aprendizaje. 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 [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].

La Inteligencia Ambiental plantea una nueva forma de interactuar entre las personas y la tecnología [Tapia *et al*., 2009]. La tecnología 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]. 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 mejoran 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 sensibles al contexto. Así la mejora en la interacción entre los usuarios y el entorno es el resultado de una constante innovación que permite

acercarse cada vez más al fundamento principal de la AmI [Emiliani, *et al*., 2005] [Aarts, *et al*., 2006].

2.6.2 La computación ubicua.

La Computación Ubicua, también denominada Computación Pervasiva, fue descrita por primera vez por Weiser (1993). La idea que tenía Weiser era la creación de entornos repletos de computación y capacidad de comunicación, integrados de forma inapreciable con las personas. Cuando Weiser describió su idea no se disponía de los avances tecnológicos necesarios para concretar su idea, por lo tanto no era posible llevarla a cabo. Después de más de una década de progreso, las ideas son perfectamente viables. El objetivo principal de la Computación Ubicua es integrar los dispositivos computacionales en el contexto de tal forma que se mezclen en la vida cotidiana de forma no intrusiva. De esta forma, la Computación Ubicua trata de permitir que los usuarios se centren en las tareas que deben hacer, no en los dispositivos o herramientas que deben usar. Todo esto puede suponer una revolución que cambie el modo de vida [Lee *et al.,* 2006].

En su momento la Computación Ubicua tenía una visión inalcanzable, pero hoy en día la evolución de las tecnologías, hacen que ésta sea viable. Las previsiones se van cumpliendo, y la capacidad de cómputo de los procesadores avanza rápidamente, además de la capacidad de almacenamiento o el ancho de banda para las comunicaciones. En resumen, cada poco tiempo aparecen dispositivos más baratos, más pequeños y más potentes, siendo la previsión para los próximos tiempos que siga ocurriendo lo mismo [Costa de *et al.,* 2008].

Los objetos cotidianos en los que se integra la tecnología de computación, tienen una serie de características que permiten y delimitan la creación del entorno ubicuo buscado:

- Comunicación entre dispositivos. Los dispositivos del sistema disponen no sólo de capacidad de computación, sino también de comunicación con otros dispositivos. Esta comunicación se desarrolla tanto con el usuario como con el resto de elementos de su alrededor mediante comunicación WiFi, Bluetooth, GPRS/UMTS, NFC, RFID, etc. [Reynolds *et al.,* 2006].

- Disponibilidad de memoria. Esta memoria puede ser usada para almacenar información y así mejorar la interacción con el resto de dispositivos.
- Sensibilidad al contexto. Los dispositivos tienen que ser capaces de adaptarse a diferentes situaciones, como su situación geográfica, las preferencias de los usuarios y otros dispositivos que se encuentran en el entorno, actuando en base a este contexto.
- Capacidad para reaccionar ante ciertos estímulos o eventos. Estos estímulos se pueden percibir en un entorno a través de sensores o mediante interacción con otros dispositivos [Donghai *et al.,* 2007].

De este modo, la Computación Ubicua, incorpora cuatro nuevos conceptos:

- Uso eficaz del contexto: Se parte de la localización de un usuario, sus necesidades y preferencias dentro de un contexto determinado. En un contexto hay varios dispositivos inteligentes que coinciden en el mismo espacio físico e interactúan colaborativamente para dar soporte a los usuarios que se encuentren en él. La domótica es la aplicación más popular en un domicilio.
- Invisibilidad: Es muy difícil conseguir lo que proponía Weiser para los sistemas ubicuos, la completa desaparición de la tecnología para el usuario. Se tiene que tratar de conseguir la mayor invisibilidad posible para que los usuarios se centren en sus tareas.
- Reconocimiento de voz y de gestos. La comprensión del lenguaje natural y del texto manuscrito. Esta característica ofrece a los sistemas ubicuos una nueva dimensión de posibilidades para interactuar con los usuarios.
- Escalabilidad local. Los usuarios disponen de capacidades asociadas al contexto en el que se encuentran. La interacción entre los sistemas y el usuario depende en gran medida de la distancia a la que se encuentre el usuario.

2.6.3 La computación pervasiva.

La computación pervasiva generalmente se incluye dentro de la Computación Ubicua [Weiser, 1993]. 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 [Tentori y Favela, 2008].

La computación pervasiva integra las nuevas tecnologías en el entorno personal, insertando dispositivos inteligentes en las tareas diarias y haciendo que interactúen de manera natural en todo tipo de situaciones y circunstancias. De esta forma la computación pervasiva hace que los usuarios se centren en las tareas que deben hacer, no en las herramientas que deben usar, para mejorar su calidad de vida [Tentori y Favela, 2008]. Los sistemas pervasivos para ser funcionales necesitan una gran diversidad de servicios, incluidos los multimedia, las comunicaciones, o de automatización [Bahadori *et al.,* 2007]. La automatización de los dispositivos hardware permite su conexión a las redes de comunicación [Moreno *et al.,* 2006] y su ejecución en distintas plataformas. Los recientes avances en microelectrónica y las tecnologías inalámbricas y móviles han favorecido el crecimiento de pequeños dispositivos con alta capacidad de comunicación y procesamiento. Los sistemas pervasivos tienen en cuenta este conjunto de avances. La computación pervasiva tiene como uno de sus objetivos la mejora de la calidad de vida de las personas con enfermedades crónicas y los pacientes en general. Este grupo de población requiere nuevas soluciones que puedan hacer uso de los avances tecnológicos para ofrecer servicios novedosos. Entre otros servicios, la computación pervasiva permite mejorar la calidad, el acceso, la equidad y la continuidad de los cuidados de la salud [Tapia y Corchado, 2009]. En este sentido, los entornos inteligentes pueden mejorar los servicios de cuidado de la salud y pueden tener un alto impacto social.

2.7 Conclusiones

En este capítulo se ha explicado el concepto de computación sensible al contexto (*Context-Aware Computing*) y se han estudiado sus principales aplicaciones y tendencias. Se han presentado las razones por las cuales la computación sensible al contexto está adquiriendo una gran importancia en nuestros días. Dado el gran avance tecnológico se hace necesario proporcionar nuevas soluciones que permitan que la tecnología se adapte de forma automática a los usuarios, facilitando su utilización.

Además en este capítulo se presenta cómo ha evolucionado el concepto de contexto desde que se comenzó a utilizar en el año 1994. La localización es el primer componente del contexto (cronológicamente hablando), a él se añaden parámetros relacionales, ambientales e incluso de actividad y estado de ánimo. Por supuesto, la dificultad de capturar y modelar los parámetros es inherente a su nivel de abstracción.

Por otra parte, este capítulo revisa las tecnologías relacionadas con la gestión del contexto, entendida ésta como el proceso que está por encima de la adquisición de datos física. Entre las tecnologías consideradas se mencionan las de representación del contexto, las de razonamiento, en las que la inteligencia artificial aporta todos sus métodos, y, por último, las arquitecturas generales de los sistemas. De la revisión realizada, se deduce que entre los requisitos que se pueden buscar en una plataforma de provisión de servicios basados en contexto están el soporte a componentes distribuidos y capacidad de descubrimiento de recursos, manejo de fuentes de contexto heterogéneas y soporte de métodos de razonamiento. En muchas ocasiones, los componentes del sistema de provisión de servicios contextuales tienen que estar en diferentes máquinas o entornos. El sistema debe soportar configuraciones distribuidas y mecanismos de descubrimiento y comunicaciones que permitan la incorporación de diferentes sensores y proveedores de contexto. Además en este trabajo se incide en las posibilidades de la coordinación de datos procedentes de fuentes de contexto heterogéneas para obtener mejores estimaciones de contexto. Por tanto, es adecuada la existencia de algoritmos y estructuras de gestión de los mismos que permitan la implementación de estrategias de fusión a diferentes niveles. Del mismo modo el sistema de provisión de servicios puede incluir algún mecanismo de razonamiento o integrarse con motores externos, que implementen diferentes técnicas de razonamiento sobre un modelo de datos definido y dispongan de métodos de resolución de conflictos.

Por otro lado las tecnologías de comunicación inalámbricas tales como WiFi, bluetooth, NFC, etc., facilitan la comunicación ubicua. La aparición de dispositivos ligeros como teléfonos móviles y de arquitecturas cliente/servidor orientadas a servicios facilita la computación ubicua y la resolución de problemas de forma distribuida. La aparición de nuevas tecnologías de sensorización tales como RFID, NFC, ZigBee, etc., permite obtener información precisa y en tiempo real del entorno.

Hasta el momento la computación sensible al contexto se ha centrado especialmente en el estudio del problema y en el desarrollo de sistemas que mejoren la calidad de vida de los usuarios, descuidando en cierto modo la parte inteligente de los sistemas. En este trabajo se propone la utilización de agentes inteligentes como elemento fundamental para el desarrollo de entornos inteligentes sensibles al contexto. Concretamente se propone una arquitectura multiagente inteligente que facilita el aprendizaje y la adaptación a cambios en el entorno. Las características propias de los agentes inteligentes los hacen muy adecuados para ser utilizados en entornos sensibles al contexto. Un sistema multiagente hace más sencilla la resolución de problemas de forma distribuida, de tal forma que los agentes se comunican y coordinan para alcanzar sus objetivos. Este hecho facilita las capacidades de computación y comunicación ubicua necesarias en un entorno sensible al contexto. Cada agente posee capacidades de pro-actividad y razonamiento, lo que posibilita su autonomía y adaptación al entorno en el que se encuentre. Además, los sistemas multiagente presentan estructuras organizativas, de tal forma que imitan a las sociedades humanas y a los diferentes roles que se juegan dentro de ellas. Un sistema sensible al contexto identifica los diferentes roles de un entorno y trata de buscar la adaptación a cada perfil concreto de usuario. Este es uno de los principales motivos por los que se considera la tecnología de agentes apropiada para aplicarla sobre el desarrollo de sistemas sensibles al contexto. La tecnología de agentes se presenta en el siguiente capítulo de este trabajo.

3

Tecnología de Agentes

La tecnología de agentes surge como una evolución de la inteligencia artificial distribuida [Russel y Norvig, 1995]. La evolución del software, y más en concreto del software que incorpora ideas de la inteligencia artificial, tiende a la creación de entidades con comportamientos y conductas similares a las de los humanos. La teoría de agentes se sustenta sobre el concepto de agente [Russel y Norvig, 1995]. Un agente es una entidad autónoma dotada con ciertas capacidades propias de los humanos. Puede verse como una evolución del concepto de objeto software, perfeccionada gracias a la influencia de la inteligencia artificial, que permite incorporar características como la racionalidad, la inteligencia, la autonomía o el aprendizaje. Al igual que los humanos, los agentes deben tener habilidades sociales y ser capaces de realizar trabajos o resolver problemas de forma distribuida. Se habla entonces de un sistema multiagente, en el que los agentes cooperan e interactúan para conseguir los objetivos finales del sistema.

Debido a los rápidos y constantes avances tecnológicos y el desarrollo de sistemas cada vez más complejos, 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. 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 *AgentOriented Software Engineering* (AOSE), entre las que destaca la metodología SysML. SysML facilita y mejora el proceso de ingeniería, consiguiendo así modelos más detallados y cercanos a la implementación de los sistemas multiagente.

En este capítulo se explica cómo surgen y qué son los agentes y los sistemas multiagente, así como la evolución de estos sistemas a lo largo de los últimos años. Se describen las ventajas que aportan los agentes en sistemas como el que se presenta en este trabajo de investigación. En este capítulo también se describen algunos de los sistemas de agentes desarrollados para entornos sensibles al contexto, ya que este trabajo de investigación se centra en la creación de soluciones eficientes para este tipo de entornos. En los últimos puntos del capítulo se presentan los tipos de agentes que se consideran más útiles para la resolución del problema planteado en este trabajo de investigación. El tipo de agente elegido son los agentes deliberativos BDI (*Beliefs* - Creencias, *Desires* - Deseos e *Intentions* – Intenciones). También se explican distintos modelos de razonamiento entre los que destaca el modelo CBR, que es el que se integra en el sistema multiagente descrito en el capítulo 4. Además, el sistema multiagente integra el sistema de planificación CPM que también se explica en este capítulo junto con *Case Based Planning* (CBP) como una especialización del modelo CBR.

La estructura de este capítulo se compone de siete secciones. En la sección 3.1 se realiza una breve introducción a los agentes. En la sección 3.2 se definen los sistemas multiagente y se realiza un estudio de los sistemas de agentes desarrollados, relacionados con la computación sensible al contexto. La sección 3.3 describe las distintas arquitecturas de agentes. La sección 3.4 profundiza en los sistemas de razonamiento y hace especial hincapié en el razonamiento basado en casos. En la sección 3.5 se describen los sistemas de planificación incidiendo en los métodos

utilizados en el desarrollo de este trabajo como son el método de planificación CPM y el modelo de planificación CBP. En la sección 3.6 se enumeran distintas herramientas para el análisis y diseño de la ingeniería del software orientada a agentes y se incide en SysML que es la herramienta que se utiliza en esta investigación. Finalmente, la sección 3.7 presenta las conclusiones correspondientes a este capítulo.

3.1 Introducción a los agentes

El concepto de agente no es fácil de explicar ya que supone realizar una conjunción de varias disciplinas. Este campo abarca una amplia variedad de materias de las ciencias de la computación, como son la Inteligencia Artificial, la Ingeniería del Software, las Bases de Datos, los Sistemas Distribuidos, pero, además, también abarca otras áreas de conocimiento como son la Psicología o la Sociología. Tener en cuenta tantos campos de conocimiento supone muchas ventajas, pero presenta el inconveniente de que en cada uno de ellos se tiende a acercar el problema a su visión concreta. Precisamente esto hace que sea muy difícil poder proporcionar una definición para el concepto de agente y que sean varias las posibilidades que se han presentado. Wooldridge y Jennings, (1995) definen un agente como un sistema computacional basado en hardware o, más habitualmente basado en software que disfruta de propiedades como autonomía, habilidades sociales, reactividad y pro-actividad. Posteriormente Wooldridge (2002) 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. En cambio, Russell y Norvig, (1995) consideran que la noción de un agente aparece como una herramienta para analizar sistemas, no una clasificación que divida el mundo en agentes y no agentes. Para este último autor, un agente es una entidad capaz de percibir su entorno a través de sensores y responder al entorno a través de actuadores, teniendo en cuenta que cada agente percibe acciones y puede aprender de la experiencia para definir su comportamiento.

Dado que no parece posible llegar a un acuerdo respecto a la definición de agente se ha optado por definir una serie de características que debe manifestar un agente. Para ello se parte de las características propuestas por Wooldridge en su primera

definición, como son la autonomía, la sociabilidad, la reactividad y la iniciativa. Por tanto, las capacidades que se exigen a un agente son:

- **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.** Localizarse 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, 2006] [Russell y Norvig, 1995] [Brenner, 2006]. 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 de 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.

Así pues, se puede decir que un agente es una entidad autónoma dotada con ciertas capacidades propias de los humanos. Puede verse como una evolución del concepto de objeto software, perfeccionada gracias a la influencia de la inteligencia artificial, que permite incorporar características como la racionalidad, la inteligencia, la autonomía o el aprendizaje. Al igual que los humanos, los agentes deben tener habilidades sociales y ser capaces de realizar trabajos o resolver problemas de forma distribuida. Se habla entonces de un sistema multiagente, en el que los agentes

cooperan e interactúan para conseguir los objetivos finales del sistema. En el siguiente punto se describe más ampliamente el concepto de sistema multiagente, se relacionan los sistemas multiagente con los entornos sensibles al contexto y se proponen como alternativa válida para el desarrollo de sistemas sensibles al contexto.

3.2 Los sistemas multiagente

Cuando dos o más agentes son capaces de trabajar de forma conjunta con el objetivo de resolver un problema se habla de un sistema multiagente [Mas, 2005]. Es necesario detallar esta afirmación, ya que hablar de que dos o más agentes trabajan de forma conjunta es algo muy general. En realidad, para que una asociación de agentes que trabajan de forma conjunta se considere un sistema multiagente debe cumplirse una serie de condiciones:

- al menos uno de los agentes debe de ser autónomo.
- debe existir al menos una relación entre dos agentes en la que se cumpla que uno de los agentes satisface el objetivo del otro.

Esto quiere decir que al menos uno de los agentes dispone de información incompleta o de capacidades limitadas para resolver el problema. Los sistemas multiagente se caracterizan porque no existe un sistema de control global y porque cada agente se centra en su conducta individual. Por otro lado, los datos se encuentran organizados de forma distribuida, lo que favorece su computación asíncrona. Asimismo, cada agente puede decidir con libertad, de forma dinámica, qué tareas debe efectuar y a quién asigna estas tareas [Wooldridge, 2002].

Como ya se ha mencionado con anterioridad, algunas de las características de los agentes son sus habilidades sociales, situación en un entorno y organización. A través de ellas el agente puede interaccionar con otros agentes utilizando algún tipo de lenguaje de comunicación. El agente debe utilizar su autonomía y pro-actividad y tomar iniciativas, no ser una simple entidad reactiva. El agente define metas y planes que le permitan alcanzar sus objetivos y para ello debe tener en cuenta las características propias de la organización y del entorno en los que se encuentra situado. En un sistema multiagente se hace necesario que los agentes trabajen de forma coordinada para alcanzar sus objetivos. El trabajo coordinado requiere

mecanismos de comunicación y cierto control global. Además, es posible establecer mecanismos de negociación entre los distintos agentes de un sistema multiagente [Ossowski, 2008].

Los sistemas multiagente han evolucionando durante los últimos años, tratando de adaptarse a los cambios que presentan las nuevas tecnologías. El papel que juega el entorno es cada vez más importante en los sistemas multiagente [Weyns *et al.*, 2007]. Hoy en día las personas disponen de dispositivos móviles, con lo que es frecuente que un agente que se ejecuta en un dispositivo móvil cambie frecuentemente de un entorno a otro. Además, cada vez es más frecuente encontrar dispositivos inteligentes que pueden interactuar con los agentes de forma automática. Así pues, los sistemas multiagente necesitan modelar el entorno en el que se encuentran y desarrollar mecanismos de comunicación adecuados con los elementos de dicho entorno. Esto supone la necesidad de ampliar los conceptos utilizados en las metodologías de desarrollo así como desarrollar nuevos mecanismos de comunicación y nuevas herramientas de implantación. Por otro lado, las tecnologías de comunicación han avanzado rápidamente, proporcionando nuevos medios de comunicación, como por ejemplo las redes inalámbricas [Fernández, *et al.* 2007b]. Este tipo de redes introducen un alto grado de movilidad y facilitan el acceso a recursos remotos independientemente de la localización física. Este hecho hace necesario establecer nuevos mecanismos de comunicación y estandarización, que garanticen compatibilidad entre los sistemas multiagente basados en diferentes arquitecturas. Además, la aparición de nuevos dispositivos en los que pueden ser ejecutados los agentes, supone la necesidad de adaptar las arquitecturas de agentes de tal forma que permitan ofrecer las mismas capacidades, utilizando unas cantidades mucho menores de recursos tanto de memoria como de procesamiento.

La aparición del concepto de computación sensible al contexto refleja la importancia que adquiere la tecnología en los entornos donde transcurre la vida de las personas. La computación sensible al contexto busca desarrollar entornos plagados de elementos tecnológicos, capaces de adaptarse a las necesidades de los humanos de forma automática, de forma transparente y ubicua [Dey *et al.*, 2009]. En este tipo de entornos deben vivir los agentes, situándose en distintos dispositivos de diferentes características y conviviendo con otros agentes, con las personas y con el resto de

elementos inteligentes del entorno. La noción de entorno ha sido muy importante en la definición de sistema multiagente, sin embargo, la importancia ha sido mucho menor en las aplicaciones reales. Un claro reflejo de ello es la escasa referencia que se hace al entorno en la mayoría de las metodologías de ingeniería del software orientada a agentes [Gómez-Sanz *et al.,* 2008]. Estas metodologías prácticamente carecen de mecanismos para analizar y modelar el papel del entorno dentro de un sistema multiagente [Gómez-Sanz *et al.,* 2008]. Es importante utilizar metodologías que permitan estudiar y modelar el entorno en el que se desarrolla un sistema multiagente.

3.2.1 La tecnología de agentes y la computación sensible al contexto

Tanto un agente autónomo como un sistema multiagente vienen caracterizados por estar situados en un entono con el que interactúan. Los agentes deben ser capaces de detectar los cambios que se producen en su entorno y de adaptarse a ellos. Por este motivo los agentes deben adquirir información del entorno, así como poder actuar sobre él. Por lo tanto, se hace necesario incorporar tecnologías que permitan a los agentes obtener información y actuar sobre el entorno de forma automática y en tiempo de ejecución. La Inteligencia Ambiental propone nuevas maneras de resolver diferentes problemas y expone una nueva forma de relación entre los humanos y la tecnología. Así, la tecnología se adapta a los individuos y al contexto concreto en el que se encuentren los individuos, como se ha visto en el capítulo anterior. Así pues, según la AmI, las personas se encuentran rodeadas de interfaces inteligentes que se mezclan con las entidades de contexto [Vázquez y López de Ipiña, 2005], creando entornos inteligentes que se ponen al servicio de las personas de manera natural y transparente [Camarinha-Matos y Afsarmanesh, 2008].

Uno de los primeros pasos que se dieron hacia la creación de arquitecturas para aplicaciones *context-aware* fueron los *Stick-e Documents* de Brown (2006). Se trata de notas que muestran mensajes para aplicaciones *context-aware* según unas ciertas condiciones de localización de usuarios. En los inicios de estos sistemas, el objetivo de la mayoría de aplicaciones era presentar información cuando se dieran ciertas condiciones relacionadas con la posición de usuarios o algún otro tipo de contexto dentro de un entorno. Cada *Stick-e Document* consta del propio contenido o mensaje, y del contexto relacionado. En un ejemplo, cuando un usuario lleva consigo un

dispositivo móvil que usa algún medio de localización, se consigue tener una localización constante. Así, el propio usuario utilizando su dispositivo móvil puede dejar un mensaje en la localización donde se encuentra en ese momento o en cualquier otro punto. Cuando esté próximo a ese lugar el *Stick-e note* es mostrado con el mensaje. Es una forma electrónica de dejar mensajes (*Post-its*) a lo largo de determinados lugares. Otra de las primeras arquitecturas *context-aware* creada por Dey *et al.,* (2009) en la Universidad de Berkeley es *Context Toolkit*. El objetivo de *Context Toolkit* es ayudar en la tarea de captación de contexto. El contexto tiene extensiones muy amplias y una naturaleza muy diversa. Por tanto, una de las principales problemáticas a la hora de realizar estos sistemas es la tarea de captación del contexto de una forma homogénea y reutilizable por otras aplicaciones. Por otro lado, la arquitectura *Context Managing Framework* ideada por Koskinen *et al.,* (2006) ya incluye el concepto de razonamiento en entornos sensibles al contexto. Esta arquitectura se centra básicamente en la recogida de información, modelado y razonamiento del contexto. Está organizada de manera jerárquica teniendo un elemento central, que es el gestor de contexto, al que la aplicación accede. El gestor de contexto almacena el contexto capturado para que las aplicaciones puedan acceder a él. De esta forma se crean dos capas: una con el componente central y otra con los componentes secundarios que este componente utiliza. Los componentes secundarios son los servidores de recursos que se conectan a las fuentes de datos de contexto y recogen la información. También es un componente secundario el servicio de reconocimiento de contexto que recoge el contexto de bajo nivel y devuelve contexto de alto nivel tras un proceso de tratamiento de la información. Más tarde surge el sistema *Cortex* que fue creado por Biegel *et al.,* (2004) y básicamente se centra en la creación de lo que llaman *Sentient Object*. Estos objetos reciben información de sensores, la almacenan, procesan y sacan conclusiones sobre ella, para generar una salida o bien actuar sobre actuadores. Este sistema se puede decir que recoge información del contexto, la procesa y actúa sobre el contexto. Por otro lado, surge el sistema *COBRA* creado por Chen *et al.,* (2004a). Este sistema pretende dar soporte a aplicaciones sensibles al contexto en lo que los autores llaman *Intelligent Spaces* o Espacios Inteligentes [Chen *et al.,* 2004b]. Se centra en un componente llamado *Context Broker*, que es el punto central de acceso al contexto del Espacio Inteligente.

El *Context Broker* se encarga de actualizar el modelo de contexto de los diferentes agentes repartidos en el contexto. El sistema *COBRA* está basado en agentes. El *Context Broker* recoge el contexto de los agentes y lo almacena sin redundancia de información gracias a su motor de inferencia. También se encarga de enviar la información del contexto a los agentes que la piden. Además el *Context Broker* se encarga de la seguridad de los agentes utilizando políticas de acceso. También surge el sistema *CAMUS* [Moon *et al.,* 2006] que está basado en agentes que capturan información del contexto. *CAMUS* es un sistema dividido en capas que se encarga tanto de la extracción de información como del razonamiento y entrega del contexto. Los agentes en *CAMUS* recogen el contexto y lo procesan antes de pasarlo a las capas superiores. Procesan las características más importantes del contexto, para obtener información con mayor nivel de detalle. Un poco posterior es el sistema *Citron*. El sistema *Citron* [Yamabe *et al.,* 2005] proporciona a las aplicaciones funcionalidad para adquirir información de contexto y compartirlo entre ellas. La compartición de información tanto dentro de la arquitectura como entre aplicaciones se hace a través del modelo de pizarra, concretamente en una implementación de *Tuple Spaces* [Will*,* 2011] llamada *LinuxTuples*.

Las arquitecturas vistas anteriormente emplean técnicas de captación del contexto, modelado de la información, dispersión de la información y razonamiento de la información para crear sistemas sensibles al contexto. Los sistemas se han ideado para ayudar al proceso de desarrollo de aplicaciones y por tanto pretenden solucionar muchas problemáticas características de estos entornos. Todos los sistemas ofrecen posibilidades para abstraer los detalles físicos de cada tipo de sensores. Estos detalles de los sensores pueden ser físicos (componentes hardware), lógicos (combinación de información) o virtuales (acceso a servicios o aplicaciones externas).

Los sistemas *Cortex* [Biegel *et al.,* 2004] y *CAMUS* [Moon *et al.,* 2006] incluyen pre-proceso de contexto. Este concepto se refiere a un análisis sintáctico previo de la información recibida desde los sensores. Este análisis pretende corregir errores, omitir valores fuera de rango que no se deben tener en cuenta y una adaptación del formato para el tratamiento de la información. También en esta fase se hace una agrupación de valores en categorías o rangos más difusos. Además todo sistema sensible al contexto ha de representar el contexto de una u otra forma. Según se gestione la información de

contexto influirá en su representación. También, el almacenamiento del contexto es un aspecto básico en muchos sistemas, ya que facilita la gestión de datos históricos y la compartición de los datos de contexto con otros sistemas. Por otro lado sistemas como *Context Managing Framework* [Koskinen *et al.,* 2006] o *CAMUS* [Moon *et al.,* 2006] también incluyen motores de razonamiento. Este concepto implica transformación del contexto y utilización de conocimiento. Estos procesos normalmente requieren de motores de razonamiento que utilizan bases de conocimiento para realizar el trabajo.

En todos los sistemas una vez obtenido y almacenado el contexto, ha de enviarse la información o datos procesados a las aplicaciones y componentes que lo soliciten. Otro de los aspectos que tienen en cuenta sistemas como *COBRA* [Chen *et al.,* 2004] es la seguridad. Los datos de contexto pueden incluir información sensible y personal que debe ser protegida. Un sistema debe ser capaz de proporcionar mecanismos para asegurar la privacidad de los datos. Los sistemas multiagente son una alternativa válida para modelar sistemas sensibles al contexto. Los sistemas multiagente ya han sido utilizados con éxito en sistemas de Inteligencia Ambiental [Bajo *et al.,* 2007] [Fraile *et al.,* 2008], entornos pervasivos [Corchado *et al.,* 2008] [Tapia *et al.,* 2009] y en entornos sensibles al contexto [Fraile *et al.,* 2009b].

Los sistemas multiagente son, por tanto, una buena alternativa para la construcción de sistemas sensibles al contexto. Desde el punto de vista de los sistemas distribuidos, un sistema multiagente permite enfocar el problema planteado por un entorno inteligente de forma distribuida, de tal forma que el problema se divide en otros más pequeños que son resueltos con más facilidad. Un sistema multiagente proporciona mecanismos de comunicación y coordinación que permiten a los agentes autónomos intercambiar informaciones para resolver el problema global. Además, desde el punto de vista del análisis y diseño de un entorno inteligente, los sistemas multiagente proporcionan herramientas que permiten obtener los requisitos del entorno y modelar soluciones en términos de organizaciones humanas y de los roles que se juegan dentro de dichas organizaciones. De esta forma, las herramientas de análisis y diseño de sistemas multiagente resultan más adecuadas que ninguna otra herramienta de ingeniería del software para modelar entornos inteligentes, ya que tienen en cuenta la posibilidad de trabajar con elementos inteligentes y autónomos, y modelar tanto sus comportamientos individuales como las interacciones que se establecen entre ellos.

Las herramientas tradicionales de ingeniería del software modelan los elementos de un entorno como actores, pero no tienen en cuenta ni su complejidad interna ni el dinamismo que pueden presentar en su comportamiento.

De la misma forma, un sistema multiagente permite trabajar con sensores o actuadores del entorno de forma eficiente, ya que se pueden definir tantos agentes como tipos de información capturan los sensores. De esta forma un agente puede ocuparse de interactuar con un tipo de sensor o actuador del entorno, actuando como un interfaz a través del cual el sensor o actuador puede comunicarse con el resto del sistema en tiempo de ejecución. Esto permite que un agente proporcione a cada sensor o actuador cierto grado de autonomía, así como capacidad de comunicación con el resto del entorno. Todo agente o sistema multiagente se encuentra situado en un entorno, y debe responder a los eventos que se producen en dicho entorno. Este hecho adquiere especial relevancia si los agentes son utilizados para la construcción de entornos inteligentes, en los que es posible encontrarse con elementos inteligentes y en los que es necesario adaptarse a las necesidades concretas de cada usuario. El gran avance de las tecnologías de sensibilidad de contexto permite dotar a los agentes de mecanismos sensores o actuadores de gran precisión. Además, los avances en las tecnologías de la comunicación permiten que los agentes puedan ser ejecutados en dispositivos móviles de tamaño muy reducido y con limitaciones de procesamiento y de memoria.

3.3 Tipos de arquitecturas de agentes

Las arquitecturas de agentes detallan las tareas que realiza cada agente y la manera de interactuar entre los agentes para lograr la funcionalidad requerida por dicha arquitectura. Si consideramos que un agente es un sistema más o menos complejo, la arquitectura deberá describir la estructura interna del agente. Wooldridge (2002) distingue tres tipos de arquitecturas (y por tanto tres tipos de agentes) que vienen diferenciadas básicamente por 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.

A modo de conclusión, se ha podido observar como la clasificación anterior se fija en los atributos de los agentes como criterio principal. Existen una serie de atributos que pueden servir de referencia, como son la autonomía, la pro-actividad, el aprendizaje, y la cooperación. La clasificación converge especialmente en la necesidad de diferenciar aquellos agentes que son simplemente reactivos de aquellos que incorporan algún mecanismo de razonamiento. Además se hace hincapié en la existencia de agentes híbridos, que flexibilizan las clasificaciones y permiten la integración de más de un criterio de clasificación. Por ello en el marco de este trabajo de investigación se propone una arquitectura multiagente que contemple la inclusión de agentes en una arquitectura deliberativa con autonomía, inteligencia y capacidad para cooperar con otros agentes para alcanzar sus objetivos individuales y colectivos.

3.3.1 Arquitectura deliberativa BDI

Tal y como se ha podido apreciar en el apartado anterior de este capítulo, existen muchas arquitecturas que permiten construir agentes deliberativos y muchas de ellas se basan en el modelo BDI [Bratman *et al.*, 1988]. El modelo BDI parte de la base filosófica de la teoría de las Creencias-Deseos-Intenciones presentada por Bratman (1988) como modelo de razonamiento práctico humano. En el modelo BDI la

estructura interna de un agente y su capacidad de elección se basan en aptitudes mentales. Esto tiene la ventaja de utilizar un modelo natural y de alto nivel de abstracción. El modelo BDI utiliza *Beliefs* como aptitudes informacionales, *Desires* como aptitudes motivacionales e *Intentions* como aptitudes deliberativas de los agentes [Bajo *et al.,* 2006b].

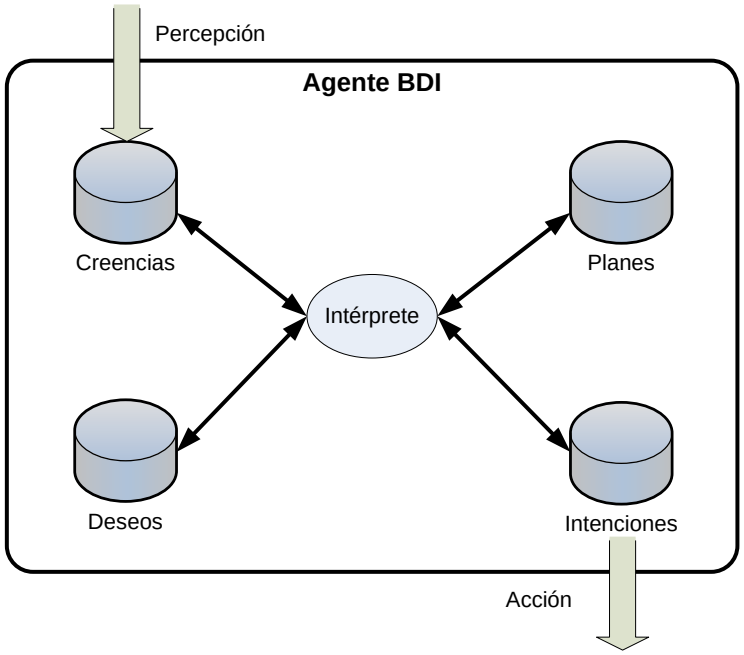


Figura 3.1 Arquitectura básica de un agente BDI.

Esta sección se centra en los agentes basados en el modelo deliberativo, y más concretamente en aquellos que se basan en la arquitectura BDI que se muestra en la Figura 3.1, donde se ve cómo un conjunto de creencias, planes, deseos e intenciones, son procesadas por un intérprete. El intérprete desempeña funciones como la clasificación de información, el razonamiento o la deliberación. El intérprete tras desempeñar sus funciones modifica los conjuntos de creencias, planes, deseos e intenciones y así el agente BDI genera una serie de acciones.

Posiblemente, el modelo BDI es el más difundido y el más estudiado dentro de los modelos de razonamiento de agentes. El motivo es que el modelo BDI combina elementos interesantes:

- un modelo de razonamiento humano fácil de comprender,
- un número considerable de implementaciones [Georgeff y Lansky, 1987], como por ejemplo sistemas de control de procesos,
- una semántica abstracta y elegante, que ha sido aceptada por la comunidad científica [Georgeff y Rao, 1998].

El modelo de agente deliberativo BDI es el más utilizado en la actualidad, ya que es el que más se asemeja al modelo de razonamiento humano. Es por ello que será el modelo de agente en el que se centra este trabajo de investigación.

En el campo de la Inteligencia Artificial, las Creencias son aptitudes informacionales capaces de representar el conocimiento que se posee del entorno. Las creencias pueden ser implementadas utilizando distintos tipos de estructuras de datos y pueden ser almacenadas de forma temporal (como el valor de una variable) o permanente (en bases de datos). Los agentes racionales tratan de resolver problemas reales. Para los agentes racionales el entorno es dinámico y se encuentra sujeto a constantes cambios. Por tanto las creencias que posee el agente permiten recordar eventos pasados.

Por otro lado los Deseos de un agente deben ser implementados, en general, como una estructura de mayor complejidad que las creencias. Un deseo o meta representa las aptitudes motivacionales del agente. Es necesario identificar los distintos tipos de motivación que puede tener un agente, es decir identificar qué es lo que quiere conseguir el agente y las condiciones bajo las que lo desea conseguir. Finalmente un agente debe ser capaz de determinar si un deseo se ha cumplido total o parcialmente. Algunas arquitecturas como Jadex [Pokahr *et al.* 2003] permiten distinguir entre tipos de deseos o metas, identificando cinco tipos de metas:

- *"perform goal"* que especifican algunas actividades que necesitan ser completadas.
- *"achieve goal"* que representa un estado objetivo que se debe alcanzar.

- *"query goal"* que se utiliza para solicitar información relativa a un determinado tema.
- *"maintain goal"* que debe observar algún estado que se desea mantener y restablecer dicho estado si las condiciones de mantenimiento son violadas.
- *"meta-level goal"* que se utilizan en el proceso de selección de planes para razonar sobre cuál es el evento o plan más adecuado.

Además en Jadex se considera que para cada meta es necesario indicar:

- si contiene o no parámetros,
- si es única (si no se permiten dos o más instancias iguales de una meta en un momento dado),
- si contiene condiciones de creación que permiten describir la situación en la que una meta es creada automáticamente, o condiciones de contexto para indicar aquellas situaciones en las que una meta tiene que ser suspendida automáticamente, o condiciones de abandono para abandonar automáticamente la consecución de una meta,
- si se utiliza un proceso de deliberación.

Se trata de una característica de la arquitectura Jadex que permite definir cómo interactúan las metas entre sí y permite decidir de forma autónoma al agente, qué metas va a perseguir de entre todas las activas.

Finalmente las Intenciones, concepto fundamental en el modelo propuesto por Bratman, representan las aptitudes deliberativas de un agente. Una Intención es un elemento que forma parte de planes parciales, cuyos roles principales en la vida humana son los de coordinación y organización tanto temporal como social. Bratman (1988) llama a esto la teoría de la planificación de intenciones. De esta forma las intenciones, en contraste con los deseos ordinarios, se encuentran sujetas a presiones racionales distintivas en cuanto a consistencia y coherencia. En particular, se pretende conseguir que las intenciones junto con las creencias se ajusten a un modelo del futuro para alcanzar un estado de consciencia. Además la representación de las intenciones debe ser coherente y consistente. Este modelo intencional ha sido adoptado y adaptado por diferentes autores al campo de la Inteligencia Artificial, y más concretamente a los agentes y sistemas multiagente [Bratman, 1987]. Las intenciones

se ocupan de controlar las acciones (planes) que ejecuta el agente. Se trata de planes parciales de acción que el agente se compromete a ejecutar para alcanzar sus deseos. Las intenciones permiten un análisis racional de la situación en un momento determinado, de las posibles situaciones futuras y de las informaciones sobre cambios en el entorno. Esto es, las intenciones permiten dotar al agente de un comportamiento deliberativo que permite al agente razonar sobre las decisiones a tomar. Cada cierto tiempo un agente debe replantearse sus intenciones.

Los componentes básicos de los sistemas creados para trabajar en un entorno dinámico incluyen de alguna manera las creencias, deseos, intenciones y planes, y a estos se le denomina agentes BDI. Los agentes BDI disponen de los componentes necesarios para enfrentarse con los problemas del mundo real [Georgeff y Rao, 1998]. Así pues un sistema puede describirse como un agente con aptitudes mentales de creencias, deseos e intenciones. Un agente ha de ser capaz de actuar de forma autónoma en un entorno dinámico. Para ello necesita almacenar los conocimientos que posee sobre su entorno, y que esta información pueda ser modificada cuando el agente detecte cambios en el entorno. El sistema utiliza creencias para realizar la labor de almacenamiento y debe tener información acerca de los objetivos que se deben alcanzar y sobre la importancia o prioridad de cada uno de estos objetivos. Estos objetivos constituyen la parte de deseos del sistema. Hay que tener en cuenta que normalmente no existe un único objetivo, sino varios objetivos que pueden incluso no ser compatibles entre sí. El sistema se completa incluyendo un tercer componente, como son las intenciones. Una intención representa el camino elegido por el sistema para alcanzar los objetivos, es decir, el orden de las acciones que ejecutará para conseguirlos.

Rao y Georgeff (1995) modelan el problema utilizando un árbol de decisión en el que cada nodo del árbol es un mundo posible. El árbol tiene ramas que representan varios posibles futuros y un único pasado. Una situación es un punto de tiempo en un mundo determinado. Las ramas del árbol representan las opciones de pasar de un mundo a otro. Se definen dos operadores modales, el operador **opcional** y el operador **inevitable** que operan sobre "*path-formulas"*. Una "*path-formula"* es opcional en una situación si es verdadera en al menos una ruta que parta de dicha situación. Una "*path-formula"* es inevitable en una situación si es verdadera en todas las rutas que

parten de dicha situación. Además de los operadores modales se definen operadores temporales (*eventually, always, until y next*). Por otro lado las ejecuciones de eventos pueden tener éxito o no. El modelo se completa definiendo las relaciones de accesibilidad:

- La probabilidad de pasar de un mundo a otro se indica mediante las creencias (mundos posibles).
- La recompensa de pasar de un mundo a otro se representa mediante los deseos (mundos que se desea alcanzar).
- La elección de las mejores acciones para un plan mediante un proceso deliberativo (mundos que el agente decide alcanzar).

Finalmente se definen axiomas de compatibilidad que deben existir entre los elementos del modelo BDI, es decir, entre las creencias, los deseos y las intenciones del agente [Rao y Georgeff, 1991]. Algunas de las más importantes se enumeran a continuación:

- Compatibilidad entre creencias y objetivos. Si el agente adopta el deseo de alcanzar un objetivo, debe creer que en alguno de los mundos accesibles por la relación de creencia, dicho objetivo es cierto.
- Compatibilidad entre objetivos e intenciones. Previamente a que el agente adopte una fórmula como intención debe haberla adoptado como deseo.
- Las intenciones conducen a acciones. Si una de las intenciones es una acción primitiva o simple, el agente la ejecuta. Es decir, no se pospone la ejecución de una acción simple.
- Relación entre creencias e intenciones. El agente conoce (cree en) sus propias intenciones.
- Relación entre creencias y objetivos. El agente conoce sus objetivos o deseos.
- No hay retrasos infinitos. Cuando un agente adopta una intención, sigue con ella hasta algún momento del futuro; es decir, no puede haber una post-posición infinita en el proceso de alcance de un determinado objetivo.

Un aspecto importante de la arquitectura BDI es el concepto de compromiso con las decisiones previas. El compromiso genera una cierta estabilidad al proceso de razonamiento. De este modo, el esfuerzo informático se reduce y, por lo tanto, la

ejecución global es más eficiente. Un compromiso tiene dos partes: una es la condición que el agente está comprometido a mantener, llamada condición comprometida; y la segunda es la condición bajo la cual el agente renuncia al compromiso, llamada condición de terminación [Rao y Georgeff, 1995].

Los agentes deliberativos BDI tienen muchas características compatibles con el desarrollo de sistemas sensibles al contexto. Sin embargo, con estas características no es suficiente, es necesario que los agentes incorporen nuevas características que les permitan solucionar problemas de forma similar al comportamiento humano. En el siguiente punto de esta memoria se describen los sistemas de razonamiento que usa el sistema multiagente descrito en el capítulo 4 de este trabajo.

3.4 Sistemas de razonamiento

El término razonamiento se define de diferente manera según el contexto. Normalmente se refiere a un conjunto de actividades mentales que se encargan de conectar unas ideas con otras de acuerdo a ciertas reglas o también puede referirse al estudio de ese proceso [Zhang *et al.,* 2006]. En sentido amplio, se entiende por razonamiento la facultad humana que permite resolver problemas [Zhang *et al.,* 2006]. Se llama también razonamiento al resultado de la actividad mental de razonar, es decir, un conjunto de proposiciones enlazadas entre sí que dan apoyo o justifican una idea [Bai et al., 2008]. Una vez vistas algunas de las definiciones de razonamiento, el proceso de razonamiento en los sistemas de provisión de servicios se ha abordado desde diferentes perspectivas:

- razonamiento lógico (basado en reglas) [Bry *et al.,* 2005],
- razonamiento probabilístico (redes bayesianas) [Koskinen *et al.,* 2006],
- razonamiento evidencial (Dempster-Shafer) [Wu, 2007],
- razonamiento borroso y soportado por técnicas de inteligencia artificial (IA) [Cheung *et al.,* 2005], como redes neuronales artificiales [Lim *et al.,* 2007]
- razonamiento basado en casos [Corchado *et al.,* 2008b] [Ma *et al.,* 2005].

A continuación se comentan brevemente, proporcionando algún ejemplo ilustrativo.

El **Razonamiento basado en reglas** consiste en establecer predicados que enuncien sucesos o resultados si se dan unas ciertas condiciones. Es la modalidad más sencilla y también una de las más utilizadas en el desarrollo de aplicaciones contextuales. Por ejemplo, Bry *et al.,* (2005) lo utilizan para inferir contextos complejos, sirviéndose para ello el *Generic Reasoner* del *Jena Semantic Web Framework*.

Las **redes bayesianas** son grafos acíclicos, donde los nodos son variables aleatorias que representan eventos y los arcos entre nodos, relaciones causales [Koskinen *et al.,* 2006]. Para estimar la distribución de probabilidad de una red bayesiana, es necesario estimar las probabilidades de todos aquellos nodos que no tienen predecesores (nodos raíz) y las probabilidades condicionadas de todos los nodos no raíz, que vienen dadas por las de sus predecesores. Una particularización de las redes bayesianas son las que representan *clasificadores bayesianos naïve* [Koskinen *et al.,* 2006], aquellos en los que los atributos de una clase son independientes entre sí, por lo que no existen arcos entre los atributos y solamente nacen del nodo raíz. Esto configura una estructura en árbol que facilita la computación. Koskinen *et al.,* (2006) utilizan *redes bayesianas naïve* para clasificar los contextos de un usuario, de un dispositivo móvil en sus actividades diarias. El sistema de clasificación se basa en características sonoras, medidas en un escenario de hogar, donde el prototipo obtiene buenos resultados de acierto de situaciones. No obstante, los autores señalan la dificultad de generalización en entornos no controlados.

La **lógica borrosa** quiere manejar el razonamiento de forma similar a como lo hace el cerebro humano, gestionando el procesado de datos inexactos, imprecisos o subjetivos [Cheung *et al.,* 2005]. Se conoce también como lógica *fuzzy*, difusa o multi-evaluada, frente a la lógica basada en predicados o binaria, en la que las variables adoptan uno de los dos valores posibles: Verdadero (1) o Falso (0). Deriva de la teoría de conjuntos borrosos, en la cual los miembros del conjunto pertenecen (1) o no (0) al mismo. Con reglas difusas, es posible procesar las relaciones entre las variables *fuzzy*. El proceso de razonamiento basado en lógica borrosa tradicional tiene tres etapas: *fuzzificación*, razonamiento por el motor de inferencias y *defuzzificación*. Cheung *et al.,* (2005) describen un *middleware* para provisión de servicios

contextuales que incorpora lógica borrosa. Entre los módulos que componen el sistema se encuentra un *ContextSpace*, que contiene objetos *ContextDetectors* y un conjunto de *Fuzzy Context Composers*; éstos últimos componen la información de bajo nivel en representaciones de más alto nivel. Por ejemplo, un compositor podría monitorizar todos los detectores de red y determinar la calidad de la conexión.

En el **Razonamiento evidencial** (*Dempster-Shafer*) la teoría de la Evidencia no necesita de un modelo de probabilidad completo, sino que trata de obtener información de la utilización de conjuntos de hipótesis (en lugar de hipótesis por separado) [Wu, 2007]. Utiliza funciones de creencia (*belief functions*) y razonamiento plausible/probable, que se utiliza para combinar trozos de información (o evidencias) para calcular la probabilidad de un evento. Permite que se considere la confianza que se tiene en las probabilidades asignadas a varias salidas. Facilita la reasignación de probabilidad de creencia en las hipótesis cuando se cambian evidencias, y modela la disminución del número de hipótesis de trabajo a partir de la acumulación de evidencias. Wu, (2007) propone una metodología sistemática de fusión de sensores implementada sobre el *Context Toolkit* y basada en el uso de este tipo de razonamiento. Zhang, (2006) presentan un mecanismo de inferencia basado en esta teoría aplicado al problema del razonamiento contextual. Los autores opinan que este método de estimación es más apropiado que las redes bayesianas para tratar el problema de la incertidumbre en la computación basada en contexto, debido a su manejo de las restricciones.

Las **redes neuronales artificiales** son una aproximación al aprendizaje que se basa en recrear las propiedades en los sistemas neuronales biológicos a través de modelos matemáticos, con el objetivo de obtener razonamientos similares a los del cerebro humano por su capacidad de abstracción y su robustez [Lim *et al.,* 2007]. Lim *et al.,* (2007) utilizan un algoritmo de red neuronal para razonar en el entorno de un *smart home*. Los datos recogidos por los sensores se almacenan en forma de casos, que después son utilizados para entrenar con un algoritmo de red neuronal; el resultado se aplica a un motor de razonamiento CBR. Sánchez *et al.,* (2007) comparan su funcionamiento con el de un modelo oculto de Markov.

El **Razonamiento Basado en Casos** (CBR) permite resolver problemas de razonamiento con información acerca de los procesos clave y de sus interrelaciones,

basándose en el conocimiento de casos anteriores por analogía [Bajo *et al.,* 2006b] [Shokouhi, *et al.,* 2009]. Por ello, necesita un proceso de entrenamiento, que después le permite generalizar su comportamiento. Sus detractores destacan que este sistema acepta la evidencia anecdótica, sin un modelado estadístico del problema, como base para la generalización. En el área del razonamiento contextual, hay algunos trabajos que utilizan CBR. Por ejemplo, Ma *et al.,* (2005) lo aplican al razonamiento sobre el contexto en un hogar inteligente, donde no se conoce a priori las interdependencias entre los servicios y los electrodomésticos y aparatos en general. La aproximación permite recuperar y adaptar las soluciones a partir de escenarios previos. Bai (2008), por su parte, propone un sistema de razonamiento para contexto representado en ontologías que utiliza CBR. Se llama *OntoCBR* y utiliza el contexto descrito en ontologías como las características de los casos para implementar el sistema CBR en la aplicación basada en contexto. A continuación se describe más en detalle el razonamiento CBR ya que es el razonamiento utilizado en la arquitectura descrita en el capítulo 4 de este trabajo de investigación.

3.4.1 Razonamiento basado en casos

El razonamiento basado en casos (CBR) es un tipo de razonamiento, utilizado en el pensamiento humano, en el que se recurre a experiencias pasadas para resolver nuevos problemas [Bajo *et al.,* 2006b]. Así pues, si en un momento pasado se resolvió un problema con una solución (o grupo de soluciones) y, una vez aplicada esa solución se obtuvo un resultado, entonces parece lógico que si se presenta un nuevo problema similar al que se resolvió en el pasado, se utilice la experiencia adquirida en el pasado para dar solución al problema nuevo. De esta forma, sabemos que disponemos en la memoria de la experiencia correspondiente a la resolución de un problema similar al actual. Si el resultado obtenido tras aplicar la solución en el pasado fue bueno, entonces parece lógico aplicar ahora una solución similar para resolver el problema actual. Por el contrario, si el resultado que se obtuvo en el pasado no fue bueno, entonces la opción lógica debe ser la de modificar la solución que se aplicó en el pasado buscando obtener un resultado mejor. Se trata pues, de razonar a partir de recuerdos. Una vez finalizado el proceso, la persona ha realizado un aprendizaje, ya que ha aprendido de la nueva experiencia vivida. Hay que tener en

cuenta dos aspectos del modelo de razonamiento basado en la experiencia. En primer lugar, el modelo se basa en la idea de que problemas similares tienen soluciones similares. Sin embargo, carecer de problemas similares no supone que el sistema no sea capaz de proponer buenos resultados, sino que la reutilización de memorias pasadas se convierte entonces en un proceso creativo. Sea cual sea el resultado de este proceso creativo, el individuo aprende de la nueva experiencia. En segundo lugar, en el proceso de razonamiento explicado se está emitiendo un juicio de valor que permite saber si la solución aplicada para resolver un determinado problema fue buena o no fue buena. Si este juicio es emitido por un experto en la materia del problema resuelto, mayores serán las posibilidades de incrementar la capacidad de aprendizaje.

Cuando las soluciones que se proponen son planes, entonces se habla de una especialización del CBR denominada CBP. Este tipo de planificación se describe más en detalle en el siguiente punto de este capítulo. Así pues, el concepto de caso es fundamental en los sistemas CBR/CBP, tanto que la estructura de un sistema CBR/CBP se diseña sobre el concepto de caso. Un caso es parte de conocimiento que representa una experiencia. De esta forma, un caso se puede ver como una lección aprendida cuando se ha resuelto un determinado problema.

Por otro lado, el modelo de actuación de un sistema CBR se conoce como ciclo de vida de un sistema CBR. En el ciclo de vida de un sistema CBR se especifican los pasos ordenados y relacionados por tiempo mediante los cuales se extrae y aprende información para resolver un problema específico. El ciclo de vida de un sistema CBR está formado por cuatro procesos secuenciales [Shokouhi, *et al.,* 2009], como muestra la Figura 3.2. Estos cuatro procesos son conocidos como las cuatro "erres": *Retrieve* (Recuperación), *Reuse* (Adaptación o Reutilización), *Revise* (Revisión) y *Retain* (Retención o aprendizaje) [Bajo *et al.*, 2006b]. Cuando se presenta un nuevo problema a un sistema CBR, se recurre a la experiencia pasada para obtener la solución más adecuada. Así pues, lo primero que hace el sistema CBR es ejecutar la fase de recuperación. Se trata de una fase en la que se buscan en la memoria aquellos casos con una descripción de problema más similar al problema actual. Seguidamente se ejecuta la fase de adaptación, en la que el sistema trabaja con las soluciones correspondientes a los casos más similares recuperados en la fase anterior. El resultado de la fase de adaptación es una solución al problema actual. La solución

propuesta se revisa o evalúa, comprobando su validez. Finalmente, en la fase de aprendizaje, el sistema almacena la nueva experiencia y aprende de ella. Tal y como el lector puede apreciar, las posibilidades para ejecutar cada una de las fases del ciclo CBR pueden ser múltiples. Por ejemplo, en la recuperación de casos similares se pueden aplicar diferentes técnicas y algoritmos de similitud. Opcionalmente se puede utilizar una fase de revisión del conocimiento experto.

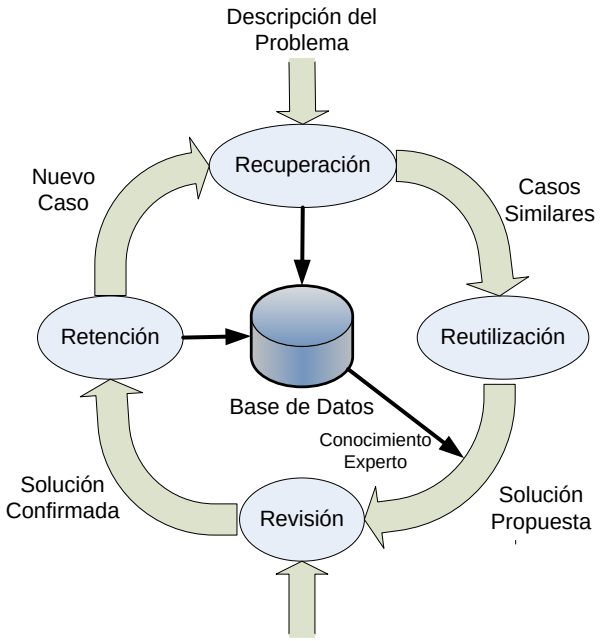


Figura 3.2 Ciclo de vida de un mecanismo CBR.

La Figura 3.2 muestra el ciclo de vida de un sistema CBR. En la Figura 3.2 se puede comprobar cómo el ciclo se inicia con la llegada de un nuevo problema y la obtención del caso correspondiente. El sistema identifica las características más relevantes del problema planteado y las plasma en la estructura correspondiente a la descripción de problema de un caso. Una vez que se ha identificado el caso, el sistema pasa a resolverlo y, para ello, ejecuta el ciclo CBR etapa por etapa:

- *Retrieve* (Recuperación): Es la primera etapa que realiza un sistema CBR. En ella se realiza la recuperación de casos, esto es, el acceso a los casos almacenados que cuentan con una descripción de problema más similar a la del

problema actual. Se llevan a cabo dos funciones distintas: acceso a los casos almacenados y establecimiento de la similitud entre casos (más concretamente entre descripciones de problema).

- *Reuse* (Reutilización o Adaptación): Una vez finalizada la recuperación, el sistema CBR pasa a ejecutar la etapa de reutilización. Esta etapa recibe como entradas los casos más similares recuperados durante la fase anterior. La reutilización o adaptación consiste en trabajar con las soluciones correspondientes a los casos más similares recuperados en la etapa anterior para poder obtener una solución para el problema actual. Trabajar con las soluciones significa modificarlas y combinarlas, o simplemente decidir cuál de ellas es la más óptima y reutilizarla.

  Al igual que ocurre en la etapa anterior, existen muchas posibilidades para realizar la adaptación. Lo ideal es encontrar un caso con una descripción de problema idéntica al actual. Pero, dado que en el mundo real es muy complicado que se presenten dos situaciones idénticas, se hace necesario crear una nueva solución basándose en las soluciones similares de las que se dispone. Además de los datos proporcionados por los casos similares recuperados se hace necesario utilizar algún tipo de conocimiento. El conocimiento puede venir dado a través de fórmulas o reglas
- *Revise* (Revisión): Al igual que ocurre con la fase de recuperación, el proceso de evaluación de la solución finalmente aplicada para resolver el problema actual puede ser problemático y necesitar un conocimiento considerable. Se comprueba si la solución propuesta en la etapa anterior es apropiada para el caso actual. Para ello se utiliza un sistema experto de conocimiento, o bien una persona experta. El resultado de esta etapa es un nuevo caso, para el que se haya obtenido una solución satisfactoria o una solución incorrecta y deba ser reparado. En ocasiones, en esta etapa se puede realizar una reparación de los fallos o errores detectados.
- *Retain* (Retención y Aprendizaje): En esta última etapa se aprende a partir de la nueva experiencia adquirida. Para ello se almacena el caso actual y la solución aplicada para resolverlo. Además se tiene en cuenta el resultado obtenido en la

etapa de revisión para asignar una eficiencia al caso y que de esta forma pueda ser indexado en la memoria de casos. Puede ser necesario reorganizar la memoria de casos. Si hay similitud con otros casos se puede aplicar generalización. En ocasiones durante esta etapa, o bien como una etapa adicional previa a la retención y aprendizaje se incluye una revisión de la base de conocimiento. Si están utilizando reglas u otro tipo de conocimiento experto, éste es revisado y actualizado en función de los resultados obtenidos para la experiencia que se acaba de vivir.

La base de casos o memoria de casos es una de las partes más importantes de la estructura de un sistema CBR. Es la encargada de mantener la representación y organización de los casos: de ella se extraen las soluciones anteriores y en ella se almacena lo aprendido. Debe tener en cuenta la estructura de los casos, y tratar de facilitar en la mayor medida posible cada una de las operaciones que se realizan en el ciclo CBR. Durante el desarrollo de esta sección se ha mencionado que adicionalmente a la memoria de casos puede hacerse necesaria la utilización de una base de conocimiento. La base de conocimiento contiene informaciones tales como teorías, principios, reglas, etc. que permiten tomar decisiones para la solución de problemas y aprender de las experiencias. Tal y como se ha indicado estos conocimientos suelen ser necesarios durante las etapas de adaptación y de revisión. Se trata de conocimientos que permiten realizar funciones tales como la generalización, la toma de decisiones, etc. Se trata de funciones complicadas en las que el mantenimiento también juega un papel muy importante. El conocimiento del que dispone el sistema puede ser un sistema experto. El sistema de conocimiento experto debe ser capaz de aprender de las experiencias adquiridas por el sistema CBR. Probablemente se trata de la parte más complicada de un sistema CBR. A los agentes deliberativos BDI que hacen uso de sistemas CBR se les llama agentes CBR-BDI [Bajo *et al.,* 2006] [Shokouhi, *et al.,* 2009]. Estos agentes se utilizan en el marco de esta memoria y debido a esto se describen a continuación.

3.4.2 Agentes CBR-BDI

Los agentes deliberativos BDI cuentan con un conjunto de características apropiadas para el desarrollo de sistemas *Context-Aware*. 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 para dotarlos de mecanismos que permitan solucionar problemas de forma dinámica y aprender de forma autónoma. Algunos de estos mecanismos son los sistemas CBR [Shokouhi *et al.,* 2009].

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, como se ha explicado en el punto anterior. 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 [Bajo *et al.,* 2006]. 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 [Fraile, *et al.,* 2008]. 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.

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 utilizan en cada una de ellas. Además hay que definir una estructura para el concepto de caso utilizando herramientas para la construcción de agentes BDI, tales como *Jadex* [Pokahr *et al*., 2003] [Bordini *et al.,* 2005]. El objetivo es crear una biblioteca de casos, que implemente características de los agentes CBR-BDI, y que permita a los desarrolladores reutilizar los agentes para resolver problemas concretos e incorporar nuevos algoritmos a las diferentes etapas. Jadex es un software para la creación de

agentes que buscan objetivos y están basados en el modelo BDI. La arquitectura Jadex se basa en una capa de agente racional que se encuentra en la parte superior de una infraestructura de *middleware*, como agente *JADE* [Bordini *et al.,* 2005], y apoya el desarrollo del agente con tecnologías tradicionales, como Java y XML. El modelo *Jadex* mejora las limitaciones tradicionales de los sistemas BDI con la introducción de nuevos conceptos tales como objetivos explícitos y mecanismos de deliberación, por lo que los resultados de los métodos de análisis y diseño son más fácilmente transferibles a la fase de implementación. Jadex ha sido utilizado para construir aplicaciones en diferentes dominios como la simulación, la planificación y la computación móvil [Bordini *et al.,* 2005]. La arquitectura Jadex desarrollada por el grupo de Sistemas de Información y Sistemas Distribuidos de la Universidad de Hamburgo es libre y puede ser descargada de http://jadex.sourceforge.net. Además del sistema *Jadex* y herramientas de desarrollo adicionales también se puede descargar un tutorial de introducción, una guía de usuario y varias aplicaciones de ejemplos con el código fuente.

De la misma forma que a lo largo del apartado 3.4 se han descrito los sistemas de razonamiento, en el siguiente punto de este capítulo, se detallan los sistemas de planificación. El sistema multiagente que se define en el capítulo 4 de esta investigación incorpora algoritmos de planificación, con lo que resulta necesario revisar el estado del arte de los sistemas de planificación.

3.5 Sistemas de planificación

La planificación es una forma de razonamiento en la que las acciones ejecutadas se encuentran dirigidas por el deseo de alcanzar alguna meta. La decisión de elegir una acción u otra suele realizarse tratando de predecir las consecuencias que tendrá la ejecución de cada acción [Lagoudakis *et al.,* 2005]. Así pues, la planificación es un área que busca la elaboración de programas que permitan obtener planes de acción de forma automática para alcanzar una determinada meta. Esto es, sistemas de inteligencia artificial que sean capaces de razonar sobre sus acciones. La planificación se suele plantear como un problema de búsqueda, en el que se trata de encontrar una

secuencia de acciones que permita pasar de un estado inicial a un estado final objetivo. A la secuencia de acciones se denomina plan [Lagoudakis *et al.,* 2005].

La planificación ha sido estudiada desde muchos puntos de vista o perspectivas. Dado que el proceso de planificación se puede abordar de muy diferentes formas, en general se intentan establecer clasificaciones dependiendo del tipo de representación del espacio de búsqueda y de la técnica o algoritmo empleado para encontrar el plan. En cuanto a los algoritmos de búsqueda, los hay de muy distintos tipos pero todos pretenden conseguir:

- ser eficaces, que el plan propuesto resuelva el problema con éxito,
- ser completos, que se valoren todos los planes posibles,
- ser óptimos de acuerdo a los criterios de optimización elegidos [Russell y Norvig, 1995] dentro de una complejidad de espacio y tiempo.

En esta sección se presenta una pequeña revisión del principal modelo de planificación utilizado en este trabajo de investigación.

3.5.1 Método CPM de planificación

El método de planificación CPM tiene como principal objetivo determinar la duración de un proyecto o plan de tareas, entendiendo el plan de tareas como una secuencia de actividades relacionadas entre sí, donde cada una de las actividades tiene una duración estimada. El método CPM se aplica ampliamente en procesos de planificación. CPM permite controlar la ejecución de actividades y la determinación de actividades críticas de obligado cumplimiento para no retrasar la planificación global [Kwak *et al.,* 2008]. CPM se aplica en diversos campos, a procesos de manufacturación [Gupta *et al.,* 1991], en entornos industriales [Dundas y Krentler, 1982], la distribución de tareas en computación GRID [Lina y Lin, 2006] o en el tratamiento de información de modo distribuido en hospitales [Kwak *et al.,* 2008]. En CPM las actividades y sus tiempos de duración son conocidos, es decir, no existe incertidumbre. Este hecho simplifica el método y hace que sea fácil de utilizar. En el método CPM la duración de un plan de tareas es igual a la ruta crítica. Esta ruta crítica es la trayectoria más grande del conjunto de actividades del plan. El método CPM es aplicable y útil en cualquier situación en la que se tenga que llevar a cabo una serie de

actividades relacionadas entre sí para alcanzar un objetivo determinado. El verdadero valor de la técnica en el método CPM aumenta cuando se aplica de forma dinámica. A medida que se presentan hechos o circunstancias imprevistas, el método de la ruta crítica proporciona el medio ideal para identificar y analizar la necesidad de replantear o reprogramar el proyecto, reduciendo al mínimo el resultado adverso de dichos imprevistos. Del mismo modo, cuando se presenta una oportunidad para mejorar la programación del proyecto, la técnica permite determinar fácilmente qué actividades deben ser aceleradas para que se logre dicha mejoría. La integración del método CPM en un sistema hace que este tenga que incorporar información sobre:

- qué tareas deben realizarse primero,
- cuándo utilizar los recursos del sistema,
- cómo programar el uso de dispositivos,
- cómo programar el avance de actividades,
- cuántas actividades hay y cuáles se realizan en determinados momentos,
- cuál es la situación del plan de tareas que se encuentra en marcha en relación con la fecha de finalización,
- cuáles son las actividades críticas, es decir, las que si se retrasan, se retrasa la duración del plan de tareas.
- cuáles son las actividades no críticas y sus holguras, es decir, los periodos de tiempo que estas tareas se pueden retrasar.

El método CPM consta básicamente de dos ciclos, uno de planificación y programación y otro de ejecución y control. En el primero de ellos se define el plan de tareas con todas sus actividades o partes principales. Luego se establecen las relaciones entre las actividades y se decide cuál debe comenzar antes y cuál debe seguir después. Posteriormente, en este primer ciclo se definen los costes y tiempos estimados para cada actividad y se evalúa la factibilidad del plan de tareas. Finalmente se identifica la trayectoria más larga del proyecto, siendo ésta la que determina la duración del plan de tareas, que es lo que se denomina ruta crítica. En el segundo ciclo se aprueba el plan de tareas, se definen las instrucciones, se controlan los avances y finalmente se implementan las medidas correctivas.

Para el desarrollo de planes de tareas óptimos el planificador debe contar con una serie de información y seguir las normas del método CPM. Un plan de tareas está formado por un conjunto de actividades que se deben realizar en un orden determinado para conseguir un objetivo. A la representación gráfica de un plan de tareas con el método CPM se le denomina red. La red es un grafo en el que los nodos son los sucesos y las aristas son las actividades. Los sucesos son instantes de la actividad que sirven como puntos de control y describen el momento de comienzo o terminación de la actividad. Cada actividad se representa por una flecha cuya longitud no representa relación alguna con su duración. Para construir una red hay que tener en cuenta que cada actividad se representa por una y sólo una flecha y se debe identificar por medio de dos nodos, teniendo como mucho dos actividades distintas un solo nodo común.

Cada arco de la red, representa una actividad. Cada actividad es una de las tareas que requiere el proyecto. Así un nodo representa un suceso que se define como el momento en que se terminan todas las actividades que llegan a ese nodo. El sentido de las flechas indica el orden en el que debe ocurrir cada uno de esos sucesos. Un suceso debe preceder al inicio de las actividades que salen de ese nodo. El nodo hacia el que se dirigen todas las actividades es el suceso que corresponde a la terminación del plan de tareas. La red puede representar un plan de tareas desde su inicio, o bien, si el plan de tareas ya está iniciado, representa el plan para su terminación. En último caso, cada nodo de la red representa el suceso de continuar una actividad en marcha o el suceso de iniciar una nueva actividad que puede comenzar en cualquier momento.

Cada flecha juega un doble papel, el de representar una actividad y el de ayudar a representar las relaciones de precedencia entre las distintas actividades. En ocasiones, se necesita una flecha para definir las relaciones de precedencia, aunque no haya una actividad real que representar. En este caso, se introduce una actividad ficticia que se muestra como una flecha pintada con línea discontinua, que indica esta relación de precedencia. La actividad ficticia o fantasma no consume tiempo y cumple dos objetivos:

- salvar una relación de precedencia,

- individualizar cada actividad, es decir, que cada actividad tenga un único par de nodos que la identifique.

Para realizar una red de un plan de tareas se deben contestar tres preguntas básicas sobre cada actividad específica:

- ¿qué actividades deben ser realizadas inmediatamente antes de la ejecución de una actividad específica?
- ¿qué actividades deben llevarse a cabo inmediatamente después de realizar una actividad concreta?
- ¿qué actividades se pueden realizar simultáneamente a la ejecución de una activada específica?

Una regla común para construir este tipo de redes de planes de tareas es que dos nodos no pueden estar conectados directamente por más de dos flechas. Las actividades ficticias también pueden utilizarse para cumplir esta regla cuando se tienen dos o más actividades concurrentes. Además la numeración de los nodos permite identificar las diferentes actividades mediante los eventos de iniciación *i* y de terminación *j*. También se pueden numerar los nodos al azar y no hay razón por la que no se pueda o no se deba hacer. Sin embargo, la experiencia ha demostrado que el numerar los nodos de una manera especial hace más simple el procedimiento aritmético. Es buena práctica numerarlos de tal manera que el número precedente del nodo de cualquier flecha sea siempre menor que el del siguiente; en otras palabras *i* debe ser menor que *j*. Tampoco se numerarán nunca dos nodos diferentes con el mismo número.

En cuanto a las actividades, se dice que dos actividades son simultáneas cuando completamente o en parte pueden ser realizadas en un mismo intervalo de tiempo sin entorpecerse mutuamente. Dos actividades están ligadas cuando el inicio de una de ellas depende de que se hayan terminado la otra (u otras), como se puede ver en la Figura 3.3b con las actividades B y E. Además actividades concurrentes son aquellas que terminan en el mismo suceso y actividades divergentes las que empiezan en un mismo evento, como ocurre en la Figura 3.3a con las actividades A y B. Otro ejemplo de representación que muestra la Figura 3.3c es que para realizar la actividad N se requiere concluir primero las actividades L y M y para iniciar la actividad O se

requiere concluir sólo la actividad L. Además la Figura 3.3d muestra la representación de una red que incluye actividades ficticias y es que para realizar las actividades M, N se necesita haber acabado la actividad L y para realizar la actividad O se requiere haber concluido las actividades M y N.

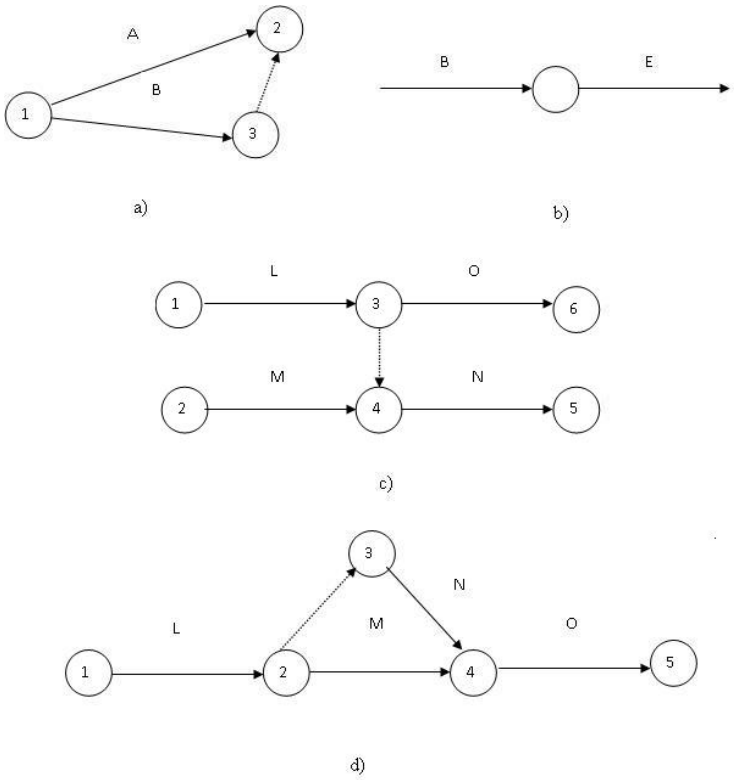


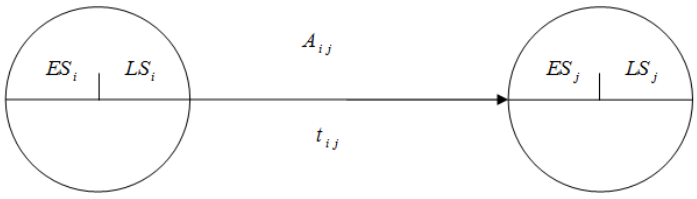
Figura 3.3 a) Actividades concurrentes y divergentes, b) actividades ligadas, c) representación de actividades y d) actividades ficticias en una red.

La única restricción existente hasta el momento para la correcta elaboración de la red, es el establecimiento lógico de la secuencia de actividades. Para realizar dicha secuencia se tiene en cuenta la información que contiene el planificador. Una vez realizada la red de actividades, se debe asignar la duración correspondiente a cada una de ellas. De esta forma el planificador calcula la duración total del proyecto y la

determinación de las fechas tempranas de realización de cada actividad. Para llevar a cabo estos cálculos el planificador tiene en cuenta:

- el comienzo del plan de tareas,
- no se debe iniciar ninguna actividad sin antes haber completado las tareas cuya ejecución depende de ella,
- que la realización de cada actividad debe iniciarse tan pronto como sea posible.
- que, una vez iniciada, cada actividad se ejecuta sin interrupción, hasta ser terminada.

Igual que es posible calcular las fechas o tiempos tempranos de iniciación y terminación de una actividad, el planificador realiza el mismo procedimiento de cálculo para obtener los tiempos tardíos de iniciación y de terminación de cada actividad con la duración total del proyecto. El cálculo de estos tiempos es muy sencillo, lo más pronto que una actividad se puede iniciar es el tiempo o la fecha más próxima en la que todas sus actividades precedentes se pueden terminar. Lo más pronto que se puede terminar una actividad es la fecha de iniciación más temprana a la que se suma el tiempo requerido para la terminación.



A_{ij} : Actividad que comienza en un evento "i" y finaliza en un evento "j"
ES_i : Tiempo más temprano de comienzo del evento o suceso "i"
LS_i : Tiempo más tardío de comienzo del evento o suceso "i"
t_{ij} : Duración de la actividad (i, j) con i < j
ES_j : Tiempo más temprano de comienzo del evento "j"
LS_j : Tiempo más tardío de comienzo del evento "j"

Figura 3.4 Representación de los nodos de una actividad.

El primer cálculo que hace el planificador es el de los tiempos tempranos de iniciación de cada actividad, como gráficamente muestra la Figura 3.4. El procedimiento es el siguiente:

1. Asigna al suceso de iniciación *i* de la primera actividad de la red una fecha u hora de inicio ES_i.
2. Se suma la duración de cada una de las actividades que preceden al suceso *j* y se anota la suma como ES_j. ES_j es también su tiempo más temprano de inicio para el suceso *j*. Se puede definir como:

$$ES_j = \max_i \left\{ ES_i + t_{ij} \right\} \quad con \quad i < j$$

3. En el caso de actividades cuyo suceso de terminación sea el mismo, se debe considerar el valor máximo que arrojen los cálculos del paso 2.
4. Se repiten los pasos 2 y 3 hasta que se calcule el tiempo temprano de realización ES_i, de todas las actividades.
5. La cifra final de tiempos tempranos de inicio constituye el tiempo en el que se puede llevar a cabo el proyecto.

El segundo cálculo que hace el planificador es el de los tiempos tardíos de terminación, como gráficamente muestra la Figura 3.4. El procedimiento es el siguiente:

1. El tiempo tardío de terminación del último suceso *j* es igual a su tiempo más temprano de iniciación ES_j. Se toma como dato inicial la duración total del proyecto y se anota en el extremo derecho del suceso final como LS_j.
2. Después se restan de LS_j las duraciones de cada una de las actividades que terminan en el suceso *j*. Siendo los valores de las restas el tiempo tardío de terminación LS_i. Se puede definir como:

$$LS_i = \min_j \left\{ LS_j - t_{ij} \right\} \quad con \quad i < j$$

3. Cuando dos o más actividades tengan el mismo suceso de inicio, debe considerarse el valor mínimo que arrojen los cálculos del paso 2 siendo este valor mínimo el tiempo tardío de terminación LS_i de las actividades anteriores.

Por otro lado el planificador también calcula la holgura TF *(Time Free)* para un suceso como la diferencia entre su tiempo más tardío y su tiempo más temprano para comenzar o terminar una actividad y se define como:

$$TF_{ij} = LS_j - ES_i - t_{ij} = LS_i - ES_i \quad \forall_{ij}$$

La holgura para un suceso indica cuánto retraso se puede tolerar para llegar a ese suceso sin retrasar la terminación del plan de tareas. La holgura para una actividad indica lo mismo respecto a un retraso en la terminación de esa actividad. Las actividades que tienen holgura cero son críticas, ya que cualquier retraso en ellas retrasa la terminación del plan de tareas. Si una actividad A es crítica se denota:

$$\underset{actividad\ crítica}{{}^{*}A_{ij}} \rightarrow \underset{tiempo\ crítico}{{}^{*}t_{ij}}$$

La ruta o camino crítico para un plan de tareas es una ruta a través de la red de tal forma que todas sus actividades tienen holgura cero. Es el camino más largo de la red, que corresponde al tiempo mínimo para finalizar el plan de tareas. La duración total del plan de tareas (D) es la suma de los tiempos críticos de las actividades críticas y se representa como:

$$D = \sum {}^{*}t_{ij}$$

Además las rutas críticas tienen una serie de propiedades que las caracteriza. Un plan de tareas siempre tiene una ruta crítica y a veces tiene más de una. Todas las actividades que tienen holgura cero deben estar en una ruta crítica, mientras que ninguna actividad que tiene holgura mayor que cero pueden estar en una ruta crítica. Todos los sucesos que tienen holgura cero deben de estar en una ruta crítica, mientras que ningún suceso que tiene holgura mayor de cero puede estar en una ruta crítica. Una trayectoria a través de una red tal que los sucesos de inicio y fin tienen holgura cero no necesariamente es crítica, porque puede suceder que una o más actividades sobre esa trayectoria pueden tener holgura mayor que cero.

Todo lo descrito anteriormente hace que el método CPM sea el más adecuado para ser integrado en agentes en el sistema descrito en el capítulo 4 de esta memoria. El método de planificación CPM, debido a su sencillez y eficacia resulta muy sencillo de integrar en la funcionalidad de un agente. Tal y como se ha mencionado, un planificador se define de forma general como un sistema que posee un objetivo, una

representación del estado actual del mundo, y genera una secuencia de acciones. Esta definición es perfectamente compatible con el componente planificador de un agente, en el que se posee un estado que pretende alcanzar, un conocimiento del sistema, y se ejecutan planes de acción para alcanzar un estado deseado. Si se asume esta hipótesis que indica que, en su forma de tratar con el mundo, un agente es esencialmente un planificador, es posible desarrollar un modelado de agentes donde el planificador siga el método CPM. A continuación se explica otro modelo de planificación que se utiliza en el marco de esta investigación. Se trata de CBP muy relacionada con CBR, como se ha mencionado en el punto 3.4 de este capítulo.

3.5.2 Planificación basada en casos

En la planificación basada en casos (CBP), 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. En la mente humana el concepto de intención se encuentra íntimamente ligado al concepto de plan. Los planes guían las acciones humanas. Hammond (1989) propuso la planificación basada en casos, entendiendo la planificación como una tarea de la memoria. De esta forma se establece una analogía entre la planificación y el recuerdo. Se podrán reutilizar, descartar o volver a aplicar recuerdos de aciertos, fallos o modificaciones realizadas en el pasado. Según Hammond, un sistema CBP utiliza el conocimiento que posee de su entorno y los efectos que tienen sus acciones en su entorno para construir planes. Así el plan construido sirve para testear dicho conocimiento. Las ventajas que presenta el CBP con respecto a otras aproximaciones a la planificación son que permite:

- anticiparse a posibles problemas que puedan aparecer durante la ejecución del plan y descartar aquellos planes que puedan presentar problemas,
- depurar aquellos planes fallidos,
- almacenar planes para su uso posterior en una memoria de casos.

Tal y como indican Cox *et al.*, (2006) CBP presenta una visión totalmente diferente a las de otras técnicas de planificación. En lugar de buscar el refinamiento del plan actual, se adaptan casos para resolver los nuevos problemas. Los casos

pueden incluir anotaciones que describan cómo fue obtenido el plan, o anticiparse a posibles fallos en el plan, tal y como había propuesto Hammond. Además pueden utilizarse distintas representaciones para los casos (plana, jerárquica, abstracta, etc.), lo que dota al proceso de razonamiento de una gran flexibilidad.

La planificación basada en casos es una especialización del razonamiento basado en casos, en la que las soluciones que se generan son planes. Al igual que en un sistema CBR, los conceptos fundamentales en un sistema CBP son la representación que se utilice para la memoria de planes y la implementación elegida para las etapas del ciclo CBP. Se trata de factores que influyen de forma decisiva en la eficiencia del planificador. En cuanto a las etapas que forman el ciclo CBP, son muy similares a las de los sistemas CBR, vistas en el punto 3.4.1 de este capítulo. Las etapas de recuperación y aprendizaje van a ser muy dependientes de la representación y organización elegidas para la memoria de planes.

3.6 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 (AOSE: *AgentOriented Software Engineering*). Existen distintas metodologías y herramientas de ingeniería de este tipo tales como *Gaia* [Wooldridge, *et al*., 2000], *ROADMAP* [Nasri Zazif *et al*., 2011], *MASCommonKADS* [Hasan *et al*., 2011], *AUML* [Beydoun *et al*., 2009], *Prometheus* [Gascueña *et al*., 2008], *MaSE* [García-Ojeda, 2008], *Tropos* [Giorgini, *et al*., 2011], *SysML* [SysML.org, 2010], etc., aunque muchas de ellas presentan importantes limitaciones y no disponen de herramientas de desarrollo especializadas.

La utilización de herramientas AOSE facilita enormemente el desarrollo de sistemas multiagente. En este trabajo de investigación se ha optado por hacer uso de la herramienta *SysML*, debido a que su integración permite obtener modelos mucho más cercanos a la implementación de los sistemas, de forma relativamente rápida y eficaz. El lenguaje de modelado *System Modeling Language* (*SysML*) que se detalla en el Apéndice B de esta memoria, es un robusto lenguaje para la ingeniería de sistemas basado en *Unified Modeling Language* (UML) [SysML.org, 2010] [UML, 2010].

Aunque este lenguaje no se considera exclusivamente dentro de las herramientas AOSE, proporciona numerosas características que le hacen adecuad para el diseño y representación de sistemas multiagente complejos e incluso ha logrado desplazar a lenguajes como AUML [AUML.org, 2010]. *SysML* permite el análisis y diseño detallado de sistemas que combinan elementos *hardware* como sensores con aplicaciones o componentes *software* de manera sencilla y eficaz.

3.7 Conclusiones

En este capítulo se ha estudiado el concepto de agente inteligente y de sistema multiagente. El hecho de que los agentes imiten el comportamiento humano y posean varias de las capacidades humanas hace que resulten muy adecuados para la construcción de sistemas de Inteligencia Artificial y que sean aplicables a un amplio rango de problemas de diferentes características como por ejemplo los sistemas sensibles al contexto.

Las características de los agentes deliberativos BDI, así como la posibilidad para modelar sus capacidades e integrar mecanismos de razonamiento y planificación, hacen que resulten adecuados para ser utilizados en entornos variados y muy dinámicos. Las arquitecturas deliberativas permiten modelar la estructura interna del agente en función de aptitudes mentales. Y ello permite que un agente BDI pueda resolver problemas siguiendo un proceso intencional, en el que las intenciones son planes, es decir, secuencias de acciones que se ejecutan para alcanzar un objetivo.

También en este capítulo se han presentado las diferentes técnicas de razonamiento y planificación, haciendo especial incidencia en el método de razonamiento CBR y el método de planificación CPM. 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. Además los sistemas también cambian teniendo en cuenta las características del contexto en una situación determinada.

Los agentes CBR-BDI permiten dotar a los agentes BDI de un sistema de memoria capaz de almacenar experiencias pasadas. También esto permite que los agentes puedan trabajar con dichas experiencias tomando aptitudes proactivas que les

permiten aprender a partir de sus experiencias. Esta capacidad de aprendizaje hace que los agentes tengan una mayor autonomía y una mayor capacidad de adaptación.

Por otra parte se ha presentado la herramienta para la ingeniería del software SysML, que permite obtener modelos de los sistemas multiagente cercanos a la implementación, facilitando la labor de los desarrolladores.

Sin lugar a dudas, los sistemas multiagente son una alternativa válida que merece la pena estudiar para abordar los desafíos que se plantean en el desarrollo de sistemas sensibles al contexto. Las capacidades de autonomía y adaptación hacen que los agentes CBR-BDI y el método de planificación CPM sean muy adecuados para ser aplicados en entornos sensibles al contexto que son altamente dinámicos y es necesario adaptarse rápidamente a los cambios que se producen. Los sistemas sensibles al contexto deben facilitar la interacción del usuario con la tecnología, haciendo que el usuario pueda hacer uso de los beneficios de la tecnología de forma transparente, por ello los agentes se presentan como una muy buena alternativa para su construcción.

Los conceptos detallados en este capítulo han permitido diseñar la arquitectura multiagente HoCCAC, descrita en detalle en el capítulo 4 de esta memoria. HoCCAC combina las ventajas de los sistemas multiagente, incorporando mecanismos de razonamiento y planificación que procesan información contextual. De esta forma HoCCAC, satisface las necesidades identificadas en el desarrollo de sistemas sensibles al contexto.

4

Arquitectura HoCCAC

En este capítulo se presenta la arquitectura HoCCAC propuesta en el marco de este trabajo de investigación. Se trata de una arquitectura multiagente especialmente desarrollada para satisfacer las necesidades que requieren los sistemas sensibles al contexto. La utilización de agentes y sistemas multiagente facilita el desarrollo de sistemas sensibles al contexto simplificando la monitorización del entorno y adaptando su propio funcionamiento a las necesidades del usuario. Para optimizar el rendimiento de los sistemas sensibles al contexto se propone la utilización de una arquitectura basada en agentes inteligentes. Esta inteligencia debe permitir a los agentes disponer de unas grandes capacidades de aprendizaje y de adaptación a los cambios que se producen en el entorno. Además los agentes deben ser capaces de elaborar y hacer cumplir planes de tareas personalizados para los usuarios que varían al mismo tiempo que el entorno. La arquitectura HoCCAC también tiene en cuenta la necesidad de ejecutar agentes en dispositivos inalámbricos, con pocos recursos de memoria y de almacenamiento. Por otro lado, los agentes en HoCCAC son capaces de integrarse con tecnologías inalámbricas de manera transparente para el usuario.

A continuación se realiza una introducción al capítulo. En la sección 4.2 se presenta la arquitectura HoCCAC, describiendo de forma general los principales elementos que la componen, entre los que destaca la plataforma de agentes. En esta sección también se describe en detalle cada tipo de agente. En la sección 4.3 se describe el funcionamiento de los agentes en HoCCAC y su interacción sobre los dispositivos en el entorno. En la sección 4.4 se describe el modelo de datos del sistema. En la sección 4.5 se muestran las soluciones aportadas por el agente Intérprete a través de los métodos de razonamiento y planificación que incorpora. En la sección 4.6 se describe un diseño detallado del agente Intérprete. Finalmente, se presentan las conclusiones correspondientes a este capítulo.

4.1 Introducción

En la actualidad, los 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, la computación sensible al contexto trata de integrar la tecnología automáticamente en el contexto de los usuarios. Para conseguirlo 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. Son varias las deficiencias principales que se encuentran en los sistemas sensibles al contexto desarrollados [Bettini et al., 2009] [Bricon-Souf y Newman, 2007] [Meier *et al.,* 2006] [Koskinen y Suomela, 2006] [Kleinberger et al., 2007] [Wu *et al.,* 2007] [Niemelä *et al.,* 2007] [Emiliani y Stephanidis, 2005] [Yoerger *et al.,* 2005] [Korpiää *et al.,* 2003] [Biegel *et al.,* 2004] [Chen, 2004a] [Moon *et al.,* 2006] [Yamabe *et al.,* 2005]. Por un lado, son pocos los sistemas que usan información de diferentes atributos de contexto y relacionan esta información para interactuar sobre los usuarios. Por otro lado los sistemas desarrollados se centran en resolver problemas muy concretos y por último los sistemas desarrollados no integran la combinación de mecanismos de razonamiento y planificación de actividades que permitan adaptar el

contexto a las necesidades del usuario. Así pues, se detecta la necesidad de definir una arquitectura capaz de:

- Modelar un contexto tecnológicamente avanzado y que proporciona información contextual poco relacionada entre sí.
- Facilitar la captura de información contextual y su clasificación.
- Utilizar toda la información disponible para adaptar el contexto a las necesidades del usuario.

Por estos motivos en este trabajo de investigación se presenta una arquitectura distribuida (HoCCAC) cuyas principales características son la utilización de agentes Proveedores de información y la integración de mecanismos de planificación (el método CPM) y razonamiento (CBR) en un agente inteligente (el agente Intérprete). Los agentes Proveedores interaccionan autónomamente con el entorno para capturar la información contextual y clasificarla y el agente Intérprete utiliza la información disponible para razonarla y generar planes de tareas que mejoren la adaptación del usuario al contexto.

HoCCAC es una arquitectura multiagente especialmente diseñada para facilitar la construcción de sistemas sensibles al contexto. Para ello proporciona capacidades avanzadas para solucionar las carencias que presentan las soluciones actuales, así como para modelar el contexto e interactuar con él. Los agentes desarrollados en el marco de esta investigación incorporan mecanismos de razonamiento y planificación adecuados para la construcción de sistemas multiagente. En este sentido, se debe disponer de agentes capaces de interactuar con los usuarios de forma inteligente, agentes capaces de interactuar con los dispositivos de un entorno de forma automática, mecanismos que faciliten una comunicación ubicua, y mecanismos que faciliten una computación sensible al contexto de los agentes. El objetivo de la arquitectura multiagente HoCCAC se ha centrado inicialmente en un entorno concreto: facilitar la asistencia de usuarios dependientes en su hogar, pero puede ser fácilmente extensible a entornos de características similares.

La arquitectura proporciona un mecanismo novedoso que integra en los agentes inteligentes un modelo de planificación de tareas basado en el método CPM, descrito en el capítulo 3 de este trabajo de investigación. El método CPM o del camino crítico

conecta de manera optimizada una serie de actividades relacionadas entre sí para alcanzar un objetivo determinado. Con la utilización del método de planificación de tareas CPM los agentes actúan de manera proactiva sobre el entorno. HoCCAC utiliza la información del entorno sensible al contexto para predecir las necesidades de los usuarios y proporcionar soluciones efectivas. HoCCAC además integra un tipo de agente CBR-BDI capaz de aprender a partir de un conocimiento inicial. Los agentes CBR-BDI interaccionan de forma autónoma con el entorno y con los usuarios del sistema para adaptarse a las necesidades del entorno [Fraile *et al.,* 2010c] [Corchado *et al.,* 2011]. La integración en los agentes CBR-BDI y del método de planificación de tareas CPM hace que el sistema inteligente HoCCAC sea capaz de mejorar las planificaciones y aprender con el paso del tiempo. HoCCAC tiene un modelo de datos sensible al contexto definido para la optimización de la gestión de la información donde almacena su base de conocimiento.

HoCCAC 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 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 WiFi, y la identificación por radiofrecuencia junto con la tecnología NFC, explicadas en detalle en el apéndice A, componen un esquema de tecnologías inalámbricas que se integran en el entorno de manera natural para los usuarios. La sencilla integración e interacción de agentes inteligentes, métodos de planificación, agentes Proveedores de información, sensores, dispositivos y un modelo de datos definido es lo que nos lleva a proponer la arquitectura HoCCAC que se describe a continuación.

4.2 Arquitectura Propuesta

La arquitectura HoCCAC es un sistema distribuido, diseñado para facilitar la construcción de sistemas sensibles al contexto compuesto por agentes inteligentes que

razonan y desarrollan planes de actividades optimizados basados en el método de planificación CPM y en el método de razonamiento CBR. La arquitectura HoCCAC se centra principalmente en la captura y clasificación de información contextual y el seguimiento y el control de planes de actividades que tratan de adaptar el contexto a las necesidades del usuario. HoCCAC se define por la necesidad de controlar dispositivos distribuidos y recoger información de usuarios de manera no intrusiva y automática en entornos sensibles al contexto. Además HoCCAC aporta tratamiento, almacenamiento y razonamiento sobre la información sensible al contexto y seguridad en la gestión de accesos y administración de la información. El sistema HoCCAC es capaz de percibir información de su entorno a través de sensores. De la información percibida se obtienen los datos de contexto para que los agentes inteligentes procesen la información recibida y actúen en su entorno de manera correcta. Con los datos proporcionados, los agentes inteligentes crean o actualizan planes de actividades, tendiendo siempre a maximizar y mejorar el resultado esperado. Además HoCCAC combina la gestión de información personal con un modelo de actividades diarias, definido por los agentes inteligentes, a partir de los datos que proporcionan los sensores instalados en la red contextual. La interpretación y procesamiento del conocimiento base y de los modelos diarios desarrollados por los agentes inteligentes proporciona un valor añadido al sistema. Así HoCCAC se adelanta a las peticiones o posibles incidencias que tiene un usuario en su contexto y ofrece soluciones basadas en experiencias pasadas que mejoran los resultados esperados [Bajo *et al.,* 2007] [Fraile *et al.,* 2010b].

Para obtener los datos del contexto, procesarlos y ofrecer soluciones a los usuarios, el sistema HoCCAC se basa en una arquitectura multiagente que está compuesta por varios tipos de agentes inteligentes como se pueden ver en la Figura 4.1. Estos agentes conocen su entorno y actúan sobre él. En HoCCAC, los servicios ofrecidos son gestionados y personalizados por agentes basados en el modelo deliberativo BDI. La personalización se basa en las preferencias, 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. Adicionalmente HoCCAC proporciona a los agentes mecanismos de acceso a los servicios del sistema a través de dispositivos móviles.

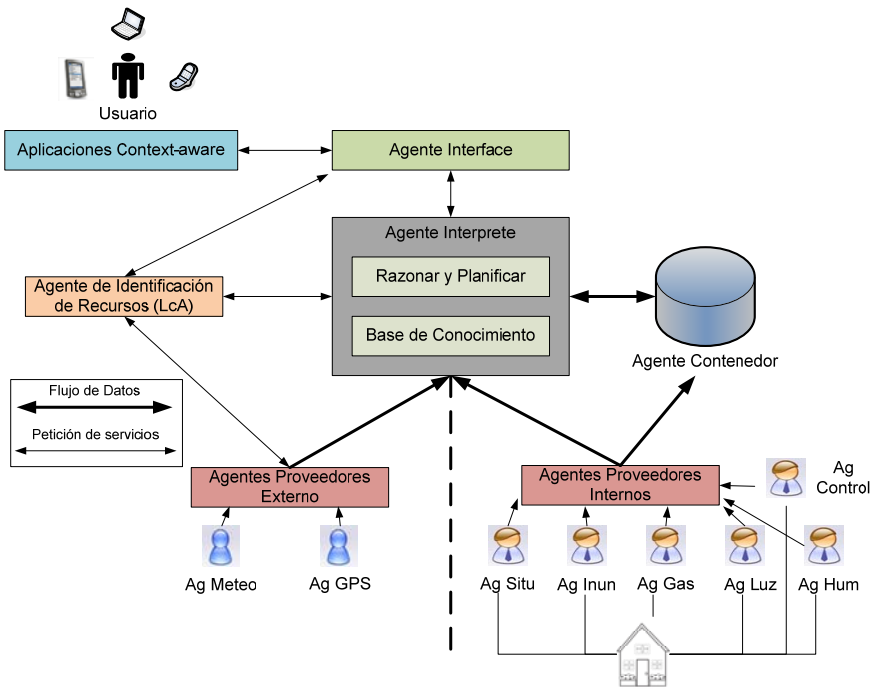


Figura 4.1 Descripción general del sistema multiagente HoCCAC.

La arquitectura multiagente está formada por los siguientes tipos de agentes:

- **Agentes Proveedores** que capturan los datos obtenidos de fuentes heterogéneas de contexto, internas y externas, para que los agentes Intérprete y Contenedor puedan tratar y reutilizar estos datos. Dada la gran cantidad de dispositivos y la distinta naturaleza de cada uno de ellos, parece razonable utilizar un tipo de agente Proveedor que permita interactuar con cada dispositivo de forma sencilla y eficiente. Así, un agente Proveedor interpreta la información

que recibe desde el sensor correspondiente, la analiza y la envía al agente Intérprete dentro del sistema multiagente. De la misma forma, cuando sea necesario comunicarse sobre un actuador, el agente Proveedor es el encargado de poner la información en la forma adecuada y comunicarse con el dispositivo. Por ejemplo en el caso de dispositivos RFID es necesario procesar las señales que se reciben desde un lector RFID para que puedan ser utilizadas como información útil por el sistema multiagente. En este caso el agente Proveedor actúa como interfaz entre el dispositivo RFID y el resto del sistema, interrogando al lector e interpretando la información que recibe de él.

Los agentes Proveedores se encuentran situados en nodos de red que se comunican vía WiFi con el agente LcA y el agente Intérprete. Estos agentes Proveedores deben ser tan sencillos como sea posible para minimizar su impacto sobre los nodos en los que se ejecutan. Los agentes Proveedores se comunican con los sensores o actuadores emitiendo comandos y recibiendo respuestas con el estado del dispositivo. La utilización de agentes capaces de interactuar con dispositivos facilita la implementación de la arquitectura en diferentes entornos y la compatibilidad con distintos dispositivos.

El agente Proveedor se encarga de dialogar con los dispositivos y de traducir sus datos a un formato entendible por el resto de agentes del sistema. La utilización del agente Proveedor facilita por tanto la interacción con dispositivos de distinta naturaleza de forma transparente y no invasiva para los usuarios y para el resto del sistema. Algunos de los tipos de agentes Proveedores que se pueden conectar al sistema son:

- Agente de situación, que mantiene continuamente localizado al paciente en el hogar.
- Agente detector de inundaciones, que se encarga de detectar fugas de agua o de cualquier otro líquido.
- Agente detector de gas, para comprobar la existencia de gas en el ambiente.
- Agente detector de humos, que se encarga de detectar la presencia de humo en el aire.

- Agente detector de luminosidad, que se encarga de medir la cantidad de luz que llega a una célula foto-eléctrica.
- Agente termostato, que se encarga de abrir o cerrar un circuito eléctrico en función de una temperatura.
- Agente de control, que establece y controla los parámetros diarios que desea el paciente.
- Agente meteorológico, que registra el tiempo en el exterior para poder actuar en consecuencia sobre las variables internas en el hogar.
- Agente GPS, que mantiene localizado al paciente fuera de su hogar.

El sistema es dinámico y tiene la capacidad de incorporar nuevos agentes Proveedores de información en cualquier momento añadiendo el sensor correspondiente o capturando la información necesaria de un servidor o Proveedor externo.

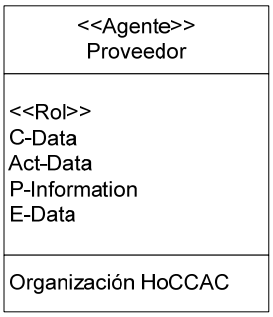


Figura 4.2 Roles del Agente Proveedor.

Como muestra la Figura 4.2, los principales roles de un agente Proveedor son la captura de información del sensor o dispositivo de contexto (*C-Data*), la actuación sobre el sensor (*Act-Data*), el tratamiento de los datos que recoge del sensor (*P-Information*) y el envío de información al resto de agentes del sistema (*E-Data*).

- **Agente Contenedor** que almacena los datos relativos al contexto en un modelo de datos orientado a objetos que se describe en el punto 4.4 de este capítulo. Para almacenar la información el agente utiliza el lenguaje de consultas orientadas a

objetos *Object Constraint Language*[3] (OCL). Este agente se encarga de la gestión de los dispositivos, de su localización y un detalle del contexto en el que actúa. Además también gestiona la información antigua relativa a usuarios que ya no interactúan con el entorno y dispositivos desactivados.

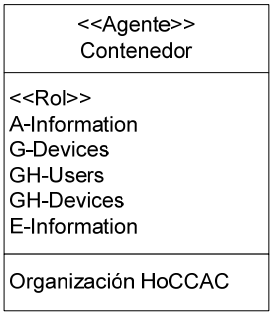


Figura 4.3 Roles del Agente Contenedor.

Como muestra la Figura 4.3, los principales roles de un agente Contenedor son el almacenamiento de la información que recibe (*A-Information*), la gestión de sensores y dispositivos en su contexto (*G-Devices*), la gestión de información histórica de usuarios (*GH-Users*) y dispositivos (*GH-Devices*) y el envío de información solicitada por otro agente del sistema (*E-Information*).

- **Agente Intérprete** proporciona servicios de razonamiento lógico para procesar la información de contexto. Se basa en el concepto de agente BDI e incorpora el mecanismo de razonamiento CBR para mejorar su autonomía y aumentar sus capacidades de resolución de problemas. La información sensorial es interpretada para obtener la información de contexto y actualizar el estado del paciente. El agente Intérprete a menudo recibe la misma información de sensores cercanos entre sí, por este motivo, es capaz de clasificar e interpretar la información que recibe y no duplicar información a la hora de almacenarla en el sistema. Razona sobre las acciones a realizar en el sistema utilizando la información disponible en el sistema, las señales recibidas por el agente Interface y la base de conocimientos y creencias relativa al contexto que guarda. De esta forma crea un razonamiento que determina un curso de acciones óptimas

[3] http://www.omg.org/technology/documents/formal/ocl.htm

o planes para que el usuario alcance un objetivo. En este sentido el agente Intérprete utiliza el mecanismo de planificación CPM para generar planes como soluciones que se ejecutan en el espacio de tiempo más corto posible. El agente Intérprete proporciona la habilidad para alcanzar los objetivos de alto nivel y evita errores que pueden conducir a ineficacias. También permite mayor flexibilidad ante la presentación de nuevos objetivos y el desarrollo de replanificaciones en tiempo de ejecución.

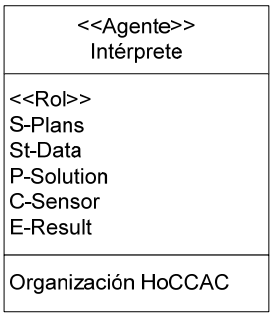


Figura 4.4 Roles del Agente Intérprete.

Como muestra la Figura 4.4, los principales roles de un agente Intérprete son la generación de planes de tareas (*S-Plans*), almacenamiento del conocimiento (*St-Data*), revisión de la ejecución de los planes de tareas (*P-Solution*), recuperación de la información necesaria (*C-Sensor*) y reutilización de los planes ya ejecutados (*E-Result*). Estos roles se explican más en detalle en la sección 4.6 de este capítulo, dedicada al diseño de este agente.

- **Agente de Identificación de Recursos (LcA)**. Este agente se encarga de mantener un registro y administrar los agentes Proveedores activos en el sistema así como de permitir o denegar la inclusión de nuevos agentes Proveedores. El agente LcA informa de cualquier cambio al agente Interface y al agente Intérprete para que ambos puedan tener en cuenta los nuevos atributos de contexto que proporciona por ejemplo la inclusión de un nuevo agente Proveedor.

  Como muestra la Figura 4.5, los principales roles de un agente LcA son la gestión de la activación o desactivación de agentes Proveedores (*G-Provider*) y

la información al resto de agentes del sistema de la activación o desactivación de dichos agentes Proveedores (*Inf-Provider*).

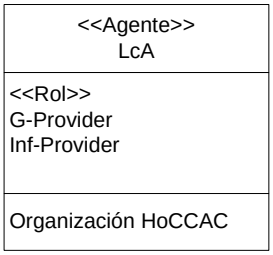


Figura 4.5 Roles del Agente LcA.

- **Agente Interface** se comunica con el agente Intérprete y el agente LcA. El usuario puede interactuar con el sistema de forma explícita a través del agente Interface. El agente Interface lee las entradas que proporciona el usuario a través de las aplicaciones sensibles al contexto y envía modificaciones del comportamiento al agente Intérprete o agente LcA. El agente Interface también envía notificaciones a los usuarios a través de las aplicaciones sensibles al contexto. El agente Interface por ejemplo, puede recibir entradas de aplicaciones sensibles al contexto, durante un largo período de tiempo, antes de decidir tomar una sola acción. También puede suceder que el agente Interface reciba una entrada de una única aplicación sensible al contexto que puede lanzar una serie de acciones por parte del agente Interface.

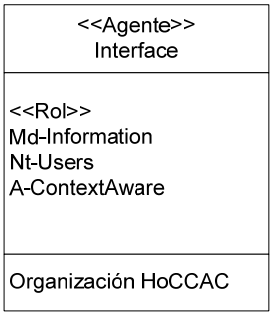


Figura 4.6 Roles del Agente Interface.

Como muestra la Figura 4.6, los principales roles de un agente Interface son la recogida de información explícita del usuario (*Md-Information*), el envío de alertas o notificaciones a usuarios (*Nt-Users*) y la gestión de las aplicaciones *context-aware* del sistema (*A-Context-Aware*).

La arquitectura HoCCAC permite la integración de aplicaciones sensibles al contexto. Las aplicaciones sensibles al contexto, son los programas que pueden usarse para aprovechar las funcionalidades del sistema. HoCCAC hace posible que en espacios físicos variables las aplicaciones sensibles al contexto puedan ser utilizadas y compartidas fácilmente. Las aplicaciones sensibles al contexto integradas en la arquitectura HoCCAC como muestra la Figura 4.1, reciben la información disponible en el sistema a través del agente Interface. Además utilizan diferentes niveles de información de contexto y adaptan la forma de actuar de acuerdo al contexto activo. Las aplicaciones sensibles al contexto son dinámicas y adaptables, reaccionando siempre sobre los cambios de estado en el entorno. Las aplicaciones sensibles al contexto pueden ser ejecutadas localmente o en remoto, incluso en dispositivos móviles con capacidad limitada de proceso. Una forma de desarrollar aplicaciones de este tipo es especificando acciones que respondan a cambios de contexto ante unas determinadas condiciones y reglas. De esta manera se incorpora a una aplicación un grupo de reglas y condiciones que lanzan la puesta en marcha de una acción sobre el contexto de usuario. Se trata de información estática y con capacidad de influencia sobre el contexto. Las aplicaciones sensibles al contexto reciben las reglas y condiciones y sus acciones asociadas a través del agente Interface cuando se trata de tareas específicas que se repiten a menudo. De esta forma las aplicaciones sensibles al contexto liberan la carga de proceso del agente Intérprete. La diferencia principal entre este tipo de aplicaciones y el agente Intérprete es que este tiene capacidad de ir mejorando los resultados que ofrece con el paso del tiempo mientras que las aplicaciones sensibles al contexto van a ofrecer los mismos resultados siempre, ante unas condiciones específicas del contexto.

En la Figura 4.7 muestra una visión general de la infraestructura del sistema distribuido HoCCAC. En esta imagen se ve cómo los portátiles y dispositivos móviles se conectan al sistema a través de tecnologías inalámbricas. Además todos los dispositivos están interconectados a través de redes de comunicación inalámbrica,

redes móviles o tecnología RFID. El sistema HoCCAC facilita la integración y gestión de agentes, dispositivos y sistemas de control.

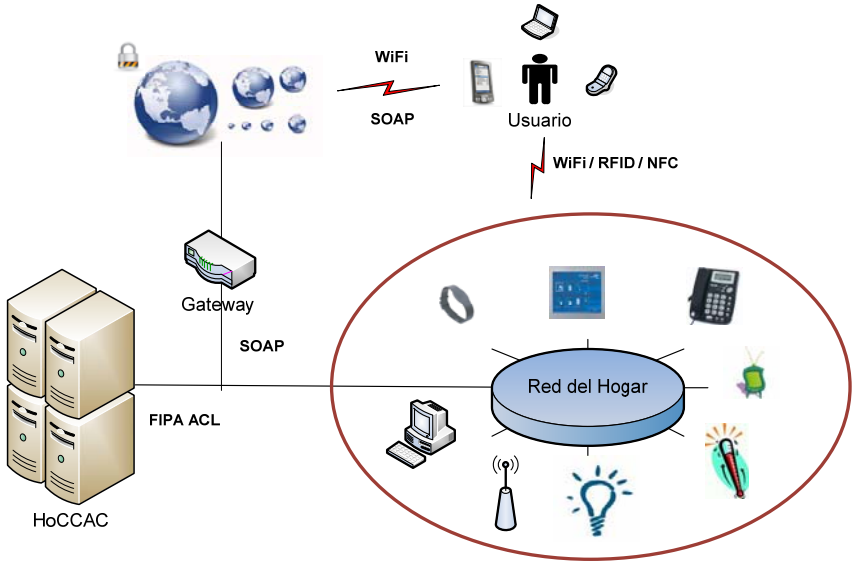


Figura 4.7 Descripción general del sistema multiagente HoCCAC.

La comunicación entre agentes en la plataforma sigue el estándar *Foundation for Intelligent Physical Agents* (FIPA) *Agent Communication Language* (ACL). El protocolo de comunicación entre los agentes y los servicios está basado en el estándar *Service Oriented Architecture Protocol* (SOAP) [Corchado *et al.,* 2008a]. Los servicios que invocan los agentes pueden ser de dos tipos, servicios que capturan información del contexto a partir de los datos que se obtienen de los componentes autónomos y servicios que actúan sobre los sistemas de control automático y dispositivos instalados en el contexto. Todo esto permite que HoCCAC sea un sistema fácil de implantar en entornos complejos y que no dependa de dispositivos o de una plataforma específica.

En el siguiente punto se describe el funcionamiento de los agentes en HoCCAC y como pueden interactuar sobre distintos tipos de dispositivos en el entorno.

4.3 Interacción entre agentes y dispositivos en el sistema

Aunque la optimización de las funcionalidades internas de HoCCAC es muy importante, también lo es una adecuada interacción con los usuarios, dispositivos y el entorno. Una incorrecta interacción conlleva el desarrollo de complicados sistemas que no cumplen los requerimientos de un sistema sensible al contexto 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 HoCCAC sea capaz de afrontar los retos que plantean los desarrollos de sistemas *context-aware*, es necesario que los agentes sean sensibles al contexto, obteniendo información tanto de los usuarios como del entorno. Una de las innovaciones propuestas por HoCCAC consiste en la utilización de un conjunto de tecnologías que permiten a los agentes obtener dicha información de manera no intrusiva para el usuario. 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 deben reaccionar de forma automática, eligiendo una de las posibles soluciones con las que cuenta el sistema para esa situación. Por la propia naturaleza flexible de HoCCAC, no es conveniente definir el uso de una tecnología en particular. Sin embargo, existen tecnologías de referencia que son consideradas por HoCCAC debido a su propio diseño permiten ser aplicadas en gran cantidad de contextos.

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. 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.

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 los sistemas sensibles al contexto 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 *context-aware* es ZigBee. Existe una gran cantidad de dispositivos ZigBee y su utilización en escenarios domóticos y *context-aware* es cada vez mayor. La versatilidad de esta tecnología permite dotar a los agentes prácticamente de cualquier capacidad sensorial, por ejemplo, iluminación, humedad, humo, etc. El conjunto de tecnologías RFID, ZigBee y WiFi representan un gran avance hacia el desarrollo de sistemas *context-aware*. Es importante señalar que la simple utilización de nuevas tecnologías no es suficiente para que un sistema cumpla con los requisitos de los sistemas *context-aware*. Por lo tanto, es necesario diseñar mecanismos e interfaces que hagan uso de dichas tecnologías y faciliten a los usuarios su utilización. En HoCCAC, esto se logra a través del agente Interface o mediante aplicaciones *context-aware* independientes. Los agentes Proveedores obtienen información de forma constante y la envían al núcleo central de la plataforma de agentes para que sea procesada e interpretada, como muestra el diagrama de interacción de la Figura 4.8.

Por su parte, el agente Interface es una de las piezas más importantes para lograr una adecuada interacción con los usuarios. Este tipo de agente interpreta 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 del agente Interface son modeladas a través de aplicaciones, 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.

La Figura 4.8 muestra un diagrama de interacción donde se muestra como un usuario recibe la información de un sensor de contexto. Los nuevos datos de contexto los recibe el agente Proveedor y este agente los envía al núcleo central de HoCCAC para que sean procesados. El usuario, por su parte es informado a través del agente

Interface. El usuario puede modificar la parametrización de un sensor a través del agente Interface como muestra la Figura 4.8 a través del diagrama de interacción representado.

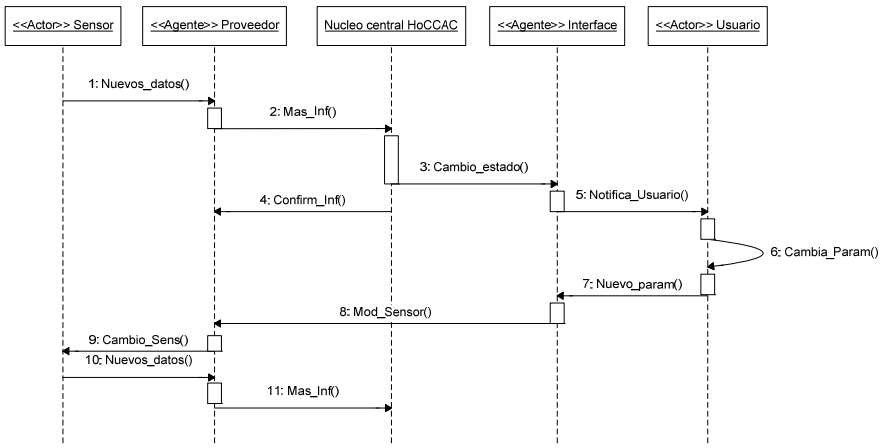


Figura 4.8 Diagrama de interacción: Cambio de parametrización en sensor.

Por otro lado, los agentes Proveedores externos obtienen información de contexto a través de componentes autónomos externos como por ejemplo un servidor de información meteorológica que proporciona información sobre el tiempo en un determinado lugar o un servidor de localización que puede proporcionar información sobre la ubicación de una persona cuando no está en su hogar. Del mismo modo, los agentes Proveedores internos recogen directamente información de sensores instalados en el entorno, como por ejemplo sensores de localización basados en RFID existen en el hogar de un paciente, o sensores de luz. Además, los agentes Proveedores pueden actuar sobre el usuario a través de actuadores situados en el contexto. Los agentes Proveedores son una de las piezas más importantes para lograr información contextual a cerca de los usuarios. Este tipo de agentes interpretan la información enviada por la plataforma de agentes y la presentan a los usuarios.

Para entender mejor esta interacción, se describe el siguiente ejemplo: un servicio de localización recibe constantemente información de dispositivos RFID. Esta información es enviada periódicamente a la plataforma de agentes. La información es

interpretada y enviada al dispositivo móvil del usuario para saber su posición dentro una determinada área. De esta forma, el dispositivo móvil 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 en la pantalla.

Con toda la información recibida, las funciones del agente Intérprete son por un lado el procesamiento de información que le proporciona el agente Contenedor y por otro lado el procesamiento de la información del contexto que le proporcionan los agentes Proveedores. El modelo de datos sobre el que interactúa el agente Contenedor se muestra en la Figura 4.11 del apartado 4.4 de este capítulo. En el mismo apartado de este capítulo (4.4) hay un ejemplo que representa la información que recibe el agente Intérprete de los agentes Proveedores y se muestra en la Figura 4.12. Con la información conseguida a partir de su procesamiento y basándose también en el método de razonamiento CBR el agente Intérprete desarrolla planes de actividades y lanza servicios que actúan sobre el usuario y su contexto. El agente Intérprete desarrolla y controla la ejecución de estos planes de actividades mediante el método CPM.

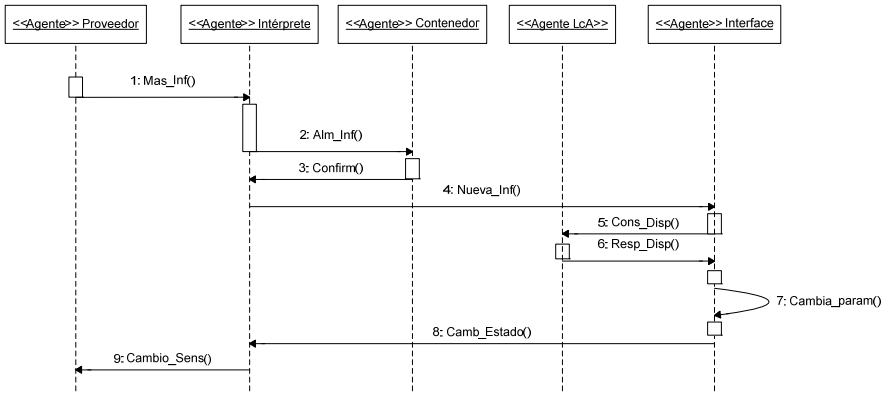


Figura 4.9 Diagrama de interacción: Comunicación entre agentes en HoCCAC.

La Figura 4.9 muestra un ejemplo de comunicación entre los agentes del sistema HoCCAC. Primeramente la información la recoge un agente Proveedor, luego pasa

por todos los procesos de almacenamiento y tratamiento de la información del resto de agentes y finalmente el agente Proveedor vuelve a interactuar sobre un dispositivo del sistema. Por ejemplo cuando los agentes Proveedores detectan a un nuevo usuario en el contexto, informan al resto de agentes, como muestra el mensaje 1. Luego el agente Intérprete procesa la información almacenada relativa al usuario y determina la temperatura y luminosidad deseada por el usuario en el entorno, basándose en experiencias y datos pasados. El agente Intérprete genera planes de tareas y actúa sobre los sensores y actuadores instalados en el contexto a través de las aplicaciones *context-aware,* como muestran los mensajes del 4 al 9.

Las aplicaciones *context-aware* utilizan diferentes niveles de información de contexto, pudiendo así adaptar su comportamiento. Estas aplicaciones al consultar los datos registrados en el sistema de información por el agente LcA, pueden localizar los servicios de todos los Proveedores de contexto que le interesen. Las aplicaciones *context-aware* para obtener datos del contexto lo pueden hacer preguntando al agente Interface o esperando un evento del agente Interface. El LcA permite a los usuarios y agentes localizar las diferentes aplicaciones del contexto. Las principales características del LcA incluyen la escalabilidad, la adaptabilidad, y capacidad de procesamiento múltiple. El LcA controla grandes áreas donde actúan los Proveedores de contexto, que pueden estar localizados en redes internas o externas. El LcA rastrea y se adapta a los cambios que se introducen en el contexto al añadir o eliminar sensores físicos o reconfiguraciones de los mismos dispositivos. También despliega un mecanismo para permitir a los Proveedores de contexto informar de su funcionalidad al sistema.

En la Figura 4.10 se muestra un ejemplo para el control de la localización y monitorización de un usuario en su contexto, donde el *Agente Proveedor interno Situ* envía señales al agente Intérprete para comunicarle la situación del usuario. El *Agente Situ* también recibe señales del agente Intérprete para, por ejemplo, permitir o prohibir el acceso a zonas controladas al usuario. Por otro lado el agente LcA tiene registrados todos los servicios que ofrece el *Agente Situ*. El agente *LcA* mantiene continuamente actualizada la lista de servicios proporcionada por el *Agente Situ* para así poder trasladársela a las aplicaciones activas en el contexto. Además el agente Contenedor gestiona la información del contexto a partir de los datos proporcionados por el agente

Intérprete. El agente Contenedor gestiona toda la información recibida usando el modelo de datos que se describe en el siguiente apartado de este capítulo.

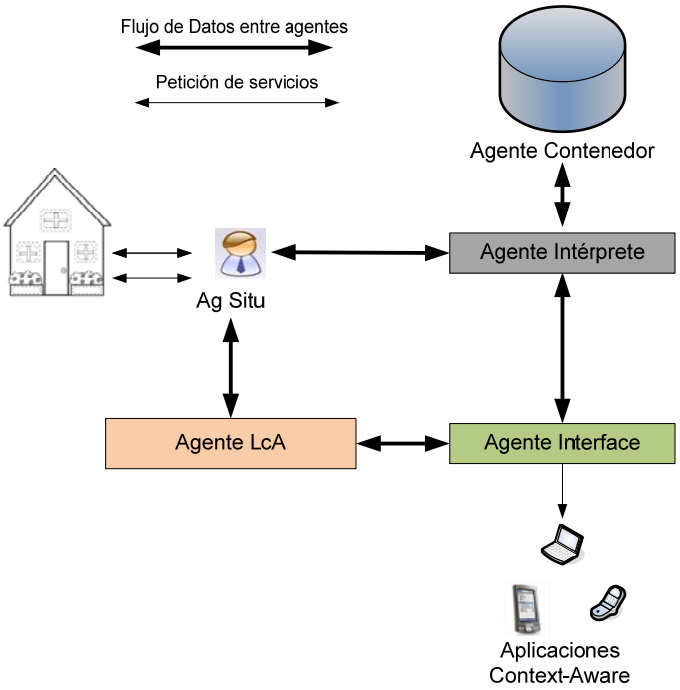


Figura 4.10 Ejemplo de interacción entre agentes y aplicaciones *context-aware*.

4.4 Descripción del modelo de datos

El objetivo principal del sistema HoCCAC es facilitar la captura de información contextual y poder generar planes de tareas o re-planificar planes existentes en tiempo de ejecución. Para ello un conjunto de dispositivos y aplicaciones *context-aware* capturan información del contexto de usuario y su rutina. Toda esta información es recogida por los agentes definidos en el sistema y el agente Contenedor se encarga de almacenar la información relativa a la gestión del contexto en el modelo de datos del sistema que se muestra en la Figura 4.11. Como se puede ver en dicha figura el modelo del dominio almacena las preferencias de varios usuarios sobre los dispositivos. Las reglas de acciones se especifican en formato texto en función de los

atributos, métodos y operadores especificados para los dispositivos. Las reglas de actuación se interpretan mediante un *parser* de modo similar a como se interpreta un árbol de decisión por parte de software de minería de datos como Weka [Hall *et al.,* 2009]. Las acciones se pueden especificar de modo manual o bien se generan automáticamente mediante un modelo deliberativo. A partir de las preferencias y la información de contexto almacenada en el sistema de información de HoCCAC el agente Intérprete ejecuta los servicios disponibles para actuar sobre el contexto y el usuario.

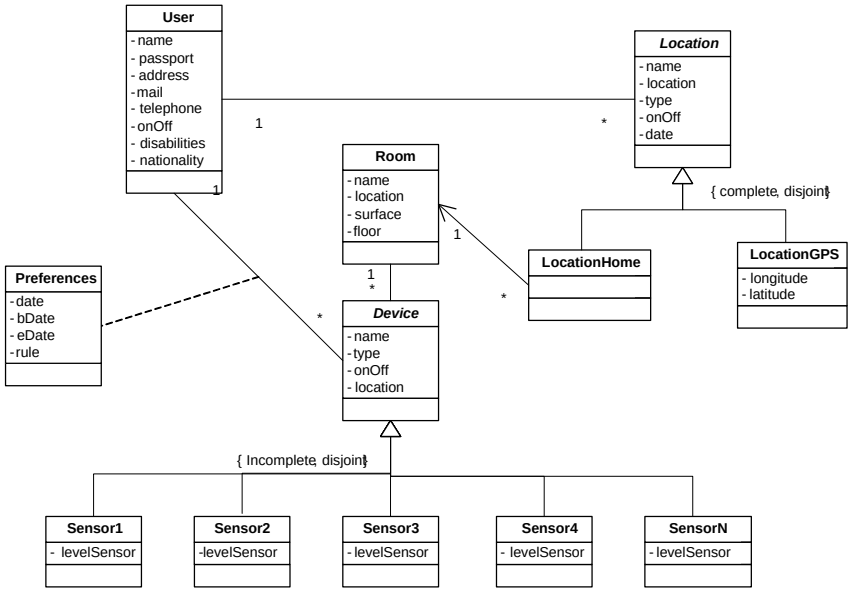


Figura 4.11 Modelo del dominio del sistema HoCCAC.

El agente Intérprete guarda en su base de conocimiento la información que indica la situación y estado actual del usuario en el contexto para poder actuar de forma rápida sobre el usuario a través de los dispositivos más cercanos a él. Por ejemplo, en caso que el usuario se encuentre fuera del entorno *context-aware* el agente Proveedor GPS se encarga de registrar su posición periódicamente a través de las coordenadas GPS. Cada uno de los valores que captan estos sensores es recogido por los agentes

Proveedores correspondientes y enviados al agente Intérprete para que registre la información.

Los agentes en HoCCAC se intercambian información a través de mensajes ACL. Por ejemplo en un instante determinado el agente Intérprete recibe dentro de un mensaje ACL el fichero XML que se muestra en la Figura 4.12. La información de dicho fichero que muestra la Figura 4.12, se almacena en el dominio del sistema por el agente Contenedor para posteriormente ser explotada y tratada por el resto de agentes del sistema.

```
1<habitacion id="3">
2  <temperatura id="12">
3    <attribs name="devicet01" temperatura="22" type="dim" onOff="on"        date="sysdate" location="25,46,85" />
4  </temperatura>
5  <light id="7">
6    <attribs name="devicel04" light="5" type="dim" onOff="on" date="sysdate" location="22,12,44" />
7  </light>
8  <humo id="3">
9    <attribs name="devices08" humo="0" type="dim" onOff="on" date="sysdate" location="07,14,05" />
10 </humo>
11 <gas id="6">
12   <attribs name="deviceg02" gas="0" type="dim" onOff="on" date="sysdate" location="36,15,41" />
13 </gas>
14 <inundacion id="2">
15   <attribs name="devicef06" inundacion="1" type="dim" onOff="on" date="sysdate" location="23,58,42" />
16 </flood>
17</room>
```

Figura 4.12 Fichero XML recibido por el Agente Intérprete.

La información que muestra la Figura 4.12 está asociada a cinco sensores de contexto instalados en el caso de estudio que se describe en el punto 5 de este trabajo de investigación. Cada sensor recoge información de un tipo y en las especificaciones indican por ejemplo su localización a través del atributo *location*, si están activos o no a través del atributo *onOff*, el nombre que tienen asignado a través de atributo *name* y los atributos *temperatura*, *light*, *humo*, *gas* e *inundación* el valor de cada uno de los sensores. Por ejmplo hay sensores como *deviceg02* que sólo puede tener el valor 0, cuando no detecta escape de gas o 1 cuando si detecta un escape de gas. La forma de ofrecer soluciones, intercambiar información y los métodos de razonamiento y planificación utilizados por el agente Intérprete es lo que se describe en el siguiente punto.

4.5 Soluciones proporcionadas por el agente Intérprete

El agente Intérprete integrado en el sistema HoCCAC tiene como objetivo proporcionar al usuario soluciones eficientes en tiempo de ejecución en su contexto para mejorar su calidad de vida. Sus características más importantes son:

- Capacidad de razonamiento, analiza y razona los datos de contexto recogidos por el sistema, desarrolla planes de tareas y proporciona soluciones proactivas.
- Facilidad de adaptación al contexto en el que actúa.
- Capacidad de comunicación con los agentes Proveedores para recoger datos de sensores y mensajes de otros agentes para mejorar los planes de actividades.

Un ejemplo sencillo de funcionamiento del agente Intérprete aplicado a las condiciones de temperatura deseadas por un usuario, se produce cuando el sistema detecta su presencia en un contexto determinado a través de los sensores y el chip RFID que identifica al usuario. El sistema tiene guardadas las preferencias de su temperatura deseada en el contexto. Por otro lado el sistema también tiene guardada la base de casos para el usuario en otras épocas o tiempos similares y además también obtiene la temperatura ambiental del exterior. Con las entradas mencionadas el agente Intérprete basado en CBR genera planes dinámicos de acciones de estabilización de la temperatura en el entorno, mientras se detecte la presencia del usuario. A su vez, el agente Intérprete también es responsable de la planificación de las tareas del usuario. Así, por ejemplo, planifica su jornada en función de las acciones a realizar, planifica actividades como la realización de ejercicio físico diario y otras actividades que puede llevar a cabo de modo concurrente el usuario.

Estos planes de acciones se desarrollan y controlan a través del método CPM. El agente Intérprete envía los planes de acciones al agente Proveedor para que éste actúe sobre el dispositivo de control correspondiente. De esta manera el sistema mantiene, por ejemplo, la temperatura deseada por el usuario en un contexto determinado. Para conseguir los resultados esperados por el usuario, el agente Intérprete se basa en el método de razonamiento CBR utilizando como estrategia de planificación el método CPM. En el desarrollo de planes de tareas óptimos el agente Intérprete recibe una serie de información del sistema HoCCAC a través de los agentes Proveedores y sigue

una serie de normas del método CPM vistas en el capítulo 3 de este trabajo de investigación.

4.6 Diseño del Agente Intérprete

Como se ha descrito en el capítulo 3 de este trabajo de investigación, CBR es un paradigma que se basa en la idea de que los problemas similares tienen soluciones similares. Los agentes que se diseñan e implementan utilizando sistemas CBR pueden razonar de forma autónoma y además adaptarse a cambios del entorno [Fraile *et al.,* 2010c] [Corchado *et al.,* 2011]. De esta manera se cumplen dos de las características más importantes del agente Intérprete que son citadas en el punto anterior:

- Capacidad de razonamiento.
- Facilidad de re-planificación para adaptarse al contexto.

La otra característica importante es la capacidad de comunicación con los agentes Proveedores para recoger información del entorno a través de sensores y otros agentes. Por tanto para el diseño e implementación del agente Intérprete es necesario tener en cuenta los intercambios de información entre los agentes del sistema. La especificación FIPA[4] puede ser considerada hoy en día como un estándar válido para la comunicación entre agentes. Para el diseño del agente Intérprete se usa la metodología SysML que se describe en el apéndice B de esta memoria de tesis. SysML proporciona mecanismos para conseguir un diseño lo suficientemente detallado como para que la fase de implementación sea muy sencilla.

Por otro lado el modelo BDI es una base sólida para la modelización y la aplicación del comportamiento interno de agentes. El modelo BDI permite ver un agente como una entidad que busca un objetivo y que actúa de una manera racional. Se pueden relacionar sistemas CBR y agentes BDI si se implementan los casos como creencias, intenciones y deseos que conduzcan a la resolución del problema. En un agente CBR-BDI, cada estado equivale a una creencia. Además el objetivo a alcanzar también puede ser una creencia. Las intenciones van a ser planes que contienen un conjunto ordenado de acciones que el agente debe efectuar para alcanzar sus objetivos. El paso

[4] www.fipa.org

de un estado a otro se produce después de ejecutar una acción o tarea. Un deseo es uno de los estados finales alcanzados en el pasado. Para tener una transición correcta entre la fase de diseño y la fase de implementación el paradigma CBR-BDI tiene que ser soportado en la fase de implementación.

La plataforma *Java Agent DEvelopment Framework* (JADE) [Zhiming Zhao *et al.,* 2007] es una buena opción para desarrollar aplicaciones basadas en agentes. Además los agentes JADE siguen el modelo BDI y la plataforma JADE se puede utilizar para implementar agentes CBR-BDI. JADE también cumple con el estándar FIPA para operar con sistemas multiagente inteligentes. La plataforma JADE se centra en la aplicación del modelo de referencia FIPA, proporcionando la infraestructura necesaria de comunicaciones y una plataforma de servicios como por ejemplo un agente de gestión, y un conjunto de herramientas de desarrollo y depuración de los agentes CBR-BDI. JADE eXtension (*Jadex*) [Pokahr *et al.,* 2003] es una implementación de una arquitectura de agentes híbrida (reactiva y deliberativa) para representar estados de agentes JADE que siguen el modelo BDI. *Jadex* está diseñado para ser fácilmente integrado en JADE, añadiendo un paquete. El principal objetivo es facilitar la utilización de conceptos de razonamiento y planificación en la implementación.

Visto desde fuera el agente Intérprete es una caja negra que recibe y envía mensajes. A continuación se va a tratar de profundizar más en el funcionamiento del agente Intérprete, a través del diseño. El agente Intérprete, descrito anteriormente, se va a implementar como un agente Jadex. Para ello se introducen algunas variaciones en la arquitectura Jadex, como se puede ver en la Figura 4.13.

La Figura 4.13 muestra un resumen de la arquitectura del agente Intérprete. Los mensajes entrantes, así como los eventos internos y los nuevos objetivos, sirven como entrada para las reacciones internas y los mecanismos de deliberación y razonamiento del agente Intérprete. La principal novedad en el diseño de este agente, como se puede ver en la Figura 4.13, es que se integra un motor de razonamiento CBR y un sistema reactivo que recoge datos de sensores y de los sistemas de control. Esto hace a este diseño único en cuanto a su concepción y a sus capacidades de razonamiento.

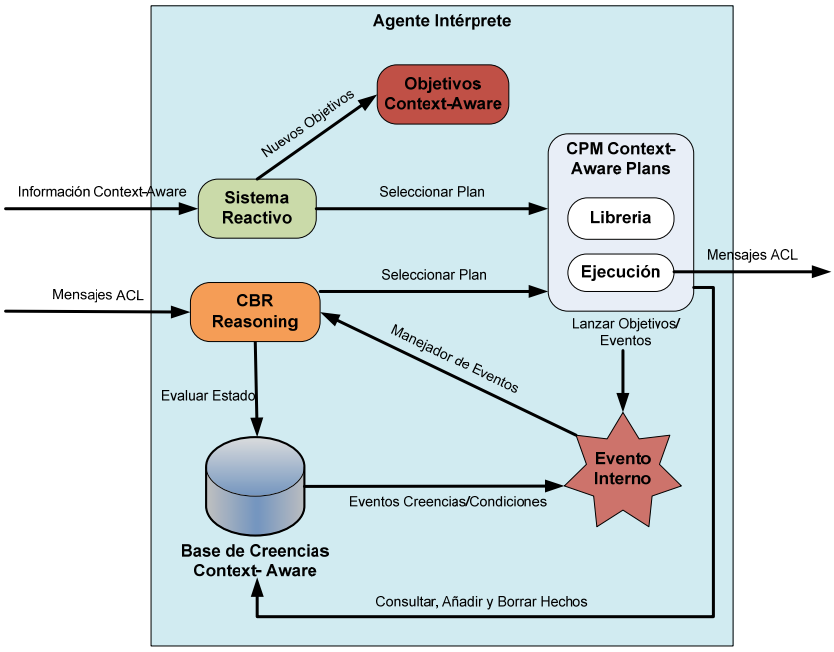


Figura 4.13 Descripción general de la arquitectura del agente Intérprete.

La estructura del motor de razonamiento CBR ha sido diseñada alrededor del concepto de caso. La Figura 4.14 muestra los componentes de un caso: el problema define la situación del contexto en un momento dado, la solución es el conjunto de estados que componen el contexto como una consecuencia de acciones que han sido llevadas a cabo dentro del contexto y el resultado muestra la situación del contexto una vez que el problema ha sido solucionado.

Caso: Problema, Solución, Resultado
Problema: estado_inicial
Solucion: <acción,[estado_intermedio]>*

<>: Secuencia
[]: Opcional
*: 0 o más repeticiones

Figura 4.14 Definición de un caso en el sistema CBR.

La Figura 4.15 define cómo cada estado se considera una Creencia; una creencia puede también ser el objetivo a alcanzar. Las intenciones son los planes de acción que el agente elige para llevar a cabo con el fin de lograr sus objetivos, por lo que

una intención es un conjunto ordenado de acciones; cada cambio de estado se realiza después de realizar una acción (el agente recuerda la acción llevada a cabo en el pasado cuando era un estado específico y su resultado). Un deseo será cualquiera de los estados finales alcanzados en el pasado (si el agente tiene que tratar una situación, que es similar a un caso del pasado, trata de generar un resultado similar al obtenido en el caso del pasado).

Creencia: estado, **Deseo**: <estado_final>*, **Intención**: <acción>*

Figura 4.15 Definición de las actitudes mentales de una agente BDI.

La relación directa entre agentes BDI y sistemas CBR puede ser identificada comparando las Figuras 4.14 y 4.15. Usando esta relación se pueden implementar agentes (a nivel conceptual) usando un modelo CBR (a nivel de implementación). La ventaja de este enfoque es que un problema puede ser fácilmente formalizado en términos de agentes y luego implementado como un sistema CBR.

El agente CBR-BDI se denota por un conjunto de siete elementos <E, GAL, CM, PAL, EK, O, M>. Esta notación puede también ser usada para definir las creencias, deseos e intenciones de los agentes, debido a su correspondencia con los elementos que componen un sistema CBR [Bajo *et al.,* 2006b]. Los componentes de un sistema CBR en un contexto son los siguientes:

E: Entorno o contexto E descrito por un conjunto de valores para un conjunto fijo no vacío de las variables de entorno, denominado $E = <e_1, e_2, ...,e_n>$ donde $e_i \in E$, $i \in \{1,2,...,n\}$. Por lo tanto E describirá el contexto actual. Cada variable e_i es un conjunto de tres elementos $<nombre\_var_i, valor_i, indice_i>$ donde $nombre\_var_i$ es el nombre de la variable, $valor_i$ es un conjunto de valores para cada variable e $indice_i$ es un campo *boolean*, *1* si la variable es un índice y *0* si es otra cosa.

GAL: Librería de Acciones General $GAL = <ga_1, ga_2, ..., ga_p>$ donde $ga_i \in GAL$, $i \in \{1,2,...,p\}$. Cada ga_i es descrito como un conjunto de tres elementos $<Nombre_i, InPar_i, OuPar_i>$, donde $Nombre_i$ es el nombre de la acción, $InPar_i$ es un conjunto de parámetros de entrada $InPar_i \subseteq E$ y $OuPar_i$ es un conjunto de parámetros de salida $OuPar_i \subseteq E$.

CM: Memoria de Casos CM, que almacena un conjunto de casos previos CM = $<c_1, c_2, ...,c_k>$, donde cada caso c_i está formulado como un conjunto de tres elementos $<B, D, I>$ representando creencias pasadas, deseos pasados e intenciones pasadas.

- *B*: Creencias del agente son los valores exactos para cada variable del contexto en un instante de tiempo. $B = <b_1,b_2,...,b_m>$ donde cada creencia es un estado, el cual es un conjunto de dos elementos $b_i = <E, valueE>$. $E = <e_1, e_2, ...,e_n>$ donde e_i son las variables que describen el contexto y $valueE = <v_1,v_2,...,v_n>$ donde $v_i \in value_i$, $i \in \{1,2,...,n\}$, es el valor exacto para cada variable en un instante concreto.
- *D*: Los deseos *D* corresponden a un conjunto de objetivos que determinan la resolución del problema. $D = <d_1, d_2, ..., d_l>$, donde $d_i \in B$, $D \subset B$.
- *I*: Las intenciones *I* representan un *trigger* para el correspondiente plan de la Librería de Planes del Productor PAL. Son acciones previas y tienen los valores de los parámetros de entrada y los parámetros de salida.

PAL: Librería de Acciones del Productor que es una colección de acciones. Cada intención es una secuencia ordenada, $I = <a_1, a_2, ..., a_o>$ donde cada cambio de estado se produce después de llevar a cabo una acción. Cada $a_i \in$ PAL, $i \in \{1,2,...,o\}$ es un conjunto de tres elementos $<Nombre_i, ValorInputPar_i, ValorOutputPar_i>$ donde $Nombre_i \in ga_i$, ga_i es una acción general de GAL, $ValorInputPar_i$ y $ValorOutputPar_i$ son los respectivos conjuntos de valores exactos para parámetros de entrada y parámetros de salida de ga_i. $ValorInputPar_i = <v_{i1},v_{i2},...,v_{in}>$ y $ValorOutputPar_i = <v_{i1},v_{i2},...,v_{in}>$ donde $v_{ij} \in value_i$, $i,j \in \{1,2,...,n\}$.

EK: Conocimiento Experto EK está compuesto de un conjunto de reglas por defecto asociadas con la adaptación de casos y el proceso de retención de un caso. La adquisición de estas reglas es automatizada. $EK = <r_1,r_2,...,r_v>$, donde r_i es conjunto de dos elementos (*Previous*, *Consistent*). $Previous = < v_1,v_2,...,v_s>$, donde $v_i \in value_i$, $i \in \{1,2,...,s\}$, es el valor exacto para algunas variables. $Consistent = <v_{s+1},...,v_n>$, donde $v_i \in value_i$, $i \in \{s+1, s+2,...,n\}$, es el valor exacto para otras variables.

O: Esto se considera como un conjunto de objetivos actuales (O) en un particular *belief-world*. Los objetivos son estados finales apropiados en el contexto. $O = <o_1, o_2,$

..., o_t>, donde $o_i \in B$ y $i \in \{1,2,...,t\}$, $O \subset B$. Este conjunto es nulo si el contexto no está definido.

M: Conjunto de funciones de similitud. Una función de similitud determina el grado de igualdad entre dos estados. M = <m_1, m_2, ..., m_j>, donde $m_i \in M$ y $i \in \{1,2,...,j\}$.

Cuando el sistema CBR recibe un problema nuevo b_{new} genera una secuencia de estados intermedios <(b_{new},a_1), (b_2,a_2), ..., (b_n,a_n)> antes de buscar el estado final b_n.

Las cuatro fases del ciclo de vida de un sistema CBR se formalizan a continuación. Una métrica o técnica diferente debe ser identificada para cada uno de los pasos del CBR dependiendo del problema que trate el agente y del contexto en el que se desarrolla el problema.

- **Recuperar:** Los casos similares del problema nuevo b_{new}, (c_1,c_2,...,c_k) son recuperados del caso base CM usando una métrica de similitud m_1(Coseno, Seno,...). Los casos recuperados son c_1,c_2,...,c_k.
- **Reusar:** Una primera solución <(b_{new},a_1), (b_2,a_2), ..., (b_n,a_n)> se obtiene a partir de los casos recuperados y el problema b_{new}. Esta solución inicial es un plan, una secuencia ordenada de estados y acciones. Un experto puede adaptar esta solución ajustando los parámetros para cada acción.
- **Revisar:** En esta fase se evalua el plan <(b_{new},a_1), (b_2,a_2), ..., (b_n,a_n)> obtenido en la fase previa. Se debe chequear si el estado final b_n y el plan desarrollado para lograrlo son adecuados. La revisión puede ser realizada por un experto.
- **Retener:** El plan nuevo <(b_{new},a_1), (b_2,a_2), ..., (b_n,a_n)> se indexa y almacena en el correspondiente caso base CM.

Usando esta formalización se necesita identificar un conjunto de creencias, deseos e intenciones iniciales e incluirlos en el caso base del agente como casos. Luego hay que definir un número de métricas para los pasos de recuperar, reusar, revisar y retener.

Basándose en la formalización descrita y los resultados del motor de razonamiento CBR el agente Intérprete desarrolla planes de tareas a través del *CPM Context-Aware Plans* teniendo en cuenta las normas y procedimientos descritos en el capítulo 3 de este trabajo de investigación, sobre el método de planificación CPM. Estos planes

pueden ser ejecutados inmediatamente como eventos o también pueden generarse nuevos planes para ser ejecutados en el futuro y ser almacenados en la librería de planes *context-aware*. La ejecución de planes puede modificar la base de creencias *context-aware*, enviar mensajes a otros agentes, crear nuevos objetivos *context-aware* o causar eventos internos.

El agente Intérprete tiene una base de creencias *context-aware* como muestra la Figura 4.13 en la que almacena los hechos que constituyen su base de conocimiento. Estas creencias están relacionadas con el entorno *context-aware* y con el usuario. Son creencias como la localización del usuario, la temperatura del exterior, del hogar y habitaciones o la luminosidad y el nivel de humos en las habitaciones del hogar. Los hechos son estructurados a través de objetos Java que representan las creencias según se representa en la Figura 4.11. Estos objetos tienen un nombre y atributos que pueden tener valores simples o valores múltiples. La base de creencias también incorpora el concepto de las bases de datos orientadas a objetos. OCL permite recuperar subconjuntos de creencias *context-aware*. Otra característica especial de la base de creencias *context-aware* son las condiciones. Las condiciones representan una expresión de un cierto estado, por ejemplo, de una o de varias creencias. Una vez que la condición se cumple, se genera un evento interno, este evento puede activar un plan, o dar lugar a la adopción de nuevos objetivos. En el agente Intérprete las creencias representan los cambios de estado de los sensores instalados en el entorno *context-aware,* es decir la información contextual. Además las creencias también representan la localización del usuario, las preferencias y gustos del usuario, sus dolencias o problemas físicos, sus alergias y dietas alimenticias. Esto permite que sea sencillo añadir nuevos tipos de sensores que ayuden en las tareas diarias al usuario y que los planes de tareas añadan nuevos estados para un sensor en el futuro. Además el agente Intérprete a través del lenguaje OCL puede consultar las creencias que cumplan una determinada condición y así simplificar la elaboración de planes de tareas. Todos los planes de tareas y acciones específicas del agente Intérprete conducen a conseguir el objetivo final. El agente Intérprete puede definir tres tipos de objetivos de acuerdo con el modelo de implementación *Jadex*:

- Lograr un objetivo. Para lograr un objetivo antes hay que definir un estado al que se quiere llegar pero no es necesario especificar la forma de alcanzarlo. En este caso el agente Intérprete puede lograr el objetivo con distintas alternativas o planes de tareas. Para ello, el agente Intérprete recomienda el plan de tareas más óptimo y da la posibilidad al usuario de elegir un plan.
- Mantener un objetivo. Para mantener un objetivo el agente Intérprete realiza un seguimiento del estado y ejecuta los planes para restablecer el estado cuando sea necesario.
- Realizar un objetivo. Para realizar un objetivo el agente Intérprete desarrolla planes de tareas que especifican las acciones a ejecutar para lograr el objetivo de la forma más óptima posible.

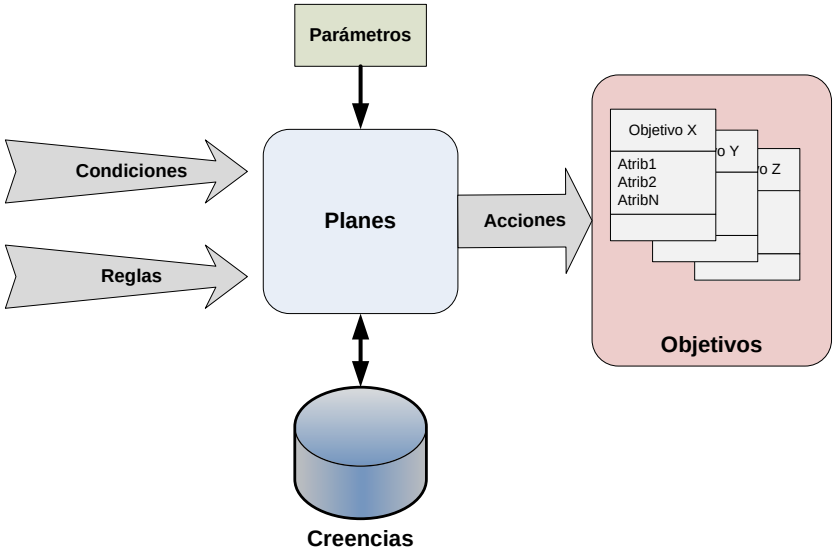


Figura 4.16 Esquema que representa los objetivos y la relación con los componentes necesarios para alcanzarlos.

La Figura 4.16 muestra un esquema con la representación de los objetivos y la forma de alcanzarlos a partir de planes, creencias, parámetros, condiciones, reglas y acciones. Puede verse que los objetivos en el agente Intérprete se representan mediante objetos con varios atributos. Los planes para conseguir el objetivo se

especifican de forma explícita a través de condiciones, utilizando las creencias. El nombre y las propiedades del objetivo facilitan la selección del plan a lanzar y los parámetros guían las tareas de los planes en ejecución.

La realización de un objetivo define directamente el plan que se ejecuta. Así pues, el agente Intérprete puede ser implementado de dos formas:

- Mediante la activación y desactivación de condiciones, el agente Intérprete aplica reglas según el enfoque deseado para activar o desactivar objetivos cuando se cumplen ciertas condiciones internas, por ejemplo condiciones definidas sobre la base de creencias en las preferencias del usuario.
- Mediante la activación y desactivación de objetivos manualmente a través de planes de tareas. El agente Intérprete mantiene un objetivo, por ejemplo al mantener la temperatura deseada en una habitación, desarrolla o realiza un objetivo por ejemplo cuando genera planes de tareas rutinarias para un usuario concreto, y logra un objetivo, cuando por ejemplo consigue activar el extractor de gases ante la activación de la alarma de humos.

El agente Intérprete posee una librería de planes *context-aware* basados en el método CPM. Con la librería de planes genera soluciones basadas en planes de acciones que han sido lanzados con anterioridad. Estos planes de acciones actúan sobre la rutina diaria del usuario y tratan de hacerle la estancia más confortable en el entorno *context-aware*. Una parte de la implementación del agente Intérprete descompone su funcionalidad en planes separados compuestos de acciones que son implementadas en clases Java. Por tanto, se puede decir que las técnicas de orientación a objetos son utilizadas también en el desarrollo de los planes. Además la funcionalidad implementada en clases Java se puede incorporar en otros sistemas similares o heredados. Así pues, a continuación se propone un diseño SysML a bajo nivel para el agente Intérprete para posteriormente proceder a la implementación con Jadex. Con el diseño SysML se obtiene un diagrama de clases para el agente Intérprete que se muestra en la Figura 4.17. Se trata del agente más importante dentro de la arquitectura HoCCAC.

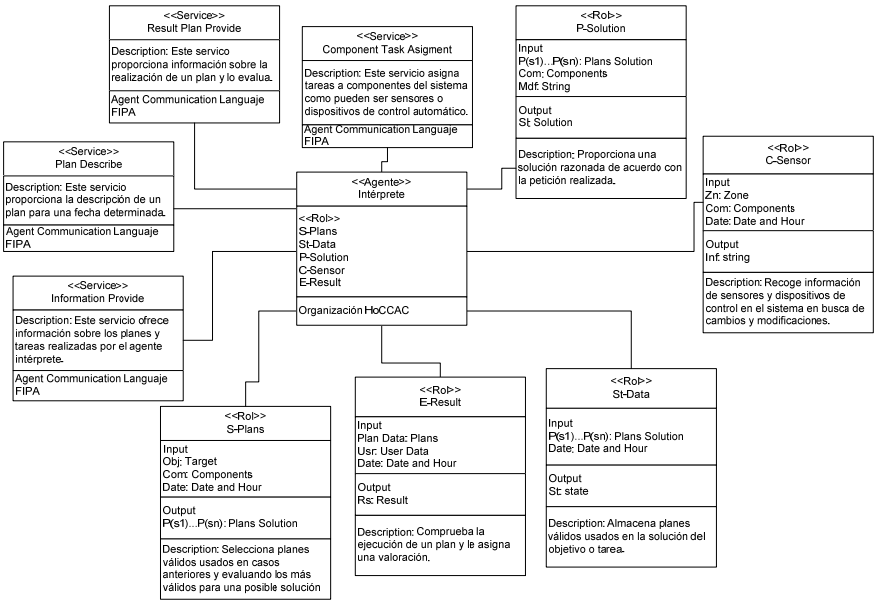


Figura 4.17 Diagrama de definición de bloques SysML para el agente Intérprete.

Este agente como se puede ver en la Figura 4.17, tiene cinco roles o capacidades y cuatro servicios. Los roles son:

- *C-Sensor*, desarrolla la funcionalidad correspondiente a la etapa de recuperación de casos. En esta capacidad se utiliza un conjunto de creencias (cada una de ellas es una descripción de un problema). Cada creencia necesita ser importada de las creencias principales del agente. De esta forma, en esta capacidad es posible acceder a las descripciones de problemas pasados que posee el agente, recuperando dichas descripciones, o bien introduciendo nuevas descripciones de problema. Además, la capacidad *C-Sensor* define una serie de metas necesarias para alcanzar los deseos de recuperación o actualización de creencias e intenciones.
- *E-Result*, es una capacidad que se encarga de realizar la etapa de reutilización de casos. Para ello dispone de una creencia. Se trata de una base de creencias en la que se almacenen las soluciones obtenidas para las distintas descripciones de problema resueltas en el pasado. El concepto de solución en el razonamiento basado en casos no solamente implica los resultados obtenidos tras aplicar la

solución, sino también la forma en la que se construye la solución. Por ello, será necesario anotar la secuencia de planes ejecutados que permitieron alcanzar la solución, así como el estado de las metas una vez obtenida la solución.

- *P-Solution* es la capacidad encargada de llevar a cabo la etapa de revisión, aunque puede considerarse que también realiza parte de la etapa de reutilización si en la capacidad *E-Result* se han propuesto varias soluciones al problema, o si es necesario realizar una re-planificación dinámica en tiempo de ejecución. Existen varias posibilidades para evaluar la solución propuesta. En muchas ocasiones es un experto externo quien la evalúa aunque la solución obtenida se evalúa mediante un sistema experto de conocimiento. En cualquier caso, la capacidad *P-Solution* posee creencias en las que se almacena el resultado de la revisión de cada solución.
- *S-Plans* en esta capacidad se utiliza un conjunto de intenciones. Cada intención es un plan que está compuesto de una serie de actividades. Cuando la capacidad detecta la activación de una meta u objetivo (a través de algún evento interno o de la llegada de algún mensaje), dispara la ejecución de planes que permitan alcanzar las metas activadas. El mecanismo de selección de planes para cada meta marcada permite elegir el plan más adecuado, o bien ejecutar varios planes al mismo tiempo que intentarán alcanzar dicha meta. La meta cambia de estado en función de la respuesta obtenida a través de los planes.
- *St-Data*, esta capacidad desarrolla la funcionalidad correspondiente a la etapa de retención o aprendizaje del conocimiento. La capacidad empaqueta creencias, metas y planes que el propio agente Intérprete a desarrollado.

Los servicios son:

- *Information Provide*, que ofrece información sobre los planes y tareas realizadas por el agente Intérprete.
- *Plan Describe*, que proporciona la descripción de un plan para una fecha determinada.
- *Result Plan Provide*, que proporciona información sobre la realización de un plan y lo evalúa.

- *Component Task Asigment*, que asigna tareas a componentes del sistema como pueden ser sensores, actuadores o dispositivos de control automático.

Interpreter Agent

```
#start option
name                    = InterpreterAgent
class                   = jadexBDIAgent
#define plans
Instant_plans           = openS, closeS
plans                   = tskplan, tskadd, clean
#define plan mappings
openS                   = OpenSensorRoom()
closeS                  = CloseSensorRoom()
tskplan                 = TaskPlan()
tskadd                  = TaskAdd()
clean                   = CleanTaskPlan()
#define activation filters for passive plans
tskplan_filter          = TaskPlan.getEventFilter()
tskadd_filter           = TaskAdd.getEventFilter()
#define initial beliefs and beliefsets
beliefsets              = tskactionsensors
tskaction               = {(idsensor, action, type)}
#define goals
goals                   = hotup, colddown, levelSmoke
#define goal mappings
hotup                   = Goal.createMantainGoal(hotup, temperature==23, hot)
colddown                = Goal.createMantainGoal(colddown, temperature==21, cold)
levelSmoke              = Goal.createPerformGoal(levelSmoke, ratioSmoke==5, low)
```

Figura 4.18 Ejemplo de definición del agente Intérprete.

La ejecución del agente Intérprete a partir del modelo Jadex está basada en eventos. Todo lo que sucede dentro del agente Intérprete se representa como un evento. Los 'eventos mensaje' indican la recepción de un mensaje ACL. Los 'eventos objetivo' anuncian la consecución de un objetivo y los eventos internos informan por ejemplo de los cambios de creencias*, time-outs*, o condiciones que se han cumplido. Para crear y comenzar un agente Intérprete el sistema necesita saber las propiedades del agente Intérprete. El estado del agente Intérprete, viene determinado por las creencias, los objetivos, los planes de ejecución y la librería de planes. Todos estos conceptos (creencias, objetivos, planes, condiciones, filtros) están definidos en el *Agent Definition File* (ADF). El ADF define las creencias y objetivos iniciales a través de la declaración de objetos Java. Los planes se declaran especificando la forma de crear una instancia de la clase Java. El formato del fichero ADF es un fichero de propiedades con pares nombre/valor que mapea referencias a declaraciones de

objetos. En la Figura 4.18 se puede ver un recorte de la definición del agente Intérprete.

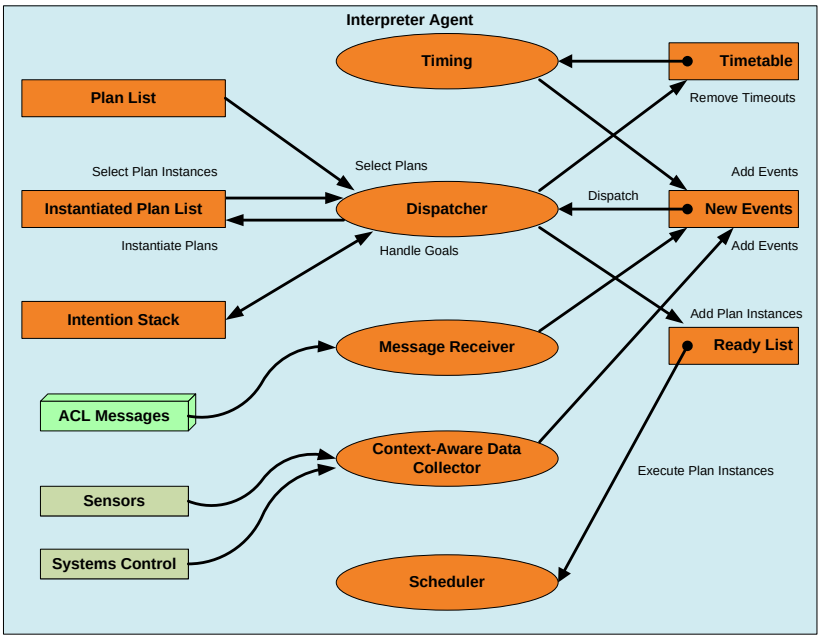


Figura 4.19 Modelo de ejecución del agente Intérprete.

A continuación se pasa a describir el modelo de ejecución que sigue el agente Intérprete. Los comportamientos o capacidades se ejecutan dentro del agente Intérprete que está basado en Jadex y son: el *context-aware data collector*, el *message receiver,* el *timing*, el *dispatcher* y el *scheduler*. Estos comportamientos, representados en la Figura 4.19, se ejecutan concurrentemente en el interior de la estructura del agente. El buen funcionamiento y la disponibilidad del *context-aware data collector* es básico para el desarrollo de tareas del resto de comportamientos del agente Intérprete. El *context-aware data collector* se encarga de recoger los datos de sensores y sistemas de control y genera los correspondientes avisos y eventos derivados de la información recogida. El *message receiver* y el proceso de *timing* tienen un comportamiento muy simple, sólo añaden nuevos eventos a la lista de eventos. El *message receiver* escucha los mensajes ACL enviados por otros agentes y genera los correspondientes 'eventos mensaje'. El *timing* elimina los eventos

planificados cuando ya han sido lanzados y añade nuevos eventos a la lista para ser lanzados. El *dispatcher* es el responsable de la adopción de objetivos para ponerlos en la cola de ejecución y seleccionar los planes que gestionan los eventos necesarios para cumplir el objetivo. Los planes seleccionados son ejecutados paso a paso siguiendo el método de planificación CPM por el *scheduler* que también gestiona la supervisión del plan. Estos comportamientos se ponen en reposo ellos mismos y se reinician entre sí para evitar que el agente consuma recursos innecesariamente. Implementar las funcionalidades del agente Intérprete en comportamientos separados proporciona un diseño limpio y permite la sustitución flexible de los comportamientos con implementaciones personalizadas como por ejemplo mecanismos de planificación y razonamiento alternativos.

4.7 Conclusiones

La adaptación automática de la tecnología a los usuarios sobre contextos variados en la sociedad de la información requiere soluciones efectivas. Una de estas soluciones consiste en la utilización de agentes y sistemas multiagente. Sin embargo, las arquitecturas de sistemas multiagente resultan ser muy generales y, en muchos casos, no resultan adecuadas para resolver algunos de los problemas concretos que plantean los sistemas sensibles al contexto. En este capítulo se ha presentado una arquitectura multiagente capaz de facilitar la construcción de entornos sensibles al contexto. La arquitectura presentada se centra en los principales aspectos de la computación sensible al contexto, de tal forma que propone la utilización de sistemas multiagente que se sitúan de forma transparente y no intrusiva como una etapa intermedia entre el usuario y el resto del entorno. La arquitectura HoCCAC permite el uso de diferentes tipos de agentes, cada uno de ellos orientado a resolver alguna de las principales preocupaciones de la computación sensible al contexto.

Las principales aportaciones de la arquitectura multiagente HoCCAC desde el punto de vista de los agentes son:

- Proporciona interfaces inteligentes a través del agente Interface y los agentes Proveedores, para interaccionar con los usuarios y con los dispositivos activos del entorno y adaptar el funcionamiento de todos estos dispositivos de manera

no intrusiva al usuario. Así, el contexto está formado por un conjunto de sensores, actuadores y agentes que facilitan las tareas rutinarias del usuario y el usuario actúa sin notar la presencia de este conjunto de dispositivos.

- Control y supervisión de los usuarios en su contexto. En este trabajo se propone un modelo de interacción con los usuarios y el entorno que se produce a través de un conjunto de agentes inteligentes, tecnología por radio frecuencia, redes inalámbricas y dispositivos móviles inalámbricos. Este conjunto de componentes y tecnologías trabajan de forma conjunta y en tiempo de ejecución, proporcionando información sobre el contexto. Así se permite, entre otras cosas, un control y supervisión preciso de los usuarios al informar en cada momento la localización exacta de cada uno de ellos dentro del entorno y aportar características propias del entorno en el que se maneja el usuario.
- Planificación de actividades rutinarias para los usuarios en tiempo de ejecución. En los sistemas actuales, el usuario debe programar a priori las tareas o actividades que debe desempeñar o recordar el sistema. Sin embargo, gracias a las capacidades de los agentes dentro de la arquitectura propuesta, se puede aprender de los hábitos de los usuarios y, sobre este conocimiento, planificar las actividades de los usuarios de forma dinámica y en tiempo de ejecución [Fraile *et al.*, 2008b] [Carrascosa *et al.*, 2008], sin importar la localización física de los recursos, gracias a la distribución de los componentes software y hardware que integra la arquitectura.

La arquitectura HoCCAC proporciona beneficios para los usuarios como la generación automática de planes de actividades rutinarias o respuestas e informes personalizados para los usuarios del entorno. En este trabajo se ha presentado el diseño del agente Intérprete que es el corazón de la arquitectura HoCCAC. Se trata de un agente deliberativo, basado en el modelo BDI, que en su diseño es único, ya que integra un sistema de planificación CPM y un sistema de razonamiento CBR. El agente Intérprete utiliza una base de planes ejecutados en el pasado para diseñar y re-diseñar estrategias variadas si sus planes en curso son interrumpidos. De esta forma, el agente puede re-planificar en tiempo de ejecución, evitando la vuelta atrás. Este hecho hace que el agente Intérprete posea grandes capacidades de aprendizaje y de

adaptación, y que resulte adecuado para ser utilizado en entornos rutinarios y dinámicos.

La computación sensible al contexto es una parte fundamental en el diseño de la arquitectura HoCCAC, integrando dispositivos y herramientas capaces de obtener y procesar información sobre el contexto automáticamente y en tiempo de ejecución y que, a través de una colaboración distribuida entre sus elementos, proporcionan la flexibilidad para adaptarse a las necesidades, tanto del contexto como de los usuarios, de manera natural, ubicua y transparente. Uno de los principales problemas que plantean los sistemas sensibles al contexto es el número variable de aplicaciones contextuales destinadas a ofrecer diferentes prestaciones y el hecho de que cada una de las aplicaciones contextuales gestione la adquisición de los parámetros de contexto y los recursos independientemente de las demás. Esto es realmente poco eficiente y sostenible en tiempo de ejecución, además de presentar evidentes problemas de escalabilidad y de incorporación de información de nuevos sensores. Otro problema es la baja calidad de los servicios, en muchas ocasiones propiciada por el uso de una única tecnología de sensorización que no es suficiente para caracterizar las situaciones. También es un problema la falta de adecuación al contexto y sistemas de razonamiento y planificación que no incluyen aspectos personales del usuario. Otras veces los sistemas sensibles al contexto tienen interfaces estáticas y poco naturales, que no se ajustan suficientemente a la situación cognitiva de las personas. Finalmente, la preocupación por la falta de privacidad que puede suponer para el usuario la implantación de un sistema sensible al contexto también llega a ser un problema.

La arquitectura HoCCAC integra sensores y dispositivos de contexto variados para caracterizar en detalle la situación contextual de los usuarios en su contexto. Las principales aportaciones de la arquitectura multiagente HoCCAC desde el punto de vista del contexto son:

- Comunicación distribuida de todos los componentes y agentes integrados en HoCCAC. Los agentes son capaces de comportarse de forma autónoma y tienen la capacidad de residir directamente en dispositivos inalámbricos, pero es necesario que se comuniquen entre ellos para la toma de decisiones y la generación de planes, a través de una arquitectura distribuida. Una arquitectura

distribuida presenta una mejor capacidad de recuperación ante errores, permitiendo inicializar agentes por separado, sin necesidad de reiniciar todo el sistema [Fraile *et al.*, 2008b] [Carrascosa *et al.*, 2008].

- Soporte para la comunicación y combinación de redes inalámbricas y elementos domóticos. Las tecnologías de información y comunicación y la domótica están integradas en prácticamente todos los ámbitos de nuestras vidas, facilitando las tareas diarias y mejorando la calidad de vida. La arquitectura HoCCAC, basada en la utilización de agentes y sistemas multiagente permite una integración de elementos software y hardware que facilitan la construcción de aplicaciones sensibles al contexto. Las redes inalámbricas 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 inalámbricas además permiten que los programas, datos y equipos estén disponibles para cualquier agente de la red que lo requiera, sin importar la localización física del recurso y del usuario.
- Soporte para tecnologías de identificación inalámbricas de los elementos contextuales. Por su parte, la tecnología de identificación por radio frecuencia (RFID) proporciona la personalización de componentes en el contexto a través de un identificador único que permite a los agentes identificar y localizar usuarios o elementos del contexto de forma automática, por ejemplo a través de dispositivos móviles con tecnología NFC, disminuyendo posibles incidencias surgidas por la incorrecta identificación de las personas.

La arquitectura HoCCAC proporciona otros beneficios como la combinación del estado contextual con distintas formas de interacción con el entorno para ofrecer nuevas posibilidades. La disponibilidad de tecnologías como NFC en el móvil hace posible lograr una mayor automatización en el lanzamiento de servicios. Dicha tecnología puede tener un interés especial para aplicaciones como el prototipo presentado en el caso de estudio, en el que la facilidad de interacción con el entorno y la facilidad de aprendizaje del uso de la tecnología representa un valor añadido al sistema. Además HoCCAC facilita la incorporación de dispositivos que recogen

atributos de contexto variados. Esto permite una caracterización detallada del entorno contextual del usuario y una forma más eficiente de poder ofrecer servicios personalizados a los usuarios.

En el siguiente capítulo se evalúa la validez de la arquitectura HoCCAC. Para ello se aplica a un caso de estudio para el desarrollo de un entorno inteligente y se presentan los resultados experimentales obtenidos. Con el caso de estudio que se plantea a continuación se pretende mejorar diversos aspectos relacionados con los cuidados médicos y la asistencia de pacientes con Enfermedad Pulmonar Obstructiva Crónica (EPOC) dentro de su hogar.

5

Caso de Estudio

En este capítulo se describe el caso de estudio que permite evaluar la arquitectura multiagente HoCCAC, propuesta en el capítulo 4 de este trabajo de investigación. El caso de estudio consiste en el desarrollo de un sistema multiagente orientado a mejorar los servicios de asistencia y cuidados médicos en el hogar a personas dependientes, especialmente con la Enfermedad Pulmonar Obstructiva Crónica (EPOC). El caso de estudio se ha desarrollado mediante la implantación de la arquitectura HoCCAC en un entorno simulado. En este sentido, el caso de estudio está compuesto por un conjunto de agentes inteligentes con capacidades razonamiento, cooperación, planificación y toma de decisiones de forma dinámica en tiempo de ejecución. El caso de estudio elegido, permite evaluar las propuestas presentadas en el marco de este trabajo de investigación, por su gran capacidad para la incorporación de elementos inteligentes, la disponibilidad de dispositivos inalámbricos y la pujante necesidad de personalización que requieren hoy en día las personas dependientes en sus hogares. De esta manera, el modelo obtenido, puede ser fácilmente reutilizado en entornos similares, aplicando simplemente las modificaciones necesarias para cubrir

las características inherentes de cada paciente y el contexto en el que realiza las actividades diarias.

La infraestructura tecnológica del caso de estudio hace uso de las tecnologías WiFi, RFID, NFC y ZigBee descritas en detalle en el Apéndice A de este trabajo de tesis. Estas tecnologías han sido aplicadas con éxito en diversos desarrollos relacionados con entornos dependientes en el grupo de investigación BISITE, grupo que cuenta con una amplia experiencia en el uso de este tipo de tecnologías inalámbricas [Corchado *et al.,* 2008a] [Bajo *et al.,* 2007] [Bajo *et al.,* 2010] [Tapia *et al.,* 2011] [Fraile *et al.,* 2009b] [Fraile *et al.,* 2010d].

El capítulo se estructura de la siguiente forma: en la sección 5.1 se presenta una introducción en la que se describe de forma detallada los objetivos del capítulo y la metodología seguida para alcanzarlos, en segundo lugar, en la sección 5.2 se detalla la problemática concreta del caso de estudio, obteniendo los casos de uso más comunes del paciente en su hogar. A continuación, en la sección 5.3, se muestra el proceso que ha permitido modelar el contexto del sistema, sentando las bases para modelar las características del caso de estudio. En la sección 5.4 se desarrolla el proceso de análisis y diseño del caso de estudio. Para ello se utiliza la metodología SysML descrita en el Apéndice B y la arquitectura HoCCAC, presentada en el capítulo 4 de este trabajo. En la sección 5.5 se presenta un ejemplo concreto de planificación dinámica de tareas de un paciente EPOC en su hogar para mejorar las condiciones de vida del paciente. Finalmente en la sección 5.6 se presentan las conclusiones y resultados obtenidos tras aplicar la arquitectura HoCCAC al caso de estudio planteado.

5.1 Introducción

Los agentes y los sistemas multiagente se están convirtiendo en una realidad en los entornos sensibles al contexto. Muchos de los desarrollos basados en agentes se centran en la monitorización de pacientes, supervisión de tratamientos y minería de datos clínicos [Fraile *et al.,* 2011. En el capítulo 3 (punto 3.2.1) de este trabajo de investigación se realiza un amplio estudio de sistemas multiagente relacionados con la computación sensible al contexto [Korpiää *et al.,* 2003] [Chen *et al.,* 2004] [Shehzad

et al., 2004a] [Yamabe *et al.,* 2005] [Fraile *et al.,* 2009b]. 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 en el hogar [Fraile *et al.,* 2009b]. Sin embargo, todos ellos se encuentran en fases tempranas de desarrollo o simplemente no han llegado a ser del todo implementados y algunos se centran en problemáticas muy concretas. Además, ninguno cubre todos los requerimientos de los sistemas basados en la computación sensible al contexto, desde el punto de vista de la interacción inteligente entre los componentes del contexto y los usuarios.

Figura 5.1 Enfermo EPOC y máquina de administración de oxígeno.

El caso de estudio que se presenta a lo largo de este capítulo proporciona una interacción inteligente entre los dispositivos del contexto y los usuarios. Para conseguirlo, se basa en la arquitectura HoCCAC y en la información recogida en el entorno para planificar las tareas de un paciente EPOC en su hogar y optimizar y mejorar las condiciones de vida del paciente. En la Figura 5.1 se muestra a un enfermo de EPOC sobre el que se ha aplicado el caso de estudio que se describe a continuación. Estos enfermos necesitan una administración artificial de oxígeno. En la

Figura 5.1 también se muestra la máquina que suministra oxígeno al enfermo del caso de estudio.

EPOC es una enfermedad pulmonar que se caracteriza por la existencia de una obstrucción progresiva de las vías aéreas. La causa principal de esta enfermedad es el humo del tabaco y produce como síntoma una disminución de la capacidad respiratoria, que avanza lentamente con el paso de los años y ocasiona un deterioro considerable en la calidad de vida de las personas afectadas. Los enfermos de EPOC son enfermos crónicos, que dependen mucho de las condiciones ambientales, pero hay que destacar, que un plan de cuidados realizado de forma adecuada puede mejorar de una forma considerable su calidad de vida. Este plan de cuidados incluye una medicación muy estricta y unas horas de administración de oxígeno diarias muy concretas que son marcadas por preinscripción médica, depende en gran medida del contexto donde se desarrolle debido por ejemplo a la temperatura y humedad ambiental. Este tipo de enfermos también necesita realizar ciertas tareas diarias como por ejemplo la toma de medicación y aerosoles o la administración de oxígeno a unas horas concretas, por tanto necesitan una planificación diaria de sus actividades. Por otro lado cualquier cambio en su rutina diaria influye sobre la planificación y es necesaria una re-planificación rápida y efectiva.

La arquitectura HoCCAC aborda toda la problemática de los enfermos EPOC y además le ofrece las soluciones más óptimas en tiempo de ejecución. Para ello HoCCAC caracteriza el contexto de usuario con la mayor cantidad de información contextual posible a través de sensores y dispositivos que localizan al usuario en su contexto, conocen la temperatura o humedad contextual, controlan escapes de agua o gas en el entorno, etc. En el hogar de un enfermo EPOC se pueden encontrar diferentes fuentes de información, que pueden variar significativamente para facilitar las tareas diarias de una persona dependiente. En este caso la arquitectura multiagente HoCCAC desarrollada en el marco de este trabajo permite aumentar la oferta de servicios en el hogar a las personas dependientes y mejorar la calidad de vida de las mismas. En particular el sistema multiagente:

- Facilita mecanismos que permiten a los usuarios acceder a la información a través de dispositivos móviles de manera sencilla y sin costes.

- Facilita al personal médico la comunicación con sus pacientes y proporciona una nueva forma de atención domiciliaria.
- Facilita la creación de planes de tareas diarias a pacientes en su contexto basándose en experiencias pasadas.
- Facilita la adquisición de información contextual a través de diferentes tipos de sensores y dispositivos.

De esta forma se implementa un sistema multiagente que mejora la atención médica y aporta valor añadido al paciente. A lo largo de este capítulo se analiza el caso de estudio, la infraestructura disponible, y sus deficiencias, así como las demandas de los pacientes y su comportamiento. A partir de este análisis se diseña un sistema multiagente, en el que cohabitarán tanto agentes instalados en el entorno del paciente como en el sistema informático encargado de gestionar el hogar.

En primer lugar ha sido necesario realizar un análisis del caso de estudio en colaboración con el personal médico del centro de salud al que pertenece el paciente y con sus familiares cercanos. En este caso ha sido necesario analizar la infraestructura, los tipos de problemas que se pueden plantear, los componentes y dispositivos que se pueden utilizar y cuáles son los gustos y preferencias del paciente a tratar. Posteriormente se ha aplicado la metodología SysML para definir un sistema multiagente que se adapta al análisis realizado, teniendo en cuenta ciertas restricciones tanto técnicas como asistenciales. En un principio, y dada la gran diversidad de dispositivos inalámbricos, se ha buscado construir agentes para aquellos dispositivos móviles más accesibles para pacientes y personal médico, con el objetivo de aumentar gradualmente el número de servicios que se ofertan y se ha proporcionado a los usuarios la información necesaria para utilizar la infraestructura generada. En cuanto a la evolución de los resultados obtenidos, se ha llevado a cabo tanto a nivel técnico como de satisfacción de los usuarios. Se establecen protocolos, métricas y mecanismos para identificar la eficiencia y eficacia de la propuesta. Además se han analizado tanto las ventajas como los inconvenientes de la propuesta.

5.2 Descripción del caso de estudio

En el caso de estudio que se presenta se ha desarrollado un prototipo para optimizar las tareas diarias de un paciente EPOC en su hogar utilizando el sistema multiagente HoCCAC. El sistema recoge información de sensores que capturan datos e interactúan con el paciente a través de planes de tareas que desarrolla el sistema multiagente propuesto. Estos planes de tareas están limitados tanto en espacio como en tiempo. La información que principalmente recogen los sensores instalados es la de localización del usuario en el entorno, ya que es la más significativa. Dependiendo la ubicación del paciente, este puede realizar unas tareas u otras. Por ejemplo el paciente no puede abordar un plan de actividades en el hogar si en ese preciso instante está fuera del hogar, por ejemplo, en la consulta médica. Para poder proporcionar servicios personalizados, el sistema necesita identificar y localizar a los usuarios de una forma transparente y ubicua.

La personalización es un aspecto muy demandado en la sociedad de la información. Un servicio personalizado permite aportar una calidad de servicio y además permite optimizar el tiempo de búsqueda del usuario, obteniendo un grado de satisfacción mucho mayor. La personalización es un servicio de valor añadido que aporta el sistema multiagente y supone una mejora con respecto a las técnicas tradicionales de cuidados en el hogar. Además el sistema también recoge, por ejemplo, información relativa a la temperatura en las dependencias del hogar del paciente y el estado de la iluminación en las zonas por las que transita el usuario. En este prototipo, la arquitectura HoCCAC se basa en la información recogida en el contexto del paciente EPOC, para planificar las tareas del mismo en su hogar y automatizar el control de las acciones diarias del paciente. El caso de estudio está compuesto por un conjunto de agentes, los cuales hacen uso de complejos mecanismos de razonamiento y planificación. Estos mecanismos generan planes de vida e informes basándose en el historial médico del paciente y en las indicaciones del personal médico y de los servicios sociales. De este modo el sistema desarrollado distribuye los tiempos, asigna tareas diarias al paciente y re-planifica dichas tareas de forma automática y dinámica sin afectar a su rutina diaria. Los agentes en el caso de estudio son capaces de actuar de forma autónoma y tienen la capacidad de ser ejecutados en dispositivos móviles

inalámbricos, pero es necesario que se comuniquen entre ellos para la toma de decisiones y la generación de planes.

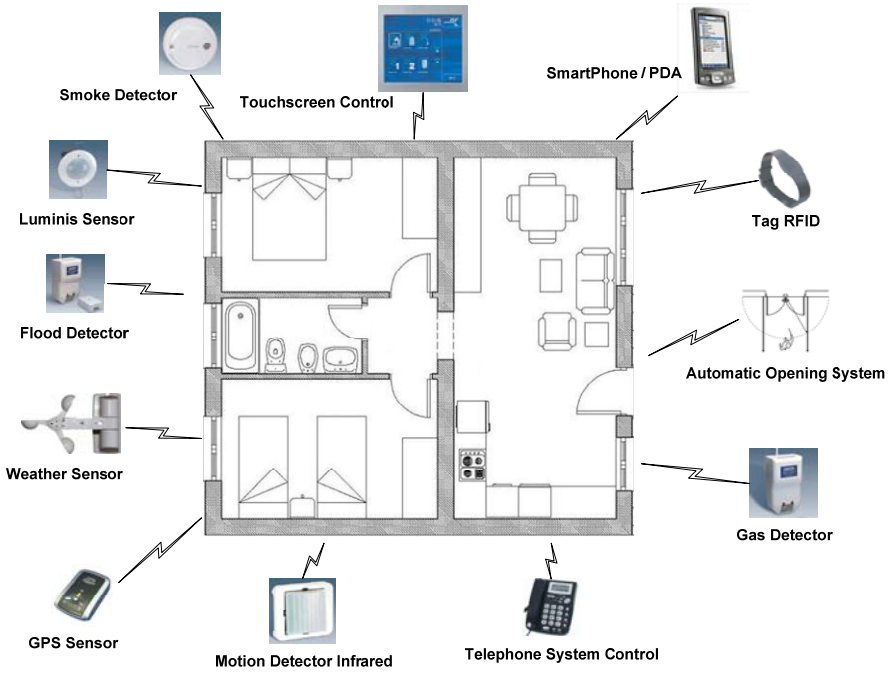


Figura 5.2 Plano general y componentes utilizados en el caso de estudio.

Para obtener los datos del contexto, procesarlos y ofrecer soluciones a los usuarios, en el caso de estudio, el sistema HoCCAC captura información del contexto a partir de una serie de componentes y dispositivos como se pueden ver en la Figura 5.2. El entorno simulado donde se desarrolla el caso de estudio trata de reproducir la vivienda de un paciente EPOC. Como muestra la Figura 5.2 la casa tiene conectados varios componentes y dispositivos que se enumeran a continuación:

- Detectores de movimiento pasivos por infrarrojos para techos Simon Vit@ 81800-30.
- Mecanismos de apertura de puertas automáticas. Tanto los detectores de movimientos como los mecanismos de apertura de puertas, interactúan a través del chip RFID con el paciente.

- Brazalete Sokymat ID Band Unique Q5 con una antena y chip RFID-Java-Crypto-Card con 32K y Crypto-CoProzessor (1024 bit RSA) compatible con SUNs JavaCard 2.1.1. Cada paciente está identificado por un brazalete.
- Sensores de luminosidad Simon Vit@ 81915-38 que se utilizan para gestionar el nivel de luminosidad de una estancia haciendo que se mantenga entre unos valores pre-asignados modificables.
- Una pantalla TFT de superficie Simon Vit@ 81221-38 que permite visualizar y gestionar los elementos principales del sistema.
- Detectores de inundación Simon Vit@ 81860-39.
- Detectores de gas Simon Vit@ 81861-39 diseñados para detectar la presencia de gases tóxicos y explosivos.
- Detectores de humo óptico Simon Vit@ 81862-39 apropiados para dar una advertencia precoz de fuegos en desarrollo.
- Módulo GPS, para mantener localizado al paciente cuando abandona el hogar.
- Estación meteorológica en comunicación con el sistema multiagente para conocer las condiciones meteorológicas fuera del hogar.
- Sistema de control telefónico, a través del cual se capturan desde la central telefónica las llamadas entrantes y salientes almacenándolas en el sistema multiagente. Este sistema está permanentemente comunicado con los servicios de atención sanitaria y además brinda los más diversos informes y estadísticas de uso.
- Dispositivos móviles que acceden a la información del sistema y que pueden modificar los parámetros del contexto a través del sistema multiagente.

Los sensores o actuadores están situados en posiciones estratégicas de la casa. Todos estos dispositivos son controlados por los agentes del sistema. Esta red de sensores es responsable de la generación de alarmas que compara el estado actual del usuario con los parámetros de su rutina diaria que tiene almacenado el sistema. El sistema puede generar alarmas si lo determinan los parámetros del sistema, por ejemplo, si el usuario se encuentra inmóvil más tiempo del establecido en una ubicación determinada, o si el usuario pasa más tiempo del especificado en la puerta de su casa, sin entrar, o si el usuario realiza un movimiento brusco en el pasillo y

luego permanece inmóvil durante un periodo largo de tiempo, etc. De este modo, se consigue que de cara a la interacción con los pacientes la infraestructura del sistema sea prácticamente imperceptible. Eso sí, en todo momento se proporciona un amplio soporte para la atención de los pacientes y detección de posibles irregularidades en su comportamiento médico y hábitos, sin interferir en sus actividades cotidianas. El caso de estudio monitoriza automáticamente la ubicación de los usuarios a través de las tecnologías RFID, NFC o GPS. 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.

5.3 Modelo de contexto

Los sistemas sensibles al contexto deben tener 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 tanto es necesario definir adecuadamente las funciones de los usuarios en cada contexto. Reproduciendo de nuevo el inicio del capítulo 2 de esta memoria de tesis, el contexto se define como "*cualquier información que puede ser usada para caracterizar la situación de entidades (usuario, lugar u objeto) que son consideradas relevantes en la interacción entre un usuario y una aplicación, incluyendo al usuario y a la aplicación"* [Dey *et al*. 2009]. 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 es lo que da 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.

Así pues, ya que el caso de estudio está basado en la arquitectura HoCCAC y a su vez, la arquitectura HoCCAC está basada en los sistemas sensibles al contexto es necesario situar a los usuarios como ejes centrales del diseño del caso de estudio. De

esta forma se logra un modelo del contexto de acuerdo a los objetivos definidos, teniendo muy presente la interacción humano-entorno. Esta interacción humano-entorno debe guiarse por la ubicuidad, sencillez y transparencia para los usuarios. El modelo de 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 resulta una tarea muy compleja diseñar un modelo de un contexto global para cualquier escenario posible. El caso de estudio que se presenta está enfocado a mejorar el plan de vida y la asistencia social a personas con algún grado de dependencia en su hogar, especialmente a personas con la enfermedad de EPOC. El campo de acción que abarca es muy concreto, pero el contexto puede ser muy variable. Va a depender de las características del hogar y del grado de dependencia y poder de decisión del paciente. Por tal motivo es necesario definir un modelo del contexto que sea flexible y escalable, que tenga en cuenta sus características específicas y describa claramente los distintos elementos que lo componen y cómo se relacionan entre ellos. A continuación se pasa a describir de forma básica el escenario para el caso de estudio que se presenta:

Un paciente EPOC se encuentra en su hogar realizando un plan de vida que ha sido marcado por el personal médico responsable de su cuidado. Esporádicamente el paciente puede recibir atención de un cuidador, el cual puede ser personal especializado o una persona cercana a su entorno familiar sin formación profesional para ejercer labores de asistencia a personas dependientes. Los cuidadores y administradores monitorizan constantemente el contexto de aplicación, formado por el paciente y el entorno. En el caso del paciente se monitoriza su plan de vida, su localización, su medicación, sus horas de oxígeno, etc. En cuanto al entorno, el paciente EPOC es muy sensible a cambios bruscos en su entorno debidos por ejemplo a cambios de temperatura, humedad, humos, etc. Cualquier cambio en los niveles enumerados puede implicar graves consecuencias sanitarias en el paciente EPOC, ya que su capacidad pulmonar es muy restringida. Los cuidadores también pueden recibir ayuda del personal médico. Los servicios ofrecidos por el sistema al paciente

pueden ser ejecutados por componentes automáticamente, por el propio paciente o por el resto de actores que interactúan con el sistema (personal médico, cuidadores, visitantes o administradores).

El escenario descrito tiene como objetivos la atención personalizada del paciente y la mejora de las condiciones de vida del mismo. Así pues, todos los elementos del modelo de contexto son diseñados en torno al paciente. El proceso seguido para obtener el modelo de contexto se detalla a continuación. El diseño de un modelo genérico del contexto, lo suficientemente abstracto para abarcar el escenario descrito anteriormente, permite añadir gradualmente nuevos componentes. De esta forma se logra un modelo especializado con mayor flexibilidad para adaptarse a las necesidades y características de cada contexto. Para ello, es necesario determinar quiénes y qué componen el contexto, información que dará origen al desarrollo del sistema.

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

Inicialmente debe identificarse quiénes están presentes en el entorno que se desea modelar, ya que en todo momento los elementos identificados pueden influir los unos en los otros de forma prevista o imprevista. Esto implica la **identificación** de los usuarios y de aquellos componentes (no humanos) que lo rodean. Identificando a los usuarios, es posible conocer sus perfiles y todo aquello que pueda influirles en un momento determinado. Una vez identificados los usuarios y componentes es indispensable situar los elementos que forman parte del modelado. La **localización** es muy importante en un entorno de computación ubicua, incluyendo la localización tanto de los usuarios como de los recursos. Según el lugar en el que se encuentren los elementos, las necesidades del usuario pueden cambiar, por lo que el entorno debe adaptarse a sus necesidades y características, para poder ofrecer adecuadamente los recursos disponibles. Así se consigue ofrecer servicios personalizados a los usuarios. Los servicios personalizados no solamente ayudan a que los usuarios obtengan una atención personalizada y de alta calidad, sino que también ayudan a los

administradores a estudiar los hábitos de comportamiento de los pacientes. En este sentido se hace necesaria una coordinación inteligente de los servicios que se ofrecen en el contexto.

Una vez identificados y localizados los usuarios y componentes 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. 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. El sistema también debe estar preparado para realizar cualquier actividad en cualquier momento. Sin embargo, las actividades pueden depender de quienes estén en una ubicación en cada momento. Por último, la medición de parámetros tanto del usuario como del entorno facilita conocer su estado para actuar en consecuencia ante una situación determinada o un imprevisto.

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 componentes) y sus interacciones fundamentales (estimulación y percepción). 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 componentes, quienes a su vez reaccionan y los estimulan conforme a una situación determinada.

Es importante resaltar de nuevo, que el enfoque del caso de estudio presentado sitúa al usuario como el centro del diseño, por lo que el resto de elementos actúan en relación a él. De esta forma, las principales entidades en este modelo son las siguientes:

- **Usuario**. Son todas las personas que hacen uso de los servicios proporcionados por el sistema. Todos los usuarios son capaces de ejecutar servicios y modificar el contexto con diferente nivel de permisos dependiendo del perfil de usuario. Es la entidad principal y en base a esta actúan el resto de entidades.
- **Componentes**. Son todos aquellos dispositivos o elementos capaces de modificar el contexto, así como de proporcionar o de recabar información sobre el mismo.

- **Servicios**. 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 en el contexto.

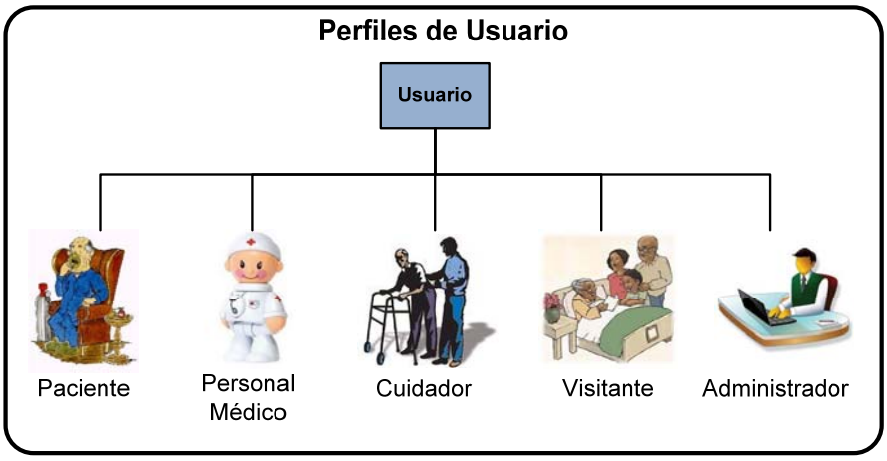


Figura 5.3 Perfiles de usuario.

A continuación se describen los perfiles de usuario, como muestra la Figura 5.3:

- **Paciente**: Enfermo EPOC que requiere atención personalizada en su hogar. Es el mayor beneficiario de los servicios que ofrece el sistema para mejorar su calidad de vida. Además también pueden interactuar con el sistema de la forma más sencilla, natural e intuitiva posible.
- **Personal Médico**: Está formado por médicos, enfermeros y auxiliares de enfermería dedicados al cuidado de pacientes, y que cuentan con formación profesional para ejercer dichos cuidados. Pueden hacer uso de una serie de servicios y dispositivos para llevar a cabo sus tareas y actividades para mejorar la calidad de vida del paciente. También monitorizan el estado del paciente. Pueden proporcionar información al sistema, con el fin de almacenarla, gestionarla y analizarla, y así modificar el comportamiento del sistema. Además pueden emitir diagnósticos y proponer tratamientos a través del sistema.
- **Cuidador**: Usuario cercano al entorno del paciente y que además cuida de él, 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 en el sistema. Tiene mucha influencia en el comportamiento del sistema, debido a su constante participación en el cuidado del paciente.

- **Visitante**: Persona que entra en el área de influencia del sistema de forma temporal y que no está relacionado con los cuidados al paciente. Tienen una interacción temporal con el sistema, limitándose a realizar acciones simples, con escasa capacidad para modificar el contexto de forma directa.
- **Administrador**: Usuario con la capacidad para supervisar el adecuado funcionamiento del sistema. Tiene la capacidad de gestionar y actuar sobre los servicios y modificar el funcionamiento global del sistema.

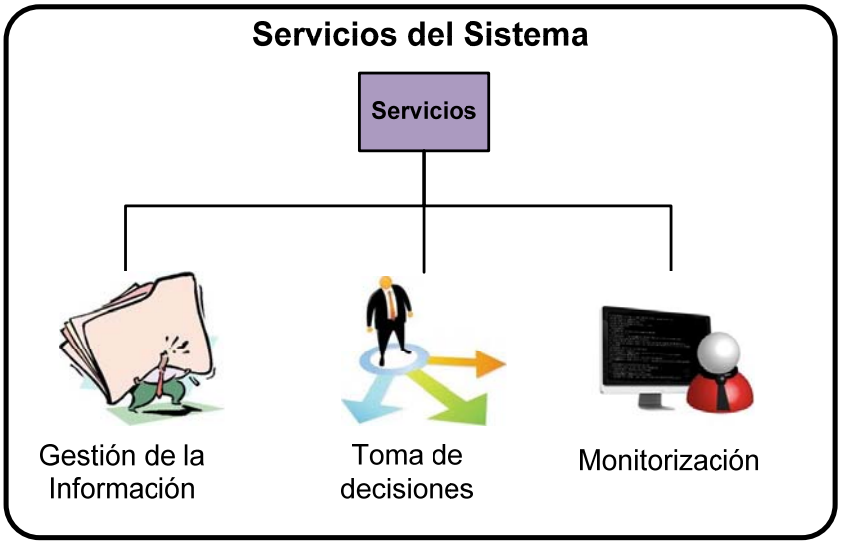


Figura 5.4 Principales servicios del sistema.

Por otro lado los principales servicios identificados en el sistema se describen a continuación y se pueden ver en la Figura 5.4:

- **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. La monitorización del usuario obtiene información sobre su estado, identificándolo

y localizándolo. La monitorización del entorno realiza el seguimiento del estado del entorno identificando y localizando los componentes necesarios para la interacción de los usuarios y su entorno.

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

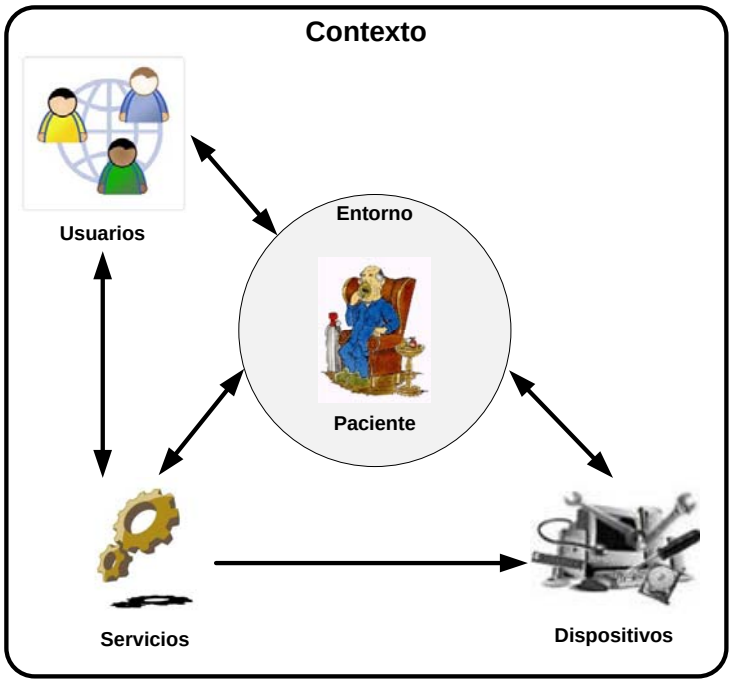


Figura 5.5 Ciclo de vida del sistema.

Una vez definidas las principales entidades, es posible describir de forma más detallada las relaciones de las entidades en el sistema. Como se muestra en la Figura 5.5 los servicios de monitorización obtienen información del usuario y su entorno a través de dispositivos o usuarios. La información se analiza y procesa por un servicio (gestor de información) que le da consistencia a los datos y los almacena. Una vez procesada la información, se definen y personalizan las actividades a efectuar y

quiénes las deben ejecutar. Las decisiones (otro servicio) se envían a los usuarios o dispositivos, para que se apliquen las acciones correspondientes y así estimular al contexto. Este ciclo se repite con cada nueva interacción con el contexto.

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

5.4 Análisis y diseño

A partir del caso de estudio presentado se obtienen los requisitos necesarios para la construcción de un sistema multiagente capaz de resolver el problema. Para ello es necesario utilizar una metodología de análisis y diseño, centrada en el contexto, que permita modelar un sistema multiagente y que facilite la construcción de entornos inteligentes sensibles al contexto. Dicho entorno inteligente permite mejorar la calidad de vida de pacientes con algunas limitaciones físicas, de manera inalámbrica, y posibilitando la personalización de perfiles, planes, actividades, y objetivos de los diferentes agentes involucrados. Los usuarios del sistema tienen un acceso personalizado al sistema por medio de un perfil de agente interface personal, que se ejecuta en su dispositivo móvil. Es necesario definir las interacciones que pueden producirse entre los elementos del sistema y proporcionar los medios adecuados para garantizar una comunicación ubicua. El sistema sensible al contexto debe ser capaz de ofrecer servicios a sus usuarios y algunos de estos servicios requieren capacidades de computación.

En cuanto a los requisitos funcionales del sistema, el sistema debe permitir la configuración de las entradas iniciales, para que los agentes implementen los comportamientos acordes a las entradas y así actúen de manera correcta. El sistema debe incorporar diferentes agentes que incorporen múltiples objetivos. Los agentes con capacidades especiales de computación son los encargados de enviar de manera cíclica y constante información variable a cerca del contexto. La comunicación debe

ser distribuida porque los agentes pueden interactuar entre ellos para determinar la información que han capturado del entorno.

Lo primero que se define a continuación, una vez visto en la sección anterior el modelo contextual, las características y requerimientos del sistema es el modelo de roles.

5.4.1 Modelo de roles

El modelo de roles presenta los roles que hay en el sistema. La identificación de roles ayuda en la captura y refinamiento de potenciales requisitos en el sistema. Con el modelo de roles se consigue llegar a los casos de uso del sistema, que proporcionan uno o más escenarios para indicar cómo interactúa el sistema con el usuario o con otro sistema y así conseguir un objetivo específico. La identificación de roles es probablemente la parte más importante del análisis SysML, ya que permiten detectar los conocimientos, metas, actividades, protocolos, permisos y responsabilidades asociados a la organización del caso de estudio. Posteriormente estos roles van a ser asignados a los agentes que finalmente implementan el sistema. Es por ello que una mala identificación de los roles del sistema supone introducir errores en el proceso de análisis y diseño, que se van magnificando conforme se avance en el nivel de detalle. Después de un cuidadoso análisis de los requisitos los roles identificados en el caso de estudio son:

- **Rol Intérprete**: Gestiona el comportamiento del paciente (monitorización, localización, tareas diarias, anomalías, etc.). Desempeña además las tareas de: controlar los datos personales del paciente; elaborar planes de acción para el paciente; llevar un seguimiento de los tratamientos asignados por el personal médico; informar periódicamente de la evolución del paciente ante los tratamientos administrados; informar al personal médico y cuidadores 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; y finalmente, manejar situaciones de riesgo en las que los datos recogidos para el paciente sean anómalos. Además junto con el rol *Información* se encarga de gestionar toda la

información que proporciona el entorno del paciente. El resto de roles se comunican con este rol para informar sobre los cambios en tratamientos, tareas e información médica para que éste actualice la información. El rol *Intérprete* administra las altas, bajas y modificaciones de pacientes y controla la conexión y desconexión de los pacientes en el sistema.

- **Rol Cuidador**: Gestiona y controla el día a día del paciente interactuando con el rol *Intérprete*. Se encarga de modificar los parámetros del sistema cuando el paciente no puede. Además se apoya en el rol *Personal Médico* cuando detecta situaciones de riesgo médico para el paciente. Es un rol muy utilizado cuando el estado de dependencia del paciente es muy alto.
- **Rol Personal Médico**: Se encarga de asignar tratamientos a los pacientes, así como las tareas que los pacientes deben realizar para completar dichos tratamientos. Las citas con los pacientes pueden ser asignadas por el propio médico o de forma automática por el sistema. Este rol interactúa de forma continua con el rol *Cuidador* cuando el paciente es una persona muy dependiente.
- **Rol Paciente**: Es un rol que se asigna al paciente. Puede recibir información del estado del paciente y puede actuar sobre parámetros del sistema. Además recibe los planes de tareas que debe realizar el paciente.
- **Rol Visitante**: Es un rol que se asigna a los visitantes del paciente y tiene las funciones mínimas. Sólo puede recibir información muy general del estado del paciente y en casos excepcionales puede actuar sobre parámetros poco influyentes en el sistema.
- **Rol Administrador**: Se encarga de administrar el sistema y controlar el buen funcionamiento de todos los componentes que lo forman.
- **Rol Componente**: Este rol hace uso de dispositivos para activar o desactivar automáticamente servicios físicos por ejemplo puertas automáticas, alarmas, sensores de humo, etc. Se encarga de proporcionar la información asociada al contexto a los roles *Intérprete* e *Información*. Además se encarga de añadir nuevos dispositivos y de eliminar los dispositivos innecesarios o que no funcionan correctamente.

- **Rol Información**: Mantiene un histórico de la información gestionada por el sistema. Se relaciona con todos los roles del sistema y les suministra información del contexto. Principalmente se relaciona con el rol *Intérprete*, con el que tiene una interacción continua de intercambio de información

5.4.2 Casos de uso

En este apartado se describen los casos de uso más comunes asociados al caso de estudio. Cada caso de uso proporciona uno o más escenarios que indican cómo debe interactuar el sistema con los usuarios o con otros sistemas para conseguir un objetivo específico. Un caso de uso es una secuencia de interacciones que se desarrollan entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y otros sistemas. Algunos de los casos de uso más comunes del caso de estudio son:

- **Inicio de un plan de tareas**: El paciente puede informar del inicio de un nuevo plan de tareas. También, de forma automática el componente encargado de detectar el inicio de la primera tarea informa del inicio del plan.
- **Fin de un plan de tareas**: El paciente puede informar del fin de un plan de tareas al sistema. También de forma automática el componente encargado de detectar el fin de la última tarea informa del fin del plan. Cuando finaliza un plan de tareas se almacena el tiempo de ejecución y el cumplimiento del plan de tareas. También se guarda un detalle de cada tarea y posibles incidencias que hayan surgido en la realización de las tareas.
- **Modificación de un plan de tareas**: El propio sistema puede modificar un plan de tareas automáticamente, por ejemplo cuando el personal médico modifica el diagnóstico del paciente. También el cuidador y personal médico en un momento dado puede modificar un plan de tareas de forma manual.
- **Re-planificación de un plan de tareas**: El sistema realiza una re-planificación de un plan de tareas cuando se encuentra un imprevisto en la realización de una tarea o cuando se produce un error en un dispositivo.

- **Alta de nuevo paciente**: Cuando un nuevo paciente desea ser monitorizado y tratado por el sistema tiene que proporcionar sus datos y los parámetros relativos a las condiciones de vida deseadas en el hogar. Esta información incluye permisos de acceso a sus cuidadores, personal médico y visitantes que pueden interactuar con el sistema.
- **Baja de un paciente**: La baja de un paciente constituye la no monitorización del paciente. Se elimina el perfil del paciente, no se eliminan los planes generados por el propio sistema para mejorar las condiciones de vida de ese paciente en particular. Estos planes generados pueden ser necesarios en el futuro para generar otros planes similares a otro paciente.
- **Conexión de un usuario**: Los usuarios solicitan conectarse al sistema haciendo uso de su respectivo nombre de usuario y contraseña. Se comprueban los datos, el perfil que le corresponde (cuidador, personal médico, visitante, administrador y, en caso de ser correctos, el usuario es autorizado y conectado al sistema.
- **Desconexión de un usuario**: Cuando un usuario se desconecta del sistema, se actualiza el estado del usuario y se notifica el éxito de la operación.
- **Alta de dispositivo**: Cuando se desea añadir un nuevo componente al sistema se parametriza y configura para indicar el tipo de información que va a suministrar al sistema. Una vez configurado el nuevo dispositivo la integración en el sistema es automática.
- **Baja de dispositivo**: Cuando se elimina un componente del sistema se informa del nuevo estado y se actualizan todos los planes que tuvieran relación con el componente.
- **Consultar estado del paciente**: Cuando un usuario (cuidador, personal médico, visitante, administrador) solicita información de un paciente, el sistema atiende la petición y le envía la información requerida.
- **Consultar parámetros del sistema**: Cuando un usuario (cuidador, personal médico, visitante, administrador) solicita información de los atributos de un paciente, el sistema atiende la petición y le envía la información requerida.

- **Modificar parámetros del sistema**: Al modificar los atributos del paciente, el usuario (cuidador, personal médico, visitante, administrador) envía los nuevos valores y se envía un aviso de confirmación al guardar la información.
- **Envío de alertas**: Cuando un paciente está en una situación de peligro, se envía una alerta a los cuidadores o personal médico responsable del paciente. Dependiendo la configuración de los tipos de alertas se avisa a un perfil de usuario u otro.

5.4.3 Diseño SysML

Llegados a este punto se da por finalizado el análisis inicial del caso de estudio y 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.

5.4.3.1 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. Estos diagramas permiten definir la estructura interna de cada uno de los agentes. Debido a que las características de los agentes de HoCCAC ya han sido descritas en el capítulo 4 de esta memoria, tan solo se describen a continuación las funcionalidades propias de los agentes en este caso de estudio:

- **Agente Proveedor Situación**: Tiene la capacidad de gestionar todos los dispositivos relacionados con la localización en el hogar del paciente. Además se comunica con el *agente Intérprete* para proporcionarle los datos de localización y con el *agente Contenedor* para mantener un histórico de la información que recibe. También se encarga de agregar los nuevos dispositivos de localización que se instalen en el sistema y darlos de baja una vez desinstalados.
- **Agente Proveedor Inundación**: Tiene la capacidad de gestionar todos los dispositivos relacionados con el control de inundaciones en el hogar. Se comunica con el *agente Intérprete* para proporcionarle los datos de los sensores de inundación y con el *agente Contendor* para mantener un histórico de la

información que recibe. También se encarga de agregar los nuevos sensores de inundación que se instalen en el sistema y darlos de baja una vez desinstalados.

- **Agente Proveedor Luz**: Tiene la capacidad de gestionar todos los dispositivos relacionados con el control de luz en el hogar. Se comunica con el *agente Intérprete* para proporcionarle los datos de los sensores de luz y con el *agente Contenedor* para mantener un histórico de la información que recibe. También se encarga de agregar los nuevos sensores de luz que se instalen en el sistema y darlos de baja una vez desinstalados.
- **Agente Proveedor Gas**: Tiene la capacidad de gestionar todos los dispositivos relacionados con el control de escapes de gas en el hogar. Se comunica con el *agente Intérprete* para proporcionarle los datos de los sensores de inundación y con el *agente Contenedor* para mantener un histórico de la información que recibe. También se encarga de agregar los nuevos sensores de escapes de gas que se instalen en el sistema y darlos de baja una vez desinstalados.
- **Agente Proveedor Humos**: Tiene la capacidad de gestionar todos los dispositivos relacionados con el control de humos en el hogar. Se comunica con el *agente Intérprete* para proporcionarle los datos de los sensores de inundación y con el *agente Contenedor* para mantener un histórico de la información que recibe. También se encarga de agregar los nuevos sensores de control de humos que se instalen en el sistema y darlos de baja una vez desinstalados.
- **Agente Proveedor Temperatura**: Tiene la capacidad de gestionar todos los dispositivos relacionados con el control de temperatura en el hogar. Se comunica con el *agente Intérprete* para proporcionarle los datos de los sensores de temperatura y con el *agente Contenedor* para mantener un histórico de la información que recibe. También se encarga de agregar los nuevos sensores de temperatura que se instalen en el sistema y darlos de baja una vez desinstalados.
- **Agente Proveedor Control**: Tiene la capacidad de gestionar todos los parámetros de habitabilidad confortable en el hogar. Cualquier usuario a través del *agente Interface* instalado sobre un dispositivo táctil, puede modificar los valores relacionados con la temperatura o niveles de luz deseados en el hogar. Se comunica con el *agente Intérprete* para proporcionarle las actualizaciones de

datos y con el *agente Contenedor* para mantener un registro de la información que se introduce. También se encarga de activar o desactivar sensores en el hogar. El *agente Intérprete* puede actuar sobre el *agente Proveedor Control* de forma automática para modificar los parámetros del sistema.

- **Agente Proveedor Meteo**: Tiene la capacidad de proporcionar información sobre el estado y previsiones meteorológicas fuera del hogar. Se comunica con el *agente Intérprete* para proporcionarle la información meteorológica y con el *agente Contenedor* para mantener un histórico de la información que recibe.
- **Agente Proveedor GPS**: Tiene la capacidad de gestionar toda la información recibida sobre la localización del paciente cuando está fuera del hogar. Se comunica con el *agente Intérprete* para proporcionarle los datos recibidos y con el *agente Contenedor* para mantener un histórico de la información.

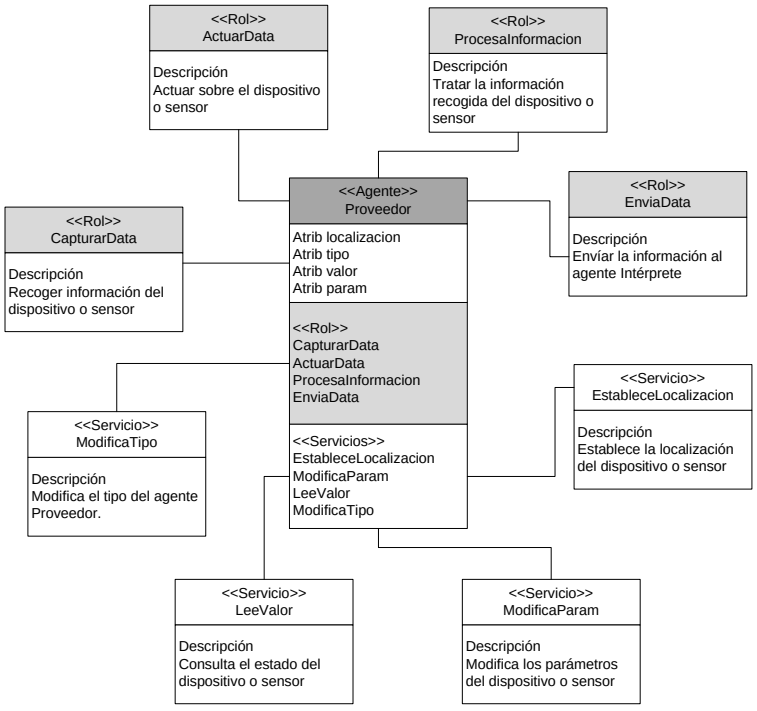


Figura 5.6 Diagrama de definición de bloques SysML para un agente Proveedor.

La Figura 5.6 muestra los roles y servicios de un *agente Proveedor*. Los distintos tipos de *agentes Proveedores* tienen la misma definición, lo único que cambia es

el tipo de información que recogen del dispositivo. Esto supone una gran ventaja para el sistema ya que es muy sencillo añadir un nuevo *agente Proveedor* al sistema cuando se desea obtener un nuevo atributo de contexto. El *agente LcA* es el encargado de las altas y bajas de *agentes Proveedores* del sistema.

- **Agente Contenedor**: Gestiona toda la información que recogen el resto de agentes del sistema. La Figura 5.7 muestra el diagrama de bloques del agente Contenedor con sus roles y servicios. Tiene una comunicación constante con el *agente Intérprete* en ambos sentidos. Por un lado el *agente Intérprete* le pasa información sobre los procesos y planes que realiza para mantener un histórico y por otro lado el *agente Contenedor* responde al *agente Intérprete* con la información que le solicita continuamente de planes pasados.

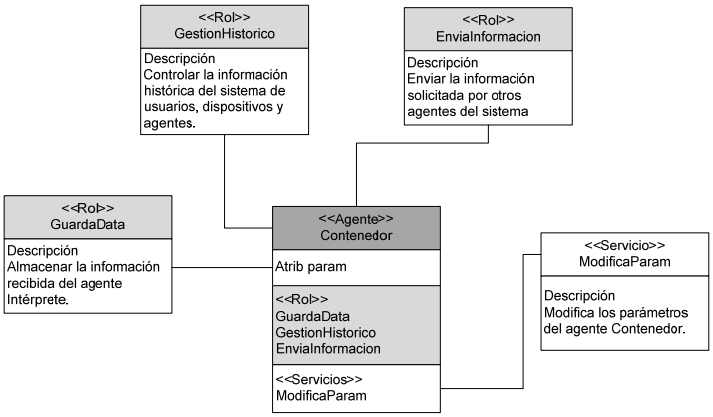


Figura 5.7 Diagrama de definición de bloques SysML para el agente Contenedor.

- **Agente Intérprete**: En este caso de estudio concreto el *agente Intérprete* tiene la capacidad de crear planes de actividades para el paciente de acuerdo a la información que proporcionan todos los agentes del sistema. Al mismo tiempo también mejora y optimiza los planes en tiempo de ejecución. Además puede re-planificar un plan de actividades concreto cuando surge una adversidad o hay un cambio brusco de los parámetros del sistema. Otra tarea del *agente Intérprete* es la monitorización y evaluación de los tratamientos asociados al paciente. La Figura 5.8 muestra el diagrama de bloques del *agente Intérprete* con sus roles y servicios.

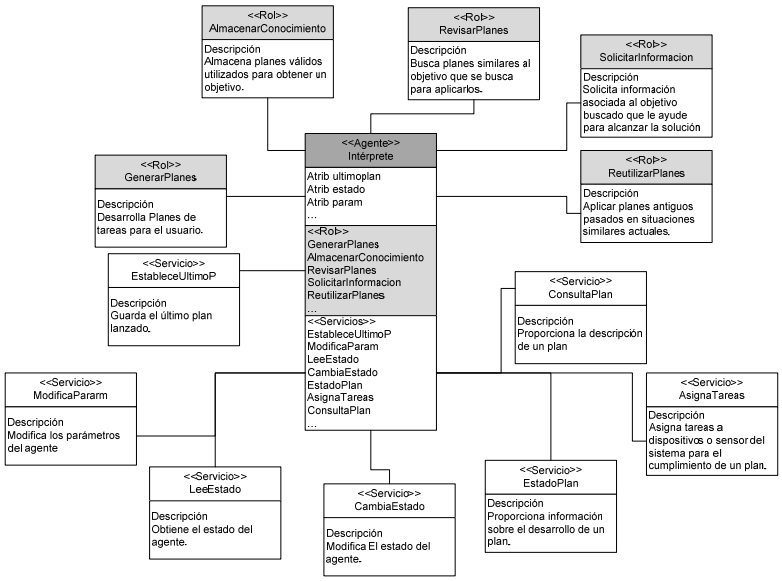


Figura 5.8 Diagrama de definición de bloques SysML para el agente Intérprete.

- **Agente LcA**: Gestiona la creación y eliminación automática de *agentes Proveedores* de información. La Figura 5.9 muestra el diagrama de bloques del *agente LcA* con sus roles y servicios. Informa al *agente Intérprete* de los nuevos agentes creados y las nuevas fuentes de información.

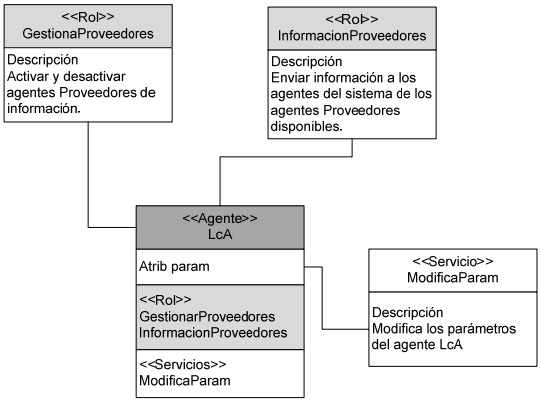


Figura 5.9 Diagrama de definición de bloques SysML para el agente LcA.

- **Agente Interface**: Gestiona la conexión y desconexión de usuarios al sistema (pacientes, cuidadores, visitantes o personal médico). Proporciona distintos

perfiles dependiendo el usuario que quiera acceder al mismo. Además al paciente le muestra el plan de actividad en curso y planes de actividad alternativos. Dependiendo del grado de dependencia del paciente el rol *Cuidador* tiene un perfil más similar al del *Paciente* o no, cuando se trata de un *Paciente* con un grado alto de dependencia el perfil de *Cuidador* es menos restrictivo para que tenga más capacidad de interactuar con el sistema. Los roles *Cuidador, Paciente* y *Personal Médico* pueden modificar los planes de actividad de forma manual. La Figura 5.10 muestra el diagrama de bloques del agente Interface con sus roles y servicios.

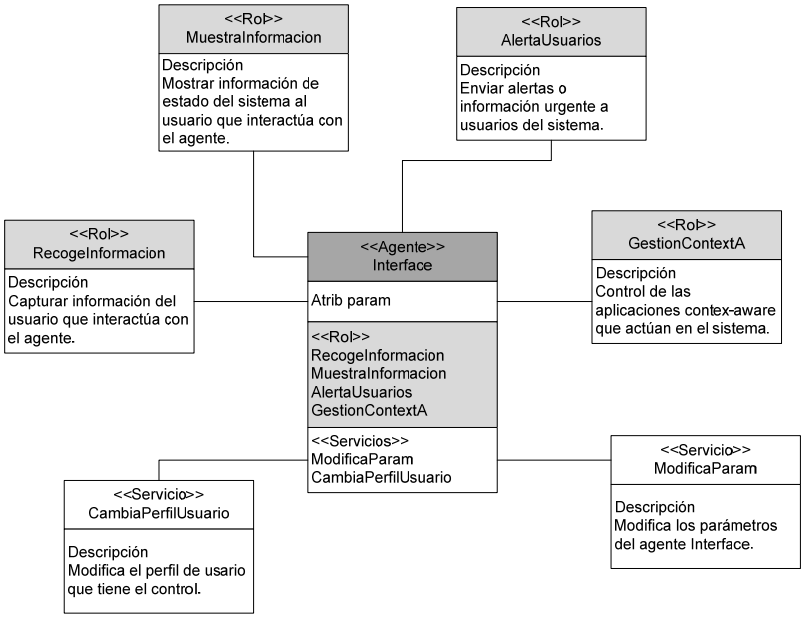


Figura 5.10 Diagrama de definición de bloques SysML para el agente Interface.

5.4.3.2 Diagramas de secuencia

Los diagramas de secuencia 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 de usuario**: El usuario introduce sus datos de registro (nombre de usuario y contraseña) a través del *agente Interface*. Éste valida los datos para comprobar si son correctos, informando en todo caso al *agente Interface*. Dependiendo de los datos introducidos, el *agente Interface* asume un rol determinado (*Paciente, Cuidador, Personal Médico, Visitante, Administrador*), activando las funcionalidades propias de cada rol. Si el *agente Interface* asume el rol *Paciente*, el *agente Intérprete* desarrolla una propuesta de planificación de tareas y envía el resultado a todos los *agentes Proveedores* y al *agente Interface* para que controlen el cumplimiento de las tareas.
- **Desconexión de usuario**: Cuando un usuario se desconecta del sistema, el *agente Interface* informa al *agente Intérprete,* quien a su vez confirma la operación. El *agente Interface* cambia de rol y se pone en espera para que un nuevo usuario se conecte. Si el usuario desconectado es un paciente y había un plan de tareas en marcha, se almacena el estado en el que queda el plan y se realiza una re-planificación inmediata para cuando el paciente recupere la conexión.
- **Nuevo Plan**: El personal médico o cuidadores asignan un nuevo plan de tareas a un paciente rellenando un formulario de alta de plan de tareas a través del *agente Interface*. Los datos se envían al *agente Intérprete*, quien los trata y se los pasa al *agente Contenedor* para que los almacene. El *agente Intérprete* informa al *agente Interface* del resultado de la operación. A continuación, el *agente Intérprete* realiza un nuevo plan de tareas y lo envía al paciente conectado al sistema.
- **Modificación de Plan**: El personal médico o cuidadores seleccionan un plan de tareas asociado a un paciente a través del *agente Interface*. La información se envía al *agente Intérprete*, quien trata la información y se los pasa al *agente Data* para que los almacene. El *agente Intérprete* informa al *agente Interface* del resultado de la operación. A continuación, el *agente Intérprete* realiza un nuevo plan de tareas y lo envía al paciente conectado al sistema.
- **Consulta Plan**: El personal médico o cuidadores seleccionan uno de los planes asociados a un paciente a través del *agente Interface*, el cual a su vez solicita al

agente Intérprete la lista de planes de tareas asignados a dicho paciente. La información de los planes de tareas es enviada al *agente Interface*.

- **Activación de componente**: El administrador a través del agente LcA informa de la incorporación de un nuevo sensor o actuador en el sistema indicando sus características. Automáticamente se crea el *agente Proveedor* asociado al dispositivo o se asocia a un *agente Proveedor* ya creado. El *agente Proveedor* envía una señal de prueba al dispositivo y este le responde para validar la comunicación. Cuando el *agente Proveedor* recibe la señal del dispositivo informa al *agente Intérprete* del nuevo proveedor de información y comienza el funcionamiento del dispositivo en el sistema. Además el *agente Intérprete* también informa al *agente Contenedor* para que asocie los datos que reciba en el sistema de información.
- **Desactivación de componente**: La desactivación de un dispositivo se puede producir de forma manual o de manera automática cuando el *agente Intérprete* detecta que los datos que recibe de un *agente Proveedor* son erróneos. De forma automática el agente Intérprete informa de la desactivación de un *agente Proveedor* al agente Lca y este la ejecuta. De manera manual el Administrador a través del *agente LcA* puede desactivar cualquier *agente Proveedor* e informar de la desactivación al *agente Intérprete*.
- **Petición de información**: El *agente Intérprete* solicita información a un *agente Proveedor* y este se conecta con el dispositivo para capturar los datos y pasárselos al *agente Intérprete*. El *agente Intérprete* procesa la información recibida y actualiza el plan de tareas en ejecución. El *agente Intérprete* también transfiere la información al *agente Contenedor*.
- **Nueva información**: Cuando un *agente Proveedor* captura información que le proporciona un dispositivo porque por ejemplo cambian los niveles de temperatura o el Paciente cambia de localización, le pasa esta información al *agente Intérprete* y este la trata y se la pasa al *agente Contenedor* para que la almacene. Tras recibir la información, el *agente Intérprete* actualiza el plan de tareas en ejecución.

5.4.3.3 Diagramas de Interacción

Los Diagramas de Interacción, 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 *agente Intérprete* realiza una asignación de un plan de tareas y la envía al paciente correspondiente. El paciente recibe el plan de tareas a través del *agente Interface* y dependiendo de la tarea el propio paciente valida su cumplimiento o automáticamente la valida el *agente Intérprete* cuando recibe una señal específica desde un *agente Proveedor*. Para realizar una planificación de tareas óptima, el *agente Intérprete* recopila toda la información asociada al paciente en el sistema, para ello realiza peticiones al *agente Contenedor*. El tiempo necesario para desarrollar una tarea se estima teniendo en cuenta el tiempo que se introduce para el desarrollo de la misma y el promedio empleado en realizar la misma tarea en ocasiones anteriores. Se asignan los planes de tareas a los pacientes teniendo en cuenta la información introducida por los cuidador y personal médico, la prioridad de las tareas de un plan, 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 paciente. Una vez realizada la asignación global del plan de tareas, el *agente Intérprete* 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 cualquier momento el *agente Intérprete* puede adelantar o retrasar la ejecución de una tarea en tiempo de ejecución.
- **Cambio de plan**: Se producen cambios de planes en ejecución cuando hay cambios en los tratamientos asociados a un paciente. Los cambios de tratamiento los introducen cuidadores y personal médico a través del *agente Interface*. El *agente Interface* le pasa la información al *agente Intérprete*. El *agente Intérprete* una vez que procesa la información realiza la asignación de un nuevo plan al paciente. Antes de asignar el nuevo plan, el *agente Intérprete* guarda el estado en el que se encuentra el plan en ejecución.

- **Re-planificación**: Se produce una re-planificación cuando hay un cambio brusco en información sensible al paciente o se produce una finalización anómala de una tarea. Cuando el *agente Intérprete* detecta alguna de estas situaciones automáticamente re-planifica el plan en ejecución y se lo envía al *agente Interface* para que lo reciba el paciente.
- **Localización de paciente**: El *agente Intérprete* actualiza la ubicación de un paciente 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 *agente Interface* solicita al *agente Intérprete* dicha información, quien a su vez envía la nueva ubicación del paciente al *agente Interface*. El *agente Interface* la representa a través de un mapa predefinido del hogar.
- **Información de paciente**: Un cuidador o personal médico selecciona un paciente a través del *agente Interface*. Éste realiza una petición al *agente Intérprete* para obtener la información del paciente seleccionado. La información es enviada al *agente Interface*. Si el cuidador o personal médico actualiza la información médica del paciente, la petición se envía al *agente Intérprete*, quien actualiza los datos e informa del resultado de la operación. Si los valores introducidos presentan algún conflicto, se avisa al paciente a través del *agente Interface*. A continuación, el *agente Intérprete* realiza una nueva planificación de tareas y la envía al paciente.
- **Inicio de plan**: Un paciente selecciona la tarea inicial de un plan. El *agente Intérprete* automáticamente también puede recibir una señal de un *agente Proveedor* y cambiar a estado iniciada, dicha tarea. El *agente Interface* marca la tarea como iniciada, guarda la fecha y hora de comienzo de la tarea e informa al *agente Intérprete* del inicio de la tarea.
- **Fin de plan**: Si una tarea está iniciada, el paciente puede seleccionarla para finalizarla o el propio *agente Intérprete* la pude dar por finalizada automáticamente cuando recibe la señal oportuna de un *agente Proveedor*. El *agente Interface* calcula el tiempo que ha tardado en realizarse la tarea, la marca como finalizada e informa al *agente Intérprete* para que la información asociada a la tarea sea almacenada por el *agente Contenedor*.

- **Nuevo parámetro**: El administrador selecciona un paciente a través del *agente Interface* e introduce los datos de un nuevo atributo asociado al paciente. El *agente Intérprete* almacena la información y envía la lista actualizada de atributos al *agente Contenedor*.
- **Modificación de parámetro**: Tras seleccionar un atributo de un paciente, el administrador a través del *agente Interface* modifica sus datos. El *agente Intérprete* recoge los cambios y envía la lista actualizada de atributos al *agente Contenedor*.
- **Borrado de parámetro**: Tras seleccionar un atributo de un paciente, el administrador a través del *agente Interface* lo elimina. El agente Interface envía la información al *agente Intérprete* para que la procese. El *agente Intérprete* envía la lista actualizada de atributos al *agente Contenedor*.

5.4.3.4 Diagramas de estado

Los diagramas de estados 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.

- **Agente Intérprete**: Es el agente principal del sistema y desde que se arranca el sistema siempre está iniciado. Sólo pasa a estado finalizado cuando el agente se reinicia o se para el sistema. El resto de estados del agente son:
    - *Planificando*: En este estado se encuentra cuando recibe un plan de tareas del *agente Interface* y desarrolla un plan apropiado para un paciente. También pasa a este estado tras la recepción de una incidencia o una finalización anómala de una tarea y tiene que re-planificar el plan en ejecución. En este estado el agente puede necesitar más información y pasar a los estados *Petición de proveedor* o *Petición de datos* para solicitar información a *agentes Proveedores* o al *agente Contenedor*. Una vez recibida la información continúa con la planificación.
    - *Procesando información*: En este estado se encuentra el *agente Intérprete* tras recibir información de los *agentes Proveedores* o del

agente Contenedor (estados, *Recepción de proveedor* o *Recepción de Datos*). Toda la información que recibe la procesa y finalizado el proceso puede solicitar nueva información (estados, *Petición de proveedor o Petición de datos*) o si la información es muy significativa y tiene que generar un plan nuevo o re-planificar pasa a estado *Planificando*.

- *Recepción de proveedor*: Cuando el agente recibe información de un proveedor se encuentra en este estado. Una vez recibida la información puede pasar al estado *Procesando información* o *Planificando* si se ha pedido información para un plan de tareas.
- *Petición de proveedor*: Cuando el agente solicita información a un proveedor se encuentra en este estado. Hasta que recibe una respuesta del *agente Proveedor* el agente se queda en espera. El agente puede recibir la información solicitada o una respuesta a petición sin información porque no hay nada nuevo.
- *Registro de proveedor*: Cuando el *agente LcA* incluye en el sistema un nuevo *agente Proveedor* informa al *agente Intérprete* y el *agente Intérprete* tiene que registrar al nuevo *agente Proveedor* y responder al *agente LcA* con petición procesada.
- *Petición de datos*: Cuando el agente solicita información al *agente Contenedor* se encuentra en este estado. Hasta que recibe una respuesta del *agente Contenedor* el agente se queda en espera. El agente puede recibir la información solicitada o una respuesta a la petición sin información porque no existe esa información.
- *Recepción de datos*: Cuando el agente recibe información del *agente Proveedor* se encuentra en este estado. Una vez recibida la información puede pasar al estado *Procesando información* o *Planificando* si se ha pedido información para un plan de tareas.

- **Agente Interface**: 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. Los estados del *agente Interface* son:

- *Registro.* Cuando se inicia el agente se pasa al estado de registro. Cuando un usuario intenta conectarse al 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 *agente Intérprete*.
- *Procesando solicitud.* Se pasa del estado de *espera* a este estado cuando por ejemplo un cuidador o personal médico trata de gestionar su plan de tareas o los parámetros de un paciente. Ocurre lo mismo cuando un usuario (sea del tipo que sea) pide la localización de un paciente. Tras enviar la solicitud al *agente Intérprete*, se vuelve al estado de *espera*.
- *Actualizando interface.* Cuando se recibe un mensaje de otro agente, se pasa del estado de *espera* a este estado, se actualiza la interface con los nuevos datos recibidos (listas de planes de tareas, mapas de localización, parámetros de paciente, tipos de tareas, tipos de perfiles, etc.), se informa al usuario del evento y se vuelve al estado anterior.

- **Agentes Proveedores**: Este tipo agente tiene dos estados principales que son *activo* y *desactivo*. Cuando el agente está *activo* puede estar en *espera* o en *comunicación* al dispositivo asociado:
  - *Espera*: Cuando el *agente Proveedor* está activo y no tiene ninguna petición de otro agente está en *espera*.
  - *En comunicación*: Cuando el *agente Proveedor* recibe una petición pasa a este estado. La petición puede ser de recogida de información relacionada con el dispositivo que gestiona o de transmisión de información hacia el dispositivo, por ejemplo, bloquear un acceso a un usuario.

- **Agente LcA**: Cuando se conecta el administrador a través del *agente Interface* se crea un *agente LcA* asociado. Por tanto el primer estado de este agente es el de inicio cuando se registra el administrador. Una vez iniciado el agente, este puede estar en espera, activando un *agente Proveedor*, desactivando un *agente Proveedor* o registrando un *agente Proveedor*. Cuando el rol *Administrador* termina las gestiones que está realizando el *agente LcA* pasa a estado *finalizado*.

- **Agente Contenedor**: El *agente Contenedor* está en estado Espera hasta que recibe alguna señal del *agente Intérprete*. Los estados del *agente Contenedor* son:
    - *Espera*: Estado en el que está mientras no recibe ninguna petición o envío de información por otros agentes del sistema.
    - *Resolviendo Solicitud*: Se pasa del estado de *espera* a este estado cuando se recibe una información o una petición de información. Dependiendo el tipo de petición se pasa al estado *Almacenando información* o *Buscando información.* Una vez resuelta la petición el agente vuelve a estado *espera*.
    - *Almacenando Información*: Estado en el que el agente se encuentra cuando está almacenando información que recibe del sistema.
    - *Buscando información*: En este estado se encuentra el agente cuando está localizando información que le ha solicitado el *agente Intérprete*.

5.5 Ejemplo de planificación

El objetivo del *agente Intérprete* es planificar de forma dinámica las actividades que el paciente desarrolla a lo largo de un día, de acuerdo a sus preferencias y a las restricciones de tiempo. Se trata de un problema de planificación apropiado para utilizar el método CPM propuesto en el capítulo 3 de este trabajo de investigación. Se trata de un problema dinámico, en el que los pacientes pueden cambiar a menudo de opinión y en el que las restricciones y tratamientos impuestos cambian con frecuencia.

Tabla 5.1 Lista de actividades para paciente EPOC.

Actividad	Nombre	Tiempo (min.)	Tarea
A	Bombona Oxígeno	600	1
B	Levantarse y hacer ejercicio	10	6
C	Desayunar	10	10
D	Pastilla y aerosol de las 8 horas	3	3
E	Visita al médico	60	14
F	Caminar	30	4
G	Almuerzo	20	9
H	Bombona Oxígeno	300	2
I	Pastilla y aerosol de las 14 horas	3	3
J	Comida	40	10
K	Merienda	20	11
L	Caminar	30	5
M	Pastilla y aerosol de las 20 horas	3	3
N	Cena	30	12

En el caso de estudio que se presenta, el *agente Intérprete* recibe a través del *agente Interface* una serie de actividades básicas que debe realizar un paciente EPOC en su hogar. Estas actividades son las creencias (*belief*) del *agente Intérprete*. Esta lista de actividades se encarga de definirla el personal médico y cuidadores que vigilan el estado del paciente. Los tiempos que marca el personal médico para cada una de las tareas es lo que el *agente Intérprete* toma como deseos (*desire*). La Tabla 5.1 muestra las tareas que componen un plan para un paciente EPOC a lo largo de un día normal: una descripción de problema viene formada por el perfil del paciente, por sus preferencias, por sus tratamientos y diagnósticos médicos asociados, por la localización inicial y por el tiempo disponible. La solución está formada por el plan o planes ejecutados para completar la recomendación, su evaluación y su eficiencia. Las soluciones para el *agente Intérprete* son las intenciones (*intention*)

Con la lista de actividades definida y las indicaciones del personal médico el agente intérprete genera una orden para poder construir el plan de tareas siguiendo el método CPM. Con la lista de actividades, la lista de actividades precedentes a cada actividad y la lista de tiempos se determina la duración de la ejecución del plan de tareas y se desarrolla la red del plan de tareas. A continuación la Tabla 5.2 muestra la lista de actividades y sus precedentes.

Tabla 5.2 Actividades y sus precedentes.

Actividad	Precedentes	Tiempo (min.)
A	-	600
B	-	10
C	-	10
D	A	3
E	A, B, C	60
F	E	30
G	F	20
H	G	300
I	G	3
J	I	40
K	J	20
L	H, K	30
M	L	3
N	M	30

A, C, E, F, G, H, L, M y N La red creada a través del método CPM y asociada al plan de tareas anterior, con los nodos numerados y los tiempos más tempranos y más tardíos se muestra en la Figura 5.11.

Según muestra la red de la Figura 5.11, las actividades que se encuentran entre dos nodos donde $ES_i = LS_i$, son tareas críticas y determinan el camino crítico. Así pues las actividades que forman el camino crítico en la red de la Figura 5.11 son A, C, E, F, G, H, L, M y N. El resto de actividades se realizan paralelamente a las anteriores. Por tanto se puede retrasar el comienzo de las actividades no críticas o aumentar su duración en su holgura total, sin aumentar la duración total del plan de tareas. Una vez que el *agente Intérprete* tiene definido el plan de tareas y la duración de dicho plan, traslada esta información al resto de agentes del sistema HoCCAC para controlar y ejecutar el plan. Los *agentes Proveedores* junto con el *agente Interface* son los encargados de controlar el cumplimiento del plan de tareas. Si en algún momento hubiese una interrupción del plan, por ejemplo, porque el paciente sufra un ahogo y tenga que hacer las actividades especificadas para ese caso, el sistema HoCCAC recibe el aviso a través del *agente Interface* e inmediatamente el *agente Intérprete* realiza una re-planificación del plan de tareas. Otra particularidad viene dada por las visitas al médico, ya que son mensuales. Cuando no hay que ir al médico, en la red el nodo representativo de dicha actividad no está presente. Dependiendo del tipo de

interrupción, puede que además el agente proveedor genere un aviso, por ejemplo, en el caso de que falle la bombona de oxígeno.

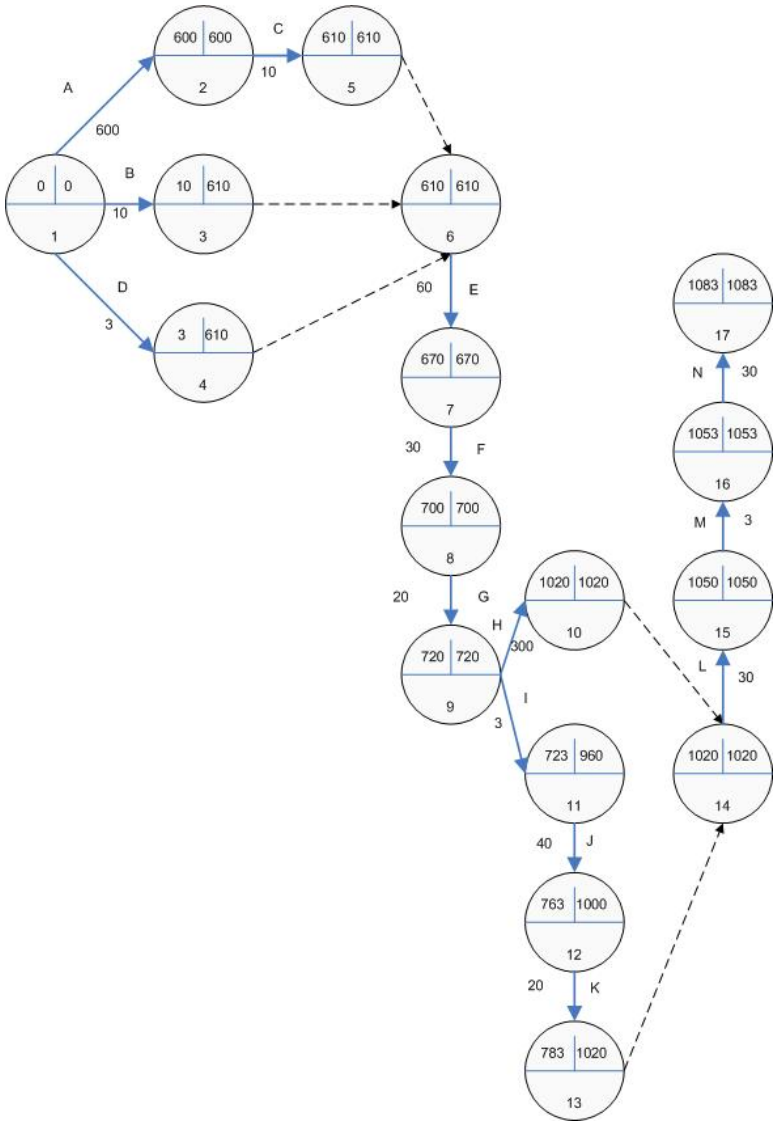


Figura 5.11 Red del plan de tareas definido para el paciente EPOC.

La Figura 5.12 muestra el diagrama de Gantt con la solución asociada al plan de tareas representado en la Figura 5.11. Además en la Figura 5.12 también se deducen las actividades que son críticas y las que no. En rojo están las críticas y en azul las que tienen holgura. También se puede apreciar las tareas que coinciden temporalmente. Así por ejemplo, coinciden las siguientes actividades, la de ponerse la bombona de oxígeno con levantarse, hacer ejercicio y desayunar y también se deduce que la visita al médico tiene lugar tras las actividades anteriores, además de la toma de la medicación de las 8 de la mañana.

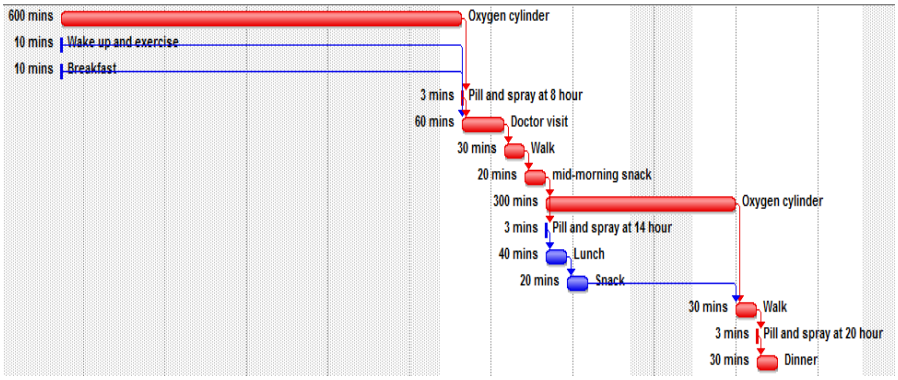


Figura 5.12 Diagrama de Gantt definido para el paciente EPOC.

Durante toda la sucesión de operaciones realizadas por los agentes del sistema hay que tener en cuenta la transformación de información que se produce en el sistema. Por un lado, se recogen datos de bajo nivel del entorno del paciente que posteriormente son almacenados en el sistema de información como datos de alto nivel, para que sea más rápida su interpretación y también sea más fácil su uso. Los *agentes Proveedores* de información en colaboración con el *agente Intérprete* y el *agente Contendor* son los encargados de realizar esta labor. Además el paciente en todo momento puede interactuar con el contexto para establecer los parámetros que rigen el funcionamiento del sistema a través del *agente Interface*.

5.6 Conclusiones y resultados obtenidos

Los sistemas sensibles al contexto utilizados para monitorizar y supervisar a personas dependientes en el hogar son una alternativa a los cuidados en el hogar tradicionales [Fraile, *et al.,* 2009b]. La gestión de un hogar inteligente puede verse como un problema dinámico en el que los pacientes o usuarios solicitan servicios, el hogar (contexto) proporciona información y los cuidadores, personal médico y visitantes tratan de mejorar la calidad de vida del paciente de la forma menos intrusiva posible. En la sociedad de la información actual, los usuarios demandan servicios personalizados y adaptados a las nuevas tecnologías. Las casas actuales no son una excepción. Dado que una casa es el lugar donde una persona, y más aún una persona dependiente pasa mucho tiempo de su vida, este (la casa) tiene que ser un contexto en el que los usuarios no encuentren problemas para realizar sus tareas habituales o deseadas. Además, una casa es un contexto en el que en muchas ocasiones resulta muy complicado prestar una atención adecuada y personalizada a las personas dependientes. Hoy en día la mayor parte de los usuarios en un hogar (pacientes, cuidadores, personal médico, visitantes) disponen de dispositivos móviles que permiten ofrecer servicios personalizados y de alta calidad. Estos dispositivos permiten que un usuario pueda integrarse fácilmente dentro de un entorno inteligente, obteniendo un acceso inalámbrico que le proporcione acceso a los servicios de un hogar independientemente de la localización [Fraile, *et al.,* 2010d].

Sin embargo, los dispositivos móviles y otros dispositivos como sensores o actuadores, por sí mismos no son capaces de proporcionar todos los elementos necesarios para interactuar dentro de un sistema. Además, entre un gran conjunto de dispositivos de diferentes fabricantes, en muchas ocasiones es difícil encontrar compatibilidades entre ellos. Por estas razones se hace conveniente desarrollar una arquitectura que facilite la compatibilidad y la homogeneidad. Tal y como se ha explicado en el capítulo 4 de este trabajo de investigación, se ha optado por crear una arquitectura multiagente, que proporciona características de gran valor añadido, como son la autonomía, la inteligencia, la capacidad de razonamiento, la capacidad de planificación y la capacidad de adaptación [Bajo, *et al.,* 2010]. Además los agentes siguen modelos organizativos y sociales, imitando en general los comportamientos

humanos. Estas características hacen que los agentes se adapten mejor a las necesidades, a los comportamientos de los humanos y por tanto a las personas dependientes en el hogar [Corchado, *et al.,* 2011].

La arquitectura HoCCAC ha sido utilizada para construir el sistema multiagente que se describe a lo largo de este capítulo. La arquitectura propone una metodología de análisis y diseño que permite modelar el problema en términos de agentes y de entornos sensibles al contexto. Se comienza por realizar un estudio de los requisitos del problema, a partir de los casos de uso obtenidos. A diferencia de otras metodologías, se hace una especial incidencia en el modelado del entorno, identificando todos los elementos del entorno que interactúan con el sistema. A continuación se obtiene un análisis en términos sociales y de organización. La etapa intermedia establecida permite obtener el diseño. Para ello se utiliza SysML que proporciona un paso directo entre el modelo abstracto del sistema y su implementación.

Figura 5.13 Gráfico de rendimiento asociada al plan de tareas presentado en el caso de estudio.

Por otro lado, el mecanismo de planificación CPM resulta ser muy adecuado y sencillo de implantar, para su aplicación en entornos dinámicos y cambiantes, en los que los sistemas informáticos requieren la incorporación de mecanismos que permitan

obtener altas capacidades de aprendizaje y adaptación. El sistema de planificación propuesto difiere de otros existentes en que optimiza la posibilidad de asignación de nuevos planes y de re-planificación dinámica en tiempo de ejecución. Para obtener resultados y datos en la evaluación del método CPM en el caso de estudio planteado se elabora una gráfica (Figura 5.13) de rendimiento que va a servir para observar el ritmo al que se desarrollan las actividades descritas en el caso de estudio. La Figura 5.13 muestra la evaluación realizada al plan desarrollado por el *agente Intérprete* en el caso de estudio. El gráfico de la Figura 5.13 permite conocer las actividades que van finalizando. En el gráfico de la Figura 5.13 el eje de ordenadas (Y) presenta una escala de porcentajes de cumplimiento del plan de tareas y en el eje de abscisas (X) las horas de duración del plan de tareas definido. En el caso de estudio que se presenta el plan de tareas tiene una duración de 18 horas y 5 minutos. Esta duración se calcula sumando los tiempos asociados a las tareas que forman el camino crítico. En esta gráfica se señala el final del plan de tareas que se encuentra en la línea del 100% de eficiencia y la coordenada del tiempo final del plan de acciones como muestra la Figura 5.13.

Tabla 5.3 Informe de avance de las actividades del plan de tareas.

Hora	Actividad	% avance	Hora	Actividad	% avance
1	A	10%	**12**	G	T
1	B	T	**12**	H	3%
1	C	T	**12**	I	T
2	A	20%	**13**	H	23%
3	A	30%	**13**	J	T
4	A	40%	**14**	H	43%
5	A	50%	**15**	H	63%
6	A	60%	**15**	K	T
7	A	70%	**16**	H	83%
8	A	80%	**17**	H	T
9	A	90%	**17**	L	5%
10	A	T	**18**	L	T
11	D	T	**18**	M	T
11	E	95%	**18**	N	98%
12	E	T	**19**	N	T
12	F	T			

Ahora ya se puede calcular el avance logrado en el proyecto cada hora como muestra la Figura 5.13. El avance del plan de tareas es el avance de la suma de los avances logrados por cada una de las actividades del plan. En la Tabla 5.3 aparecen los informes de avance de cada una de las actividades en el caso de estudio definido. Los datos de la Tabla 5.3 los va rellenando el agente intérprete de la siguiente forma:

- En el momento de recibir la información de avance real del plan de tareas:
  - Se anota la hora en la que se recibe la información (columna *Hora*).
  - En la columna *Actividad* se anota la actividad realizada durante esa hora.
  - Finalmente en la columna *"% avance*" se muestra los avances de la actividad durante esa hora. Si en esta columna aparece una 'T' significa que la actividad ha terminado.

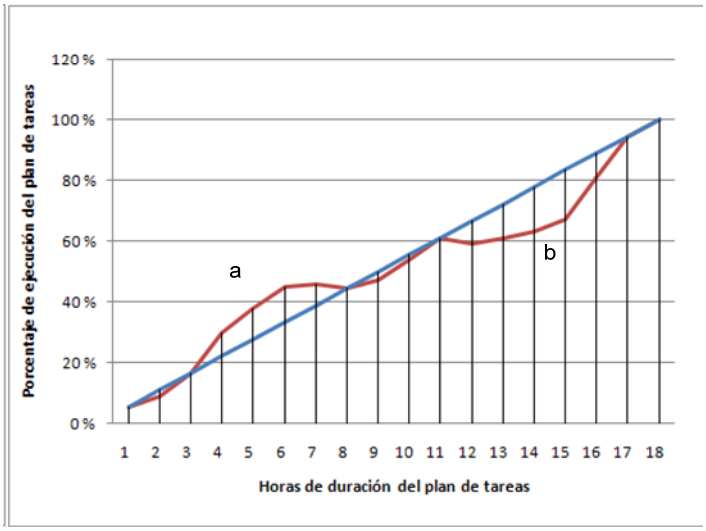


Figura 5.14 Gráfico de rendimiento asociada al plan de tareas con aceleraciones (a) y retardos (b) en las actividades.

A partir de los datos definidos en Tabla 5.3 se puede comprobar en todo momento el avance del plan de tareas sumando los avances parciales logrados por las actividades. Esta suma representa el avance real del plan de tareas. Es importante resaltar que para el caso de estudio llevado a cabo los intervalos temporales que se tienen en cuenta son de 1 hora porque es un plan de tareas diario, pero puede haber planes de tareas más concretos y cortos en los que los intervalos temporales sean

minutos. La gráfica que se presenta en la Figura 5.13 representa una línea completamente recta en diagonal porque durante la ejecución del plan de actividades del caso de estudio no se presenta ningún retardo o aceleración en ninguna actividad. Cuando el plan de tareas tiene aceleraciones o retardos la gráfica resultante no se representa como una línea recta sino como una línea con arcos como muestra la Figura 5.14. La Figura 5.14 muestra dos líneas, la línea azul muestra la evaluación de un plan de tareas ejecutado sin retardos y la línea roja muestra la evaluación de un plan de tareas ejecutado con aceleraciones y retardos. En la gráfica de la Figura 5.14 se puede ver que hay una aceleración-retardo en el progreso de las tareas en la zona (a) y también hay un retardo-aceleración en el progreso de las tareas en la zona (b) con respecto a una ejecución correcta del plan de tareas.

Los nuevos servicios ofrecidos por HoCCAC, como la planificación de tareas y el razonamiento, permiten una interacción con el usuario más cercana y natural y, a la vez, implícita. Al mismo tiempo el sistema ofrece al paciente planes de tareas proactivos que tratan de mejorar su calidad de vida. El usuario puede realizar sus tareas cotidianas y recibir apoyo del contexto inteligente sin necesidad de una interacción explícita. Por tanto, el usuario no tiene necesidad de aprender a utilizar el sistema. Esto hace que el grado de satisfacción del usuario con el sistema HoCCAC que gestiona el entorno, aumente. Además el sistema HoCCAC evalúa el uso de recursos en el entorno emitiendo informes muy útiles al personal médico. La información sobre el uso de recursos la proporcionan los agentes proveedores, al sistema HoCCAC y el *agente Interprete* se encarga de evaluar la información. El *agente Intérprete* proporciona con esta evaluación, la utilización de recursos día a día asociados a las actividades que solicitan su uso. El *agente Intérprete* también examina sus propias evaluaciones y puede reasignar actividades a recursos libres para acortar la duración de un plan de tareas. Así se consigue que el sistema HoCCAC mejore las soluciones ofrecidas de forma progresiva en el tiempo.

La Tabla 5.4 refleja de forma general las diferencias que existe entre la arquitectura HoCCAC y otros sistemas sensibles al contexto a través de las funcionalidades y tecnologías soportados. Las funcionalidades y tecnologías que se consideran importantes a la hora de valorar un sistema con respecto a HoCCAC son:

- Localización e Identificación: Esta funcionalidad proporciona en un sistema mecanismos para localización e identificación de usuarios u objetos.
- Gestión de Alertas: Esta funcionalidad incluye un servicio de alertas que informa a los usuarios en tiempo de ejecución sobre anomalías en el sistema.
- Tecnología móvil: Esta tecnología proporciona al sistema compatibilidad con dispositivos móviles y hace posible acceder al sistema de forma transparente a través de dispositivos móviles.
- Cuidados en el hogar: Esta funcionalidad proporciona al sistema servicios para aportar cuidados médicos al paciente en su hogar.
- Tecnología de Agentes: Esta tecnología proporciona al sistema un modelo inicial basado en la Tecnología de Agentes.

Tabla 5.4 Factores de comparación que relacionan funcionalidades y tecnología con sistemas sensibles al contexto.

	Localización e Identificación	**Gestión de Alertas**	**Tecnología móvil**	**Cuidados en el hogar**	**Tecnología de agentes**
Context Toolkit [Dey et al., 2009]	SI	---	---	---	---
Context Managing Framework [Koskinen et al., 2006]	SI	---	---	---	---
Cortex [Biegel et al., 2004]	SI	---	---	---	---
COBRA [Chen et al., 2004]	SI	---	---	---	SI
CAMUS [Moodn et al., 2006]	SI	---	---	---	SI
Citron [Yamabe et al., 2005]	SI	---	---	---	---
RoboCare [RoboCare, 2005]	SI	---	---	SI	SI
TeleCARE [Camarinha-Matos, et al., 2008]	SI	---	SI	SI	SI
Smart Home Technology [Angulo, et al., 2004]	SI	---	SI	SI	SI
ALZ-MAS [Corchado et al., 2008a]	SI	---	SI	---	SI
HoCCAC	**SI**	**SI**	**SI**	**SI**	**SI**

Como se puede ver en la Tabla 5.4 todos los sistemas sensibles al contexto incluyen la funcionalidad de localización e identificación y con el paso del tiempo los sistemas que han ido apareciendo van añadiendo nuevas funcionalidades y tecnologías. *TeleCare*, *Smart Home Technology, ALZ-MAS y* HoCCAC hacen uso de la tecnología móvil, todos ellos a través de agentes móviles instalados en dispositivos móviles. Por otro lado, son pocos los sistemas que se centran en ofrecer servicios de cuidados en el hogar (*RoboCare, TeleCare*, *Smart Home Technology y* HoCCAC). Además, de los sistemas enumerados, son pocos los sistemas que no se basan en agentes (*Context Toolkit, Context Managing Framework, Cortex y Citron*). Se puede destacar que solamente HoCCAC incluye gestión de alertas además del resto de funcionalidades y tecnologías que se enumeran.

La Tabla 5.5 intenta reflejar de forma general la diferencia que existe entre la arquitectura HoCCAC y otros modelos de arquitecturas distribuidas a través de características muy concretas y generalmente relacionadas con la computación sensible al contexto. Estas características de sistemas sensibles al contexto son:

- Captación de contexto: Esta característica proporciona a los sistemas técnicas para recoger información del entorno, generalmente a través de sensores y también a través de aplicaciones o dispositivos de recogida de información.
- Modelado de información: Los sistemas que incluyen esta característica destacan por disponer de herramientas y modelos para gestionar y tratar la información recogida del entorno.
- Dispersión de información: Esta característica hace que los sistemas puedan modificar y actuar sobre el entorno, generalmente a través de actuadores y dispositivos de control automático, basándose en la información que poseen.
- Razonamiento de información: Esta característica la tienen sistemas que hacen uso de métodos de razonamiento para ofrecer resultados.
- Planificación de tareas: Esta característica la poseen sistemas que utilizan métodos de planificación para generar planes de acciones basados en uno o varios objetivos.

Tabla 5.5 Factores de comparación específicos de sistemas sensibles al contexto.

	Captación de contexto	Modelado de Información	Dispersión de Información	Razonamiento	Planificación de tareas
Context Toolkit [Dey et al., 2009]	SI	---	---	---	---
Context Managing Framework[Koskinen et al., 2006]	SI	SI	---	SI	---
Cortex [Biegel et al., 2004]	SI	SI	SI	---	---
COBRA [Chen et al., 2004]	SI	SI	SI	SI	---
CAMUS [Moon et al., 2006]	SI	SI	SI	SI	---
Citron [Yamabe et al., 2005]	SI	SI	SI	---	---
RoboCare [RoboCare, 2005]	SI	---	---	---	---
TeleCARE [Camarinha-Matos, et al., 2008]	SI	---	---	---	---
Smart Home Technology [Angulo, et al., 2004]	SI	SI	---	---	---
ALZ-MAS [Corchado et al., 2008a]	SI	---	---	SI	SI
HoCCAC	**SI**	**SI**	**SI**	**SI**	**SI**

Como se puede ver en la Tabla 5.5 todos los sistemas enumerados realizan tareas de captación de contexto. Pero por otro lado son pocos los sistemas que utilizan técnicas de dispersión o actuación sobre el contexto (Cortex, COBRA, CAMUS, Citron y HoCCAC) y menos aún los que utilizan métodos de razonamiento (*Context Managing Framework, COBRA, CAMUS y* HoCCAC). Además, también se puede destacar, que sólo HoCCAC utiliza métodos de planificación, además de incluir el resto de características enumeradas, para procesar la información contextual y poder aportar planes de acciones óptimos que se ejecutan en el menor tiempo posible.

Por otra parte, la tecnología utilizada no condiciona los servicios ofertados. La apuesta por RFID y NFC para localización e identificación de objetos y usuarios es independiente de la implementación concreta de la tecnología, es decir, los servicios

son válidos para cualquier tipo de sensores, antenas, lectores y etiquetas RFID. Así se consigue que el sistema sea válido para entornos muy variados y cuyos dispositivos no tienen que tener una base tecnológica fija. La incorporación de nuevos sensores RFID y NFC en el caso de estudio mejora significativamente una serie de funciones como el control de la medicación, el control de la alimentación y la detección de anomalías en el comportamiento del paciente, como se muestra en la Tabla 5.6.

Tabla 5.6 Comparación de variables de control del caso de estudio antes y después de implantar la arquitectura HoCCAC.

	Control de Medicación	**Control de comidas**	**Detectar anomalías en el comportamiento del paciente**
Antes de HoCCAC	92%	88%	46%
Después de HoCCAC	100%	100%	95%

La tabla 5.6 muestra el porcentaje de éxito en el control de tres variables asociadas al caso de estudio. Es fácil apreciar la eficacia del sistema HoCCAC en el control de la ingesta de medicamentos y alimentos de cada paciente con una tasa de eficiencia del 100%. El sistema permite evitar errores en la administración y control de la medicación y las dietas especiales para los pacientes. Además, el sistema HoCCAC permite detectar comportamientos anómalos en los pacientes y crear patrones de comportamiento para situaciones de riesgo. El *agente Intérprete* tiene en cuenta experiencias pasadas con el fin de organizar los nuevos planes de tareas en el sistema. En este sentido, el sistema HoCCAC se aprovecha de las experiencias pasadas y aprende de las mismas. Este comportamiento proporciona una gran capacidad de adaptación, como demuestran los resultados presentados en la Tabla 5.6.

Aunque todavía queda mucho trabajo por hacer, el prototipo del sistema desarrollado mejora la seguridad en el hogar de las personas dependientes por medio de dispositivos de vigilancia y alerta. El propio *agente Intérprete* en HoCCAC es capaz de reaccionar automáticamente ante situaciones de peligro o emergencia realizando re-planificaciones de los planes en ejecución y lanzando mensajes de alerta al sistema. Como resultado HoCCAC crea un entorno *context-aware* que facilita el

desarrollo de sistemas inteligentes distribuidos y presta servicios a personas dependientes en el hogar. De esta forma es posible automatizar ciertas tareas de monitorización de personas dependientes y mejorar su calidad de vida. La combinación de sistemas multiagente y de tecnologías inalámbricas proporciona un alto nivel de interacción entre el contexto, cuidadores y pacientes. Además, el uso de los dispositivos móviles facilita las interacciones sociales y la transferencia de conocimientos. Los resultados obtenidos han sido satisfactorios, demostrando que la arquitectura multiagente HoCCAC permite modelar e implementar sistemas basados en la sensibilidad contextual de forma eficiente. Asimismo, se ha analizado el impacto de un sistema de este tipo en un entorno simulado a partir de un prototipo, comprobando que optimiza la oferta de servicios tradicionales y aumenta su calidad.

Tabla 5.7 Puntos fuertes y débiles de la arquitectura HoCCAC.

Puntos Fuertes	Puntos Débiles
El sistema facilita la creación de entornos inteligentes basados en agentes	
Los agentes actúan como una etapa intermedia entre los usuarios y la tecnología.	
El sistema se aproxima al comportamiento humano.	Falta incorporar nuevos mecanismos de razonamiento y adaptación al contexto.
El sistema facilita la incorporación de nuevos dispositivos para captura de información del contexto y la interacción inteligente entre los dispositivos del contexto y los usuarios.	Falta automatizar aún más la incorporación de cualquier tipo de dispositivo.
El sistema facilita la captura de distintos tipos de información de manera no intrusiva en un contexto variado.	Falta simplificar y optimizar aún más el sistema de información.
La metodología de análisis y diseño utilizada permite estudiar el entorno y alcanzar un alto nivel de detalle.	Falta la construcción de una herramienta automatizada de desarrollo utilizando la metodología SysML e incluir las etapas de implementación.
El método de planificación y los mecanismos de razonamiento utilizados hacen que el sistema tenga una gran capacidad de aprendizaje y de adaptación a cambios.	No se evalúa el planificador con otros métodos de planificación, tampoco se evalúan otros mecanismos de razonamiento.
El sistema facilita la monitorización y cuidados a personas dependientes en el hogar personalizando los servicios ofrecidos al paciente.	Falta probar el sistema desarrollado en contextos reales.

Con todo lo descrito anteriormente se pueden destacar de forma resumida los puntos fuertes y puntos débiles apreciados para la arquitectura propuesta. La tabla 5.7 muestra una comparativa entre puntos fuertes y puntos débiles de la arquitectura HoCCAC.

En el siguiente capítulo se analizan cada uno de los objetivos alcanzados con el caso de estudio analizado en este capítulo. Además se presentan las conclusiones a las que se ha llegado tras la realización de este trabajo y se comentan las líneas de trabajo futuro que se abren tras las conclusiones obtenidas.

6

Conclusiones y trabajo futuro

6.1 Conclusiones

Hoy en día, los sistemas sensibles al contexto integran, cada vez más, múltiples tecnologías de comunicaciones e identificación, capaces de interaccionar con infraestructuras de comunicaciones que extienden sus coberturas con rapidez (Movilidad, WiFi, RFID, NFC etc.) y con otros dispositivos de su entorno. Esta situación sugiere un futuro en el que dichas tecnologías (y sus evoluciones) se empleen coordinadamente, para, entre otros fines, adquirir información contextual, que inspire el diseño de la siguiente generación de sistemas sensibles al contexto, centrados en el usuario, adaptables e inteligentes.

El elemento diferencial de este trabajo de tesis es la adquisición y poder de decisión sobre el contexto para configurar soluciones adaptativas, que pivoten alrededor de las necesidades específicas de los usuarios, y que tengan en cuenta las capacidades de los dispositivos y de los recursos del entorno. La arquitectura multiagente propuesta HoCCAC se basa en los modelos tradicionales de sistemas

sensibles al contexto y se adapta y amplía en conceptos y funciones. Para la adquisición de contexto HoCCAC trata de unificar y simplificar la captura de información contextual a partir de *agentes Proveedores* de información. Para la toma de decisiones e interpretación de la información contextual HoCCAC integra mecanismos de planificación y razonamiento sobre un agente inteligente, el *agente Intérprete*. En la arquitectura propuesta, el contexto se trata como simplificación de la realidad, más o menos compleja en función de la misión del sistema desarrollado con la arquitectura. Sobre el sistema sensible al contexto desarrollado recae la responsabilidad de definir qué información de contexto necesita para conseguir un funcionamiento adecuado a sus expectativas. La información de contexto es dinámica, se configura a lo largo del tiempo en las características que la componen, y se instancia cuando se requiere por el *agente Intérprete* (bajo demanda o proactivamente, ante la detección de cambios en el entorno). El objetivo es minimizar los recursos necesarios para la construcción y posterior instanciación de la información de contexto, ya que en los espacios del futuro puede darse el caso de que los sistemas sensibles al contexto estén interaccionando de forma autónoma con varias entidades o usuarios simultáneamente, de la misma manera que una entidad o usuario concreto puede estar interactuando con diversos sistemas sensibles al contexto a la vez. Hasta la fecha, no se conoce una propuesta metodológica para abordar el proceso de desarrollo de sistemas sensibles al contexto, ni de los sistemas de gestión de la información subyacentes. Con la arquitectura HoCCAC se proporciona un marco de análisis y diseño funcional que puede servir de gran utilidad a la hora de acometer dicho proceso de desarrollo.

Por otro lado HoCCAC presenta una serie de características que facilitan y optimizan el desarrollo de sistemas multiagente distribuidos basados en la computación sensible al contexto. Una de estas características es la 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 (Smartphone, micro-controladores, etc.). Otra característica es la 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. Del mismo modo, las funcionalidades pueden ser replicadas para así obtener una mayor disponibilidad de recursos. HoCCAC también se caracteriza por su robustez. El enfoque distribuido de la arquitectura propuesta permite inicializar y detener aplicaciones, servicios o agentes, de forma independiente y sin afectar al resto de componentes del sistema.

La arquitectura HoCCAC se ha aplicado para desarrollar un sistema multiagente dinámico y distribuido enfocado a mejorar el cuidado y control de pacientes EPOC en su hogar [Fraile *et al.*, 2009c]. La arquitectura desarrollada integra un conjunto de tecnologías como las redes inalámbricas, la identificación por radio frecuencia y los agentes inteligentes, basándose en conceptos derivados de la computación sensible al contexto para crear la estructura de un sistema capaz de adaptarse a las necesidades del contexto de usuario, proporcionar una interacción usuario-sistema de manera no invasiva y permitir una comunicación y acceso tanto a la información como a los servicios, de forma ubicua, inalámbrica y transparente para el usuario. Se ha desarrollado un agente inteligente (el *agente Intérprete*) basado en el modelo BDI y que integra el método de razonamiento CBR y el método de planificación CPM. El *agente Intérprete* se integra junto con otros agentes complementarios en un sistema multiagente [Fraile *et al.*, 2010a]. El sistema multiagente es capaz de interactuar con el entorno utilizando principalmente tecnología de comunicación inalámbrica y tecnología RFID/NFC. El *agente Intérprete* ha sido diseñado con los objetivos de razonar la información contextual y planificar las tareas cotidianas del paciente de forma automática, mantener continuos informes sobre las actividades desarrolladas por el paciente y garantizar que los pacientes reciben los cuidados adecuados. El sistema multiagente desarrollado se completa con agentes proveedores de información, que permiten optimizar la captura de información de forma automática; con agentes LcA, encargados de administrar los agentes proveedores existentes y agregar nuevos *agentes Proveedores* de información dinámicamente; con un *agente Contenedor*, encargado de manejar el modelo de datos del sistema y organizar la información que recibe y le solicita el *agente Intérprete*; y con un *agente Interface*, encargado de actuar como componente de interacción con los usuarios del sistema.

Las funcionalidades que ofrece la plataforma se dividen en dos grandes grupos, locales, que tienen sentido en el entorno del hogar y cuyo consumidor habitual es el paciente, y remotas, que proporcionan información de lo que sucede en la casa hacia fuera de la misma. En la Tabla 6.1 se muestra un listado de las funcionalidades por categoría definida. Así pues, estas funcionalidades se pueden agrupar en las siguientes categorías:

- **Detección de emergencias.** La situación de emergencia puede ser originada por situaciones anómalas de salud o detección de alteraciones en el ambiente (p.e. escape de gas, elevado grado de humedad, etc.). El sistema multiagente detecta las emergencias y genera alertas que son recibidas en la interfaz de monitorización y en los dispositivos de los familiares asociados.
- **Salud.** Respecto al cuidado de la salud, el sistema permite mejorar el seguimiento de tratamientos farmacológicos (p.e. en pacientes EPOC control de toma de medicación, y aerosoles) y simplificar la monitorización diagnóstica, p.e. enviando información del estado del paciente en tiempo de ejecución y generando históricos que pueden ser consultados por el personal médico.
- **Seguridad en el hogar.** La seguridad física es uno de los aspectos que más preocupan a pacientes crónicos en su hogar. Aunque no se ha implementado una funcionalidad compleja de seguridad, el sistema envía notificaciones al paciente cuando se abre la puerta de entrada, con el fin de alertarle y permitirle telefonear a servicios de seguridad o familiares.
- **Confort y bienestar.** Esta categoría agrupa servicios dispares, entre los que se encuentran los de control y automatización de las condiciones ambientales, búsqueda de objetos o notificaciones prácticas, relacionadas con la gestión domótica o el tiempo. La monitorización de parámetros ambientales permite activar automáticamente (o sugerir al usuario la activación) de los sistemas de calefacción o aire acondicionado. Por otra parte, el servicio de búsqueda de objetos facilita encontrar cosas que se pueden "perder" fácilmente dentro de la casa (p.e. llaves, gafas, un bastón, etc.). La recepción de alertas a la hora de salir facilita que se apaguen las luces automáticamente, el usuario no olvide su paraguas o el paciente se acuerde de coger las medicinas de la tarde.

- **Vida social.** Evitar el aislamiento y mejorar la actividad social del paciente es un objetivo del sistema desarrollado. El sistema propuesto establece un sistema de recordatorio de citas y eventos (puede, por ejemplo, sugerir al usuario que llame a un familiar que cumple años) y hace propuestas de planes de actividad y eventos.

Tabla 6.1 Listado de funcionalidades del sistema multiagente desarrollado.

Paciente	Personal médico/ Familiares	Funcionalidad
		DETECCIÓN DE EMERGENCIAS
	X	Detección y notificación de anomalías en los tratamientos médicos.
X	X	Detección y notificación de anormalidades en los planes de cuidados.
X	X	Detección y notificación de irregularidades ambientales (p.e., escapes de gas, inundación, etc.)
		SERVICIOS DE SALUD
	X	Recepción de señal para diagnóstico remoto. Por prescripción médica, el sistema puede tener que enviar datos personales para su análisis.
X	X	Envío de alertas relacionadas con la variación de la actividad física (p.e. recomendación de disminuir el ritmo de la actividad física en caso de sobreesfuerzo por una modificación en la planificación inicial).
X		Ayuda al seguimiento de tratamientos farmacológicos (notificaciones contextuales que incluyan consejos sobre la toma de medicación: "con un vaso de agua", "come algo", "no tomes el sol", etc.)
	X	Monitorización e históricos de ingesta de medicinas, que incluyan el horario establecido, los avisos enviados y el momento en el que se estima que se ha tomado la medicación, si lo ha hecho (para valorar adecuación y cumplimiento del tratamiento).
X	X	Control de peso y recomendaciones "contextuales" al respecto (recomendaciones nutricionales a la hora de comer, por ejemplo).
		SEGURIDAD EN EL HOGAR
X		Notificaciones anti-intruso (cuando alguien entra en la casa, el paciente recibe una alerta y puede iniciar una llamada en caso de que presienta riesgo).
	X	Detección de apertura de puerta en horario nocturno (si el paciente está en su habitación y se detecta una apertura de puerta/ventana, el sistema genera un aviso a la central de monitorización).
		CONFORT Y BIENESTAR
X		Puesta en marcha automática de los aparatos de aire acondicionado o calefacción cuando se sobrepasen unos límites establecidos de temperatura y humedad.
	X	Monitorización de las condiciones ambientales (generación de logs) para detectar si el paciente soporta temperaturas perjudiciales para él.

Paciente	Personal médico/ Familiares	Funcionalidad
		CONFORT Y BIENESTAR
X		Notificación de tareas antes de salir, como coger medicinas, no olvidar el paraguas si se supone que va a llover o apagar automáticamente la luz de la habitación.
X		Búsqueda de objetos perdidos. Algunos objetos cotidianos (gafas, bastones, llaves) pueden llevar una etiqueta RFID/NFC que haga posible localizarlos.
		VIDA SOCIAL
X		Recordatorio de citas y eventos.
X		Sugerencias contextuales de planes de actividad (llamar a un familiar, asistir a algún evento, jugar con el ordenador, etc.)

Los resultados obtenidos [Fraile *et al.*, 2009b] [Fraile *et al.*, 2009c] muestran que el sistema multiagente desarrollado permite mejorar la realización de las tareas cotidianas del paciente, reducir el tiempo empleado por el personal médico en la realización de tareas indirectas, tales como la monitorización de pacientes, sin que la calidad de la atención prestada se vea afectada. Además el sistema multiagente permite obtener una gran cantidad de información acerca de los pacientes y del contexto. Esta información es analizada por el agente intérprete, para obtener conocimientos que faciliten la toma de decisiones estratégicas. Por último, el sistema multiagente permite incrementar la seguridad del paciente en el hogar en tres aspectos: Se monitoriza al paciente y garantiza que se encuentre en localizaciones válidas; los pacientes reciben planes de tareas óptimos ante situaciones anómalas; sólo el personal autorizado tiene acceso al hogar; y la información se almacena de forma segura.

6.2 Contribuciones de la investigación

Este trabajo hace nuevas contribuciones en el ámbito de la teoría de agentes y de la computación sensible al contexto y propone una arquitectura (HoCCAC) que facilita el desarrollo de sistemas sensibles al contexto. Los sistemas sensibles al contexto surgen con el objetivo de mejorar el rendimiento de los sistemas de personalización tradicionales. El principal objetivo de los sistemas sensibles al contexto es ofrecer

servicios basados en la información de contexto. HoCCAC da un paso más y aspira a lograr espacios inteligentes ubicuos, capaces de cubrir gran parte de las necesidades que plantean las acciones cotidianas de los usuarios. A continuación se resumen las aportaciones realizadas por este trabajo de investigación:

- Proporciona una amplia revisión de sistemas sensibles al contexto desarrollados con objetivos muy variados y que se relacionan con otras áreas de investigación. Por este motivo, este trabajo de investigación también proporciona un análisis de los beneficios que aportan los sistemas sensibles al contexto al relacionarse con áreas como la Inteligencia Ambiental, la domótica y las comunicaciones inalámbricas.
- Se ha propuesto una arquitectura (HoCCAC) multiagente que facilita la construcción de entornos inteligentes por medio de la utilización de agentes inteligentes. La arquitectura se centra en proporcionar elementos de computación ubicua e interfaces inteligentes. La arquitectura ha sido utilizada para crear un entorno inteligente en el hogar de una persona dependiente y se han evaluado positivamente sus capacidades para modelar problemas de computación sensible al contexto, para ofrecer mecanismos de aprendizaje, mecanismos de planificación y acoplamiento en entornos dinámicos, para adaptarse a las tecnologías Wi-Fi, ZigBee y RFID/NFC. Dado que existen muchos entornos con características parecidas al caso de estudio presentado, es posible pensar que la arquitectura es válida para ser aplicada en una gran cantidad de entornos similares.
- La arquitectura propuesta incluye el método de razonamiento CBR y lo combina con el método de planificación CPM para integrar ambos métodos en el agente intérprete, basado en el modelo BDI y así generar planes efectivos de tareas. Esto hace del agente intérprete un agente inteligente, ya que aporta soluciones proactivas a través de la interpretación de la información contextual. El método de razonamiento CBR es suficiente para el desarrollo de sistemas que controlan un número finito de estados e imágenes de contexto. Además, permite la inclusión sencilla de nuevas reglas, que pueden configurarse por alguien sin conocimientos específicos. El método de planificación CPM por su parte, genera

planes de actividades eficientes en base a las tareas rutinarias que debe realizar un usuario. Las características de los métodos de planificación y razonamiento integrados en el *agente Intérprete* hacen que el *agente Intérprete* de HoCCAC sea único en su definición y características.

- La integración de nuevos sensores de adquisición de contexto es habitualmente un proceso que consume tiempo y recursos, incluso si éste (hardware o software) dispone de interfaces para programación. HoCCAC, a través del agente LcA y los agentes Proveedores, integra las peculiaridades de cada dispositivo, los sistemas de control diferentes asociados a cada dispositivo y además, en muchas ocasiones, tiene en cuenta el consumo de recursos que el envío de información periódica puede suponer para el dispositivo para intentar minimizarlo (p.e. desactivando automáticamente el dispositivo cuando no sea necesaria la información que dicho dispositivo proporciona).
- Se crea un marco para el análisis y diseño de entornos inteligentes. Se realiza una revisión de los conceptos fundamentales de la computación sensible al contexto, de las tecnologías utilizadas en el desarrollo de entornos inteligentes y de trabajos previos de sistemas multiagentes aplicados en el desarrollo de sistemas sensibles al contexto. Se proporcionan mecanismos que facilitan la incorporación de las características de comunicación ubicua, computación ubicua e interfaces inteligentes a los sistemas multiagente.
- El sistema multiagente desarrollado con la arquitectura HoCCAC, a través de sus agentes aporta funcionalidades para funcionar con distintos niveles de colaboración de los usuarios. HoCCAC tiene esto en cuenta, ya que cuando se proporcionan funcionalidades que requieren la colaboración o aportación de información por parte del usuario, es necesario contar con que los individuos pueden comportarse de forma diferente, y el sistema debe responder independientemente de cuál sea la voluntad de interaccionar del usuario. Además HoCCAC permite la utilización del contexto como filtro de información, la "contextualización" de funcionalidades ya existentes (pe. asistencia médica remota) y el apoyo a tareas cotidianas (como es el caso del sistema-recordatorio de toma de medicación).

- La arquitectura propuesta combina el conocimiento del contexto con distintas formas de interacción con el entorno para ofrecer nuevas posibilidades. La disponibilidad de tecnologías como NFC en el móvil hace posible lograr más automatización en el lanzamiento de servicios. Dicha tecnología puede tener un interés especial para aplicaciones como el prototipo presentado en el caso de estudio, en el que la facilidad de interacción con el entorno y la facilidad de aprendizaje del uso de la tecnología representa un valor añadido al sistema.
- Durante el proceso de desarrollo de este trabajo de 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[5] y la asistencia a conferencias, congresos, *workshops*, etc.

6.3 Trabajo futuro

El final de este trabajo de investigación no es más que el comienzo de líneas de trabajo que deben permitir poner en práctica gran parte de los conceptos comentados hasta este punto y avanzar en la interconexión de las tecnologías y componentes hardware y software en cualquier contexto, además de mejorar el problema de la construcción de un contexto inteligente y eficiente. Van a continuación las principales líneas de acción que se sugieren, a sabiendas de que quedan muchas otras probablemente igual de válidas e interesantes:

- Se propone profundizar en técnicas de aprendizaje, razonamiento y planificación que permitan no tener que configurar necesariamente todos los patrones del sistema multiagente en la fase de diseño, sino que permitan su creación, calibrado, adaptación, desglose y eliminación en tiempo de operación. Es decir, mejorar las capacidades del *agente Intérprete* para que sea capaz de ser más autónomo e "inteligente".

[5] 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.

- Integración de sistemas inerciales y tecnologías inalámbricas para el posicionamiento en interiores. En este trabajo de investigación se presentan soluciones tecnológicas que pueden resolver el problema del posicionamiento estable en interiores, utilizando para ello tecnologías inalámbricas como WiFi, Bluetooth, RFID/NFC o ZigBee. Estas tecnologías se han utilizado como meros elementos conceptuales (es decir, podrían sustituirse por otras con capacidades similares). En cualquier caso, el posicionamiento en interiores sigue apoyándose en un parámetro inestable como es la potencia de señal recibida, y la alternativa de uso de técnicas basadas en tiempo de vuelo de señal no parece obtener buenos resultados en las condiciones adversas de interiores. Si bien la combinación de tecnologías con diferentes alcances y la utilización de estrategias híbridas afinan y mejoran la estabilidad del proceso de posicionamiento, la integración de dichos algoritmos con la información procedente de otros sensores, como inerciales, puede llevar a conseguir una mayor precisión y disponibilidad. Los inerciales están plenamente integrados en los dispositivos móviles de usuario para mejorar la interacción hombre-máquina (rotación de pantalla automática, apertura de mensajes de texto sólo con agitar el teléfono, corte de llamada girando el dispositivo, etc.), y esta integración puede utilizarse para pensar en combinar las tecnologías inalámbricas con localización, que mejoren la precisión del posicionamiento y aumenten la disponibilidad del servicio.
- Desarrollo de nuevas interfaces que aprovechen el conocimiento del contexto. Gracias a la incorporación de inerciales y a la integración de tecnologías como NFC en los dispositivos móviles, se abren nuevas posibilidades de interacción con el entorno que pueden aprovechar el conocimiento del contexto. La aplicación de estos conceptos de interfaz y el desarrollo de otros nuevos se pueden asociar a la situación del usuario y a su entorno de actividad. Por ejemplo, un paciente con discapacidades cognitivas leves puede recibir mensajes sencillos que le indiquen cómo interactuar físicamente con el entorno.
- Explotación de los datos de sensores biométricos y de información de contexto complementaria para la deducción de patrones de actividad y de emoción. De los datos de sensores biométricos se puede inferir información acerca de la actividad

que puede estar desempeñando una persona. Incluso, pueden colaborar a establecer patrones que permitan inferir el estado emocional de un sujeto. Las posibilidades de explotación de la información sobre las emociones son diversas, pudiendo mejorar la respuesta de los sistemas automáticos de asistencia a pacientes en el hogar, adaptar las sugerencias de sistemas de recomendación (de películas, música, etc.) y apoyar el desarrollo de nuevos tratamientos psicológicos, por ejemplo.

- Sistemas de gestión de flujos y de recomendación sujeta a la actividad. Siguiendo con la propuesta anterior, la implementación de sistemas de recomendación (guiado, información, planificación, etc.) conscientes de la actividad global en un recinto más amplio que un hogar como un aeropuerto, un centro comercial, un estadio de deportes o una ciudad en la que se celebra un evento masivo, pueden servir de base de diseño para proponer funcionalidades "conscientes" que optimicen y regulen el uso de recursos en entornos más complejos.
- Diseño de nuevos modelos de fusión de información. El proceso de extracción y combinación de conocimiento de fuentes heterogéneas es comúnmente conocido como fusión de datos. Los nuevos modelos de fusión de datos persiguen conceptualizar funcionalmente la construcción de una imagen en la que todos los usuarios o entidades implicados estén definidos en cuanto a identidad, posición, actividad e intencionalidad, a partir de los datos recabados por una infraestructura fija y a través de las relaciones con otras entidades que, por otra parte, es lo que se necesita en los sistemas sensibles al contexto.
- Seguridad y protección de datos. Es importante dotar de una mayor seguridad las comunicaciones realizadas dentro de la arquitectura HoCCAC, ya que los datos intercambiados pueden considerarse importantes, lo que hace necesario garantizar la privacidad y la protección de los datos.
- 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.

Por supuesto, hay muchos otros aspectos en los que la comunidad investigadora está trabajando, y que quedan fuera de esta lista. Quizás, entre los más importantes para garantizar la evolución de los servicios contextuales están los relacionados con la estandarización de la representación del contexto y de las interfaces entre diferentes componentes del sistema. Idealmente, los sistemas sensibles al contexto son capaces de adaptarse a la realidad humana de manera casi perfecta, pudiendo detectar el estado de ánimo del usuario, sus objetivos y sus intenciones, en una imagen que ya no es ciencia ficción. Por supuesto, esta visión no está exenta de temores, relacionados con la preservación de la privacidad, el control de la información ajena y uso fraudulento y no ético de los datos personales y de actividad. Pero la tecnología nunca es mala en sí misma: los sistemas sensibles al contexto permiten disponer de más recursos en la gestión de emergencias y crisis, mejoran el aprendizaje, colaboran en proporcionar a pacientes y discapacitados más autonomía, dan seguridad a un cirujano en su sala de operaciones y promueven nuevas formas de hacer, visitar y experimentar.

7

Conclusions and future work

7.1 Conclusions

Context-aware systems are integrating multiple communication and identification technologies at an increasing rate; they are capable of interacting with communication infrastructures that have quickly become widespread (mobile technology, WiFi, RFID, NFC, etc.) and with other devices within its surroundings. This suggests a future in which the use of these technologies (and their evolving forms) involves, among other things, a coordinated effort to acquire contextual information and inspire the design of future generations of adaptable and intelligent context-aware systems that are user based.

The distinguishing factor of this research work is the acquisition and ability to make decisions based on the context in order to configure adaptive solutions that target the specific needs of the user while also taking into account the capabilities of the devices and other resources within the environment. The HoCCAC multiagent architecture proposed in this study is based on the traditional models of context-aware systems while additionally adapting and expanding concepts and functions. In order to

obtain contextual information, HoCCAC attempts to standardize and simplify the process of gathering data from the context by using information *Provider agents*. For making decisions and interpreting the contextual information, HoCCAC incorporates planning and reasoning mechanisms into an intelligent agent known as the *Interpreter agent*. In the proposed architecture, the context is processed as a simplification of reality, more or less complex with regards to the mission as developed for the system by the architecture. The context-aware system is responsible for defining what contextual information is needed to achieve a system performance commensurate to its expectations. Contextual information is dynamic and is configured over time according to its characteristics; it is instantiated as needed by the *Interpreter agent* (by request, proactively, or upon detection of change in the environment). The goal is to minimize the resources required for the construction and subsequent instantiation of the contextual information, since there is always the future possibility that context-aware systems might interact autonomously with various entities or users simultaneously. To date, there is no known methodological proposal for addressing the process of developing context-aware systems or management systems for any underlying information. The HoCCAC architecture provides a framework for an analysis and functional design that can be of great use when the time comes for addressing these processes of development.

Additionally, the HoCCAC system provides a series of characteristics that facilitate and optimize the development of distributed multiagent systems based on context-aware computing. One of these characteristics is the distribution of resources. The functionalities of the systems and the agents themselves are modeled as independent applications and services. The resulting agents have a lighter computational load, which opens up the possibility of developing the system on devices with limited processing capabilities (Smartphone, micro-controllers, etc.). Another characteristic is the ability to reuse functionalities. Functionalities are modeled as independent applications and services. This makes it possible to reuse them in different developments, applying minor modifications that will adjust to the needs of each particular scenario. At the same time, functionalities can be repeated in order to obtain a greater availability of resources. HoCCAC is also characterized by its robust nature. Being a distributed type of architecture, it is possible to initialize and detain

applications, services or agents independently, without affecting the remaining system components.

The HoCCAC architecture was applied in the development of a dynamic and distributed multiagent system aimed at improving the care and surveillance of Chronic Obstructive Pulmonary Disease (COPD) patients at home [Fraile *et al.*, 2009c]. The architecture developed for this study incorporates a variety of technologies such as wireless networks, radio frequency identification and intelligent agents, and is based on concepts derived from context-aware systems. Its goal is to create a structure for a system that can adapt to the contextual needs of a user, provide a non-invasive interaction between the user and the system, and allow the ubiquitous and wireless communication and access of both information and services that is transparent to the user. An intelligent agent known as the *Interpreter agent* was developed for this system; it is based on the BDI model and integrates the CBR and CPM methods. The *Interpreter agent* is incorporated into the system along with other complementary agents in the multiagent system [Fraile *et al.*, 2010a]. The multiagent system can interact with its surroundings primarily by using wireless communication and RFID/NFC technologies. The *Interpreter agent* was designed with the goal of reasoning contextual information and automatically planning daily tasks for the patient, providing continuous updates regarding the patients' daily activities, and ensuring that the patients receive the appropriate care. The multiagent system developed in this study is made complete by the inclusion of: information *Provider agents* which optimizes the ability to gather information automatically; *LcA agents* responsible for dynamically managing existing information *Provider agents* and adding new information provider agents; a *Container agent* responsible for handling the model of system data and organizing the information received from and requested by the *Interpreter agent*; and with the *Interface agent*, responsible for acting as the interactive component for the system users.

The functionalities offered by this platform can be divided into two main groups: local, which are found within the home environment and used primarily by the patient; and remote, which provide information about the events that take place both inside and outside the home. Table 6.1 provides a list of functionalities according to these two categories. They can be subdivided as follows:

- **Emergency detection**. An emergency situation may occur due to a health anomaly or the detection of a change in the environment (e.g., a gas leak, increase in the level of humidity, etc.). The multiagent system detects the emergency and activates an alarm that is received by the surveillance interface and the devices associated with family members.
- **Health**. With regards to health care, the system can improve compliance with pharmacological treatments (e.g., control the intake of medication and the use of inhalers for COPD patients), and simplifying the diagnostic monitoring (e.g., send information on the state of the patient in execution time, and generate medical history reports that can be accessed by medical personnel).
- **Security at home**. The patients' physical security and well-being is among the greatest concerns of chronic patients at home. Although a complex security functionality has yet to be implemented, the present system notifies the patient when the front door has opened, providing the patient with the opportunity to alert or phone the security service or family members.
- **Comfort and well-being**. This category includes various services, among which are the control and automation of ambient conditions, search for objects, an alert system, and intelligent home design, all of which are carried out at the moment the services are executed. Controlling ambient parameters makes it possible to automatically activate the heating or air conditioning system (or suggest that the user do so). The object search service makes it easy to find items that may have been lost or misplaced inside the home (e.g., keys, glasses, cane, etc.). The alert system that is activated when the patient leaves can assist in automatically turning off the lights, ensuring the user does not forget an umbrella or to take their afternoon medication.
- **Social life**. The goal of the system developed for this study is to improve the patients' social life and avoid an isolated lifestyle. The proposed system establishes a patient reminder service for appointments and events (and may even, for example, prompt the user to call a family member on their birthday), and suggests activities and events for the user.

Table 7.1 List of functionalities for the developed multiagent system

Patient	Medical personnel / Family members	Functionality
		EMERGENCY DETECTION
	X	Detection and notification of anomalies in medical treatment.
X	X	Detection and notification of anomalies in the healthcare program.
X	X	Detection and notification of ambient changes (e.g., gas leaks, flooding, etc.)
		HEALTH SERVICES
	X	Signal reception for remote diagnosis. For prescriptions, the system may have to send personal data for analysis.
X	X	Send an alert notification for changes in physical activity (e.g., recommendation to reduce rhythm of physical activity in the event overexertion due to modifications in the original program).
X		Assist in complying with pharmacological treatment (contextualized notifications that include advice for taking the medication: "take with a glass of water" "take with food" "refrain from sun exposure" etc.).
	X	Monitor and summarize previous medications taken, including established schedule, notifications sent and the time when the medication is thought to have been taken, if that is the case (to evaluate appropriateness and compliance with treatment).
X	X	Specific recommendations for weight control (nutritional recommendations at mealtime's times, for example).
		SECURITY AT HOME
X		Intruder alerts (when somebody enters the home, the patient is alerted and can make a phone call if they sense any danger).
	X	Motion detection of doors opening during the night (if the patient is in the bedroom and the system detects a door/window opening, the system alerts the control center).
		COMFORT AND WELL-BEING
X		The air conditioning and heating will be automatically adjusted according to the temperature and humidity settings.
	X	Controlling ambient conditions (creating logs) to detect high temperature levels that may be harmful to the patient's health.
X		Alert system activated prior to patient's departure to automatically turn off bedroom lights, and remind patient to take medication or not to forget an umbrella if there is a chance of rain.
X		Search for lost objects. Some common objects (glasses, cane, and keys) can have a RFID/NFC tag attached to make it easier to find them.

Patient	Medical personnel / Family members	Functionality
		SOCIAL LIFE
X		Appointment and event reminder.
X		Contextualized suggestions for appointments and activities (calling a family member, attend an event, work on the computer, etc.)

The results obtained [Fraile *et al.*, 2009b] [Fraile *et al.*, 2009c] show that the multiagent system developed for this study can improve the process of carrying out the patient's dailly tasks, and reduce the time required by medical personnel in performing indirect tasks, such as monitoring the patient, without negatively impacting the quality of patient care. The multiagent system can also gather a significant amount of information about the patient and the surrounding environment. This information is analyzed by the *Interpreter agent* and used to make strategic decisions. Finally, the multiagent system can increase the patient's security at home in four ways: monitor the patient to ensure that the patient is at the correct location; provide the patient with the most appropriate task plans under unusual circumstances; limit home access to authorized personnel; and ensure the information is safely stored.

7.2 Contributions of this study

This study provides new contributions in the field of agent theory and context-aware computing, and proposes the HoCCAC architecture to facilitate the development of context-aware systems. The goal of context-aware systems is to improve the performance of traditional personalized systems by offering services based on contextual information. HoCCAC goes one step further by aspiring to achieve ubiquitous intelligent spaces that encompass most of the needs associated with the user's daily activities. The following list summarizes the contributions put forth in this research work:

- Provide a wide review of context-aware systems that have been developed with a variety of different objectives and that are related with other areas of research. For this reason, the present work study also provides an analysis of the benefits

of context-aware systems used in various areas such as Ambient Intelligence, domotics, and wireless communication.

- Propose the HoCCAC multiagent architecture, which facilitates the construction of intelligent environments through the use of intelligent agents. The architecture is based on providing ubiquitous computing and intelligent interface elements. The architecture was used to create an intelligent home environment for dependent persons, with its ability to model context-aware systems having been positively received, to offer learning mechanisms, planning and coupling mechanisms in dynamic environments to adapt to WiFi, ZigBee and RFID/NFC technologies. Given the many existing environments with similar characteristics to the present case study, one can imagine the many similar environments to which the architecture can be applied.
- The proposed architecture includes a CBR reasoning method which it combines with a CPM method, integrating both into the interpreter agent, which is based on a BDI model, to generate the most effective task plans. The *Interpreter agent* is thus an intelligent agent that contributes proactive solutions based on its interpretation of the contextual information. The CBR reasoning method is sufficient for developing systems that control a finite number of states and contextual images. Furthermore, it allows for the very simple inclusion of new rules, which can be configured by somebody without specific computer knowledge. The CPM method, in turn, generates efficient activity plans based on the daily tasks that the user must perform. The characteristics of the planning and reasoning methods are integrated into the HoCCAC *Interpreter agent*, making it unique in its definition and characteristics.
- The integration of new context sensors is usually both time and resource consuming, even if the hardware or software have programming interfaces. Using the *LcA* and the *Provider agents*, HoCCAC incorporates the particular characteristics of each device and the different control systems associated with each device. Furthermore, in many cases HoCCAC can calculate and subsequently attempt to minimize the resources consumed by the periodic

transmission of information for each device (e.g., automatically deactivate the device when the information it provides is not strictly necessary).

- Create a new framework for the analysis and design of intelligent environments. Review the fundamental concepts of context-aware computing, of the technologies used in the development of intelligent environments, and of previous studies of multiagent systems applied to the development of context-aware systems. Provide mechanisms that facilitate the incorporation of the characteristics of ubiquitous communication, ubiquitous computing, and intelligent interfaces for multiagent systems.
- The multiagent system developed with the HoCCAC architecture uses its agents to provide functionalities that can work with users at different levels of interaction. HoCCAC must take this into account because when it provides functionalities that require the collaboration or input of information by the user, it assumes that individuals can behave differently and that the system must respond according to each user's type of interaction. HoCCAC also uses the context as an information filter, as the "contextualization" of existing functionalities (e.g., remote medical assistance), and as support in performing daily tasks (such as the reminder service for taking medication).
- The proposed architecture combines contextual knowledge with different forms of interacting with the environment in order to offer new possibilities. The availability of technologies such as NFC in mobiles makes it possible to achieve more automation in initiating services. This technology can be of particular interest for applications such as the prototype presented in the case study, in which the eacy of interacting with the environment and the ease of learning the technology represents an added value to the system.
- During the course of developing this research study, a significant effort was made to obtain feedback from different research and research groups in similar fields with an interest in enhancing this research through the mutual exchange of ideas and knowledge. Special interest was placed on disseminating our

experiences and advances in research, from the initial stages through the final product, through various publications[6], conferences, workshops, etc.

7.3 Future work

The end of this work is only the beginning of new lines of work that should make it possible to put into practice many of the concepts mentioned throughout this work, to promote the interconnection of technologies and hardware and software components in any type of context, and to improve the problem of constructing an intelligent and efficient context.

- We propose further study in learning, reasoning and planning techniques that do not necessarily require configuring all patterns of the multiagent system at the design phase, but that allow the creation calibration, adaption, breakdown and elimination to take place in operating time. In other words, improve the capabilities of the *Interpreter agent* allowing it to be more autonomous and "intelligent".
- Integrate inertial systems and wireless technologies and place in indoor locations. This research study presents technological solutions that can resolve the problem of providing a stable indoor position, using wireless technologies such as WiFi, Bluetooth, RFID/NFC or ZigBee. These technologies have been used as nothing more than conceptual elements (i.e., they could be substituted by other similar capabilities). In any case, indoor positioning relies on unstable parameters such as the strength of the signal being received; the alternative use of techniques based on signal flight time does not seem to produce good results in adverse indoor conditions. If the combination of technologies with different signal strengths and the use of hybrid strategies improve the stability of the positioning process, the integration of the given algorithms with the information taken from other sensors, such as inertial sensors, can obtain more precision and availability. Inertial sensors are fully integrated in the user's mobile device to improve the human-machine interaction (automatically rotate screen, open text

[6] The list of publications is available at the BISITE research group website with the University of Salamanca http://bisite.usal.es

messages by simply shaking the telephone, end call by waving the device, etc.); this integration can also be used to combine wireless technologies with location, which improves the precision of identifying position and increases the availability of the service.

- Develop new interfaces that take advantage of contextual knowledge. The incorporation of inertial systems and the integration of technologies such as NFC in mobile devices opens new possibilities of interacting with the environment and exploiting contextual knowledge, The application of these interface concepts and the development of other new concepts can be directly connected to the particular situation and activities associated with the user. For example, a patient with a mild cognitive disability can receive simple messages that can indicate how to interact physically with the environment.
- Use the biometric sensor data and the complementary contextual information to deduce behavioral and emotional patterns. The biometric sensor data can be used to infer information about the activity the patient is performing. It can even assist in establishing patterns that can infer the emotional state of the subject. The possibilities for exploiting information about the subject's emotional state are quite varied, and can improve the automated response system to assist patients at home, adapt suggestions provided by the recommendation system (movies, music, etc.), and also, for example, support the development of new psychological treatments.
- Systems for managing the flow of activities and recommendations related to the activities. Following the previous proposal, the implementation of recommendation systems (guidance, information, planning, etc.) that are aware of the global activity in an area larger than the home, such as an airport, shopping mall, sports stadium, or even in a city celebrating a big event, can serve as a base for a design that proposes "conscientious" functionalities that optimize and regulate the use of resources in a more complex environment.
- Design new information fusion models. The process of extracting and combining knowledge from heterogeneous models is commonly referred to as data fusion. The new data fusion models strive to functionally conceptualize the

construction of an image in which all of the users or participating entities are defined by identity, location, activity and intention, using the data obtained from a fixed infrastructure and the relationships with other entities, which is needed in context-aware systems.

- Data security and protection. It is important to ensure the greatest possible security for any communication take place within the HoCCAC architecture. Since the data exchanged can be of great importance, it is necessary to guarantee the privacy and protection of the data.
- Test and validation. It is necessary to carry out much more extensive tests in order to evaluate the proposed models in more detail with regards to time, easy and quality of design analysis, calculation response time, quality of responses, etc. The results obtained will enable further development of more refined and robust models and systems.

Of course, there are many other aspects in which the research community is working that did not make it on our list. Perhaps among the most important for guaranteeing the evolution of contextual services are those related to standardizing the representation of the context and the interfaces between different system components. Ideally, context-aware systems are capable of adapting almost perfectly to a human reality, as they can detect the user's mood, objectives and intentions in an image that is no longer a product of science fiction. Of course, this vision is not without concerns, particularly those related to the preservation of privacy, the remote control of information, and the fraudulent and unethical use of personal data. However, technology is not itself the culprit: context-aware systems may allow access to more resources in emergency and crisis situations, they improve learning, provide more autonomy to patients and disabled persons, they provide security to the surgeon in the operating room, and promote new lifestyles.

8

Overview

This chapter is a summary of the research presented in this research work. The chapter is structured as follow. Section 8.1 is a brief introduction of the study. Section 8.2 presents the problem of context-aware computing and introduces the need for developing new systems that improve the living conditions of patients in their homes. Section 8.3 describes the proposed system, the interaction between agents and the HoCCAC system devices, the defined data model, the solutions proposed by the *Interpreter agent*, the CPM method task-planning, and a detailed design of the *Interpreter agent*. Section 8.4 presents a case study in which the HoCCAC architecture was applied at the home of a patient with Chronic Pulmonary Obstructive Disease (COPD), with special emphasis on the HoCCAC context-aware capabilities. Finally section 8.5 presents the results and conclusions obtained from the Home Care scenario prototype, and suggest future lines of work to improve the system

8.1 Introduction

A search for software environments that better adapt to the demands of users and their environment takes us to context-aware systems. These systems store and analyze all of the relevant information that surrounds and forms part of the user environment. User preferences, tastes, location, state of mind, activity, surroundings, ambient temperature, lighting conditions, etc., constitute the information that can be classified from the onset as contextual information. As a result, a context not only stores information about the user, but also contains information about user preferences. However, these systems face the problem of managing information about the environment to provide an appropriate response with regards to its current state. These systems can be applied within different environments such as the case of home care for patients.

Context-aware systems provide mechanisms to develop applications that understand their context and are able to adapt to possible changes. A context-aware system uses its surroundings to modify its behavior to best satisfy the needs of its user. The information is usually obtained through sensors. Because of the large number of small and portable devices currently available, the most common form of displaying information to system users is to distribute it through various heterogeneous systems and information networks. These systems are incorporated into the user's daily life to the point of becoming so imperceptible that the users focus on the tasks that they need to perform without worrying about the tools they must use to do so. This integration is carried out in such a way that the inherent technology of these elements or objects does not interfere with the activities for which they are used, thus providing the most simple, useful and comfortable use.

The previously described systems have wide-ranging applications. One of the environments in which they can be most useful is in home care. The high growth in the number of dependent persons and the advanced state of technological development has forced the need to generate new solutions for home care environments. Furthermore, recent commitments for satisfying the needs of this segment of the population indicate the need to modernize existing systems [Anastasopoulos *et al.,* 2005]. For this reason, home care environments require the

use of context-aware systems. There are advanced applications that can be installed in the homes of dependent individuals in order to improve their quality of life. Home care requires the use of sensors, intelligent devices and equipment to build a distributed environment in which basic home functions are automated. In this respect, multiagent systems can facilitate the development of home care environments. Multiagent systems [Bajo *et al.,* 2007] have been studied recently as monitoring system for the medical care [Angulo *et al.,* 2004] of persons who are ill or suffering from Alzheimer's [Corchado *et al.,* 2008a]. These systems provide continuous support in the daily life of these individuals [Corchado *et al.,* 2008b], predicting potentially dangerous situations, and managing the physical and cognitive support of the person being cared for [Bahadori *et al.,* 2007].

This overview presents the multiagent architecture Home Care Context Aware Computing (HoCCAC), which can supervise and monitor persons in specific contexts. The goal of HoCCAC is to facilitate the assistance of dependent users in their own home. The architecture provides a novel mechanism that integrates intelligent agents in a task-planning model based on the Critical Path Method (CPM) [Castro-Lacouture *et al.,* 2009]. The CPM method connects a series of related activities in the most optimal manner to reach a specific objective. By using the CPM task-planning method, the agents respond proactively within their environment. HoCCAC also uses information from the context-aware environment to predict user needs and provide effective solutions. HoCCAC incorporates CBR-BDI agents [Bajo *et al.,* 2007] that can learn by building upon their initial knowledge base. The CBR-BDI agents interact autonomously with the environment and the system users to adapt to the needs of the environment. Incorporating the CPM task-planning method into the CBR-BDI agents enables the HoCCAC intelligent system to improve its planning and learning over time. HoCCAC has a model of context-aware data defined to optimize information management. The simple integration and interaction of intelligent agents, planning methods, sensors, devices and a defined data model permits us to propose the HoCCAC architecture.

8.2 Context-aware computing

Context-aware systems were first introduced by Want (2006) when they presented their Active Badge Location System, generally regarded as the first context-aware application. It is a location system for individuals in an office environment, in which each person carries a badge that uses a sensor network to send signals containing information about each person's location to a centralized services area. Mid-way through the 90s, several tourist guide location-aware systems emerged [Raptis *et al.,* 2005] [Jbara *et al.,* 2007], which provided information about the user's location. Location information is by far the most used attribute of context-aware systems. Recent years have seen considerable growth in the use of other attributes of context-aware information. It is difficult to describe the term *context-aware* and many researchers attempt to provide their own description and the relationship of attributes that are included in it. The term itself first appeared in printed form by Schmidt (2005). Some authors describe context-aware as the location or identification of persons or objects [Henricksen *et al.,* 2006]. These descriptions of the term *context-aware* are often used in the initial stages of research in these systems. One of the most precise definitions was provided by Dey and Newberger (2009). These authors refer to context as any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the applications themselves.

There are various location-aware infrastructures that are capable of gathering positional data [Büscher *et al.,* 2009] [Syvänen *et al.,* 2005]. These systems include satellite GPS, mobile phone towers, proximity detectors, cameras, magnetic card readers, barcode readers, etc. These sensors can provide information about position or proximity and differ only in their precision. Some require a clear line of vision; other signals can penetrate walls, etc. The previously mentioned systems only use one context attribute: information on the position of the object or person. The use of different context attributes such as noise, light and location makes it possible to combine context objects at a higher level. These elements are needed to construct systems that are more useful, adaptive and easy to use. One example of this type of

context-aware infrastructure is the system presented by Moreno *et al.,* (2006), which improves communication by adding context-awareness to the management of information within a hospital environment. Each user (in this case, doctors, nurses, etc.) is given a mobile device to write messages that are sent upon completion of a given set of circumstances. The context attributes included in this system are location, time, roles, and the state of the user or entity being analyzed.

As noted in the previous studies, the attributes used by the majority of context-aware systems are the location or situation of persons, objects or entities. Few systems are able to use information taken from different context attributes and relate different types of data to interact with users or patients. The goal of the present study is to move one step further and, in addition to using the different context attributes in the proposed system, store and process different types of data gathered by the system in order to improve the quality of life of dependent persons in their home.

Using a context-model as our basis, we propose the addition of the HoCCAC multiagent architecture, which offers context-aware services to users in specific contexts, and includes a set of independent services that gather and interpret context data. The fundamental characteristic of the system is to improve patient care through the processing and reasoning of the data provided by the context. The system can easily develop context-aware services and applications in a variety of contexts. The system is independent, as it can be applied to various types of hardware devices and operating systems, and is based on Java technology. Patients are authenticated and located within the environment by a RFID chip with JavaCard technology that they carry. HoCCAC defines a light framework for the execution of service-oriented applications. The functions of this system include management of the installation, activation, deactivation, update and elimination of services, as well as the authentication, control and monitoring of users at all times.

8.3 HoCCAC multiagent system

Agent and multi-agent systems have become increasily relevant during the last decades and have gained relevance in different areas [Pokahr *et al.,* 2003] [Bajo *et al.,* 2010]. Multi-agent systems are distributed organizations where the components

(agents) collaborate to achieve a series of goals. The agents are autonomous entities that can be characterized through their capabilities: autonomy, proactivity, reactivity, social skills, organization, etc. [Corchado *et al.,* 2008a]. These capabilities make the agents very appropriated to be applied in business intelligence and risk management scenarios. In these scenarios the agents can play different roles and establish an organizational model where the human behaviours and the management processes can be emulated. In this sense, it is possible to obtain different agent types, specialized in concrete tasks and behaviours, that can collaborate together to increase the productivity of the business and provide an effective risk management. The agents can act as an interface between the human users and the systems, trying to provide advanced facilities and personalize the access to the system, but also can act as autonomous entities that are proactive and can make decisions independently.

HoCCAC is a distributed system composed of intelligent agents that can reason out and carry out optimal plans of action. The HoCCAC architecture is primarily focused on tracking, control and notification. HoCCAC is defined by the need to control distributed devices and gather user information in Context-Aware environments in a manner that is both non-intrusive and automatic. Furthermore, HoCCAC performs the handling, storage and reasoning of context-aware information, as well as the security for managing the access and administration of information. The HoCCAC system uses sensors to receive information from its environment. It takes the context data from this information so that the intelligent agents can processes the information received and respond accordingly. With the data they are provided, the intelligent agents create or carry out plans of action, always opting to maximize or improve the expected result. HoCCAC also combines the management of personal data with a daily activity model defined by intelligent agents, using the data provided by the sensors installed in the household network. The interpretation and processing of both the knowledge base and the daily models, which is carried out by intelligent agents, provides an added value to the system. As a result, HoCCAC anticipates the requests or possible incidents that users may confront within their context, and offers solutions based on past experiences that can improve the expected results.

In order to obtain and process context data and provide the user with solutions, the HoCCAC system employs a multiagent architecture that is composed of various types

of intelligent agents, as shown in Figure 4.1. These agents understand and respond to their environment. Additionally, HoCCAC provides the agents with mechanisms to access the system services via mobile devices. The multiagent architecture is comprised of the following types of agents:

- *Provider agents* capture the data obtained by both internal and external heterogeneous context sources so that the *Interpreter* and *Container agents* can process and reuse the data. The types of provider agents that can connect to the system are: (i) Situation agents, which maintain a constant fix on the location of a patient in the home, (ii) Flood Detector agent, (iii) Gas Detector agent, (iv) Smoke Detector agent, (v) Lighting Detector agent, (vi) Thermostat agent, (vii) Control agent, which establishes and controls the daily parameters desired by the patient, (viii) Weather agent, which registers the exterior temperature in order to respond accordingly to the internal variables of the home, (ix) GPS agent, which maintains the location of the patient outside the home. The system is dynamic and capable of incorporating an information *Provider agent* at any moment by adding the corresponding sensor or gathering the necessary information from an external server or provider.
- *Container agent* stores the context-related data. This agent is responsible for managing the devices, the location of devices, and detailed information on the context in which it is located. It also manages old information related to the users that no longer interact with the environment and deactivated devices.
- *The Interpreter agent* provides logical reasoning services to process the context information. The *Interpreter agent* is based on the concept of case-based reasoning [Corchado *et al.,* 2008b] to improve its autonomy and increase its ability to problem solve. The sensory information is merged and translated to obtain the context information and update the state of the patient. The *Interpreter agent* uses the information available in the system, the signals received from the *Interface agent*, and the knowledge base and beliefs related to the context in order to reason out the actions that it will execute. This creates a reasoning process that determines the optimal course of action or plans for the user to reach his or her objective. To this end, the *Interpreter agent* uses the concept of CPM to generate

plans as solutions. The *Interpreter agent* provides the ability to reach complex, high-level objectives and avoid errors that can lead to inefficiency. It also allows for greater flexibility when faced with new objectives.

- Resource Identification Agent (LcA). This agent is in charge of maintaining a record of the active *Provider agents* in the system, and of allowing or denying the inclusion of new provider agents. The LcA agent informs the *Interface* and *Interpreter agents* of any changes so that they can take into account any new context attributes that are provided, such as the inclusion of a new *Provider agent*.
- Interface Agent, interacts with the *Interpreter* and the *LcA agent* without the need for explicit user instructions. The *Interface agent* reads the entries provided by the user through the context-aware applications and sends any modifications in behavior to the *Interpreter* or *LcA agent*. The *Interface agent* also notifies users with context-aware applications. The *Interface agent* can, for example, receive many entries from context-aware applications over an extended period of time prior to deciding on any single action to take, or it can receive a single entry from a single context-aware application, which can initiate a series of actions by the agent.

HoCCAC allows the integration of context-aware applications. The context-aware applications represent all of the programs that can be used to take advantage of the system functionalities. HoCCAC makes it possible to easily use and share context-aware applications in a variety of physical spaces. The context-aware applications integrated in the HoCCAC architecture, also shown in Figure 4.1, receive the information available in the system from the Interface Agent. They also use different levels of context information and adapt their behavior according to the active context. Context-aware applications are dynamic and adaptable to the context, continually reacting to changes in the state of the context. Context-aware applications can be executed locally or remotely, even from mobile devices with limited processing capability. One way to develop context-aware applications is to specify the actions that respond to context information under specific rules and conditions.

Figure 4.7 provides a general description of the HoCCAC distributed system infrastructure. The image shows how laptops and mobile devices connect to the system via internet. Additionally, each of the devices are interconnected through wireless communication networks, mobile networks, or RFID technology. The HoCCAC system facilitates the integration and management of agents, devices and control systems. Communication between platform agents follows the FIPA ACL standard (Agent Communication Language). The communication protocol between the agents and the services is based on the SOAP [Tapia *et al.,* 2011] standard. Agents can invoke two types of services: those that capture context information from the data obtained from autonomous components, and those that respond to the automatic control systems installed within the context. All of this allows HoCCAC to be an easy system to implement within complex environments, without depending on a specific platform.

The following subsection 8.3.1 provides a general description of the functions of the HoCCAC agents and their interaction with devices in their environment. Subsection 8.3.2 describes the system data model. Subsection 8.3.3 presents solutions provided by the *Interpreter agent*. Subsection 8.3.4 describes the graphical representation of the task plans using the CPM method incorporated in the *Interpreter agent*. Finally, subsection 8.3.5 provides a detailed design of the Interpreter agent.

8.3.1 Interaction between agents and devices in the system.

The *External Provider agents* obtain context information from external autonomous components such as, for example, a meteorological information server that provides weather information for a particular area. A location server can also provide information about the location of a person who is not at home. The *Internal Provider agent*s gather information directly from the sensors installed in the environment, such as RFID location sensors installed in the home of a patient, or light sensors. At the same time, all of the *Provider agents* can interact with the user. The *Provider agents* are some of the most important components for achieving appropriate interaction with users. This type of agent interprets the information sent from the agent platform and presents it to the users. The following example can be used to better understand this interaction: a location service receives constant information

from RFID devices. This information is sent periodically to the agent platform. The information is interpreted and sent to a user's mobile device to find the position of the user within a specific area. The user's device does not require complex calculations to determine the position of the user since it needs only to interpret the data sent by the platform and display them to the user in a simple manner.

The functions of the *Interpreter agent* are to process information provided by the *Container agent*, and to process the context information. The *Interpreter agent* uses both the information it has processed and past experiences to develop plans of actions and initiate services that interact with the user and his or her context. For example, when the *Provider agents* detect a new user in the context, they inform the other agents, then the *Interpreter agent* processes the stored information for that user, and determines the temperature and lighting desired by the user in the environment, based on past experiences and data.

The *Interpreter agent* uses the context-aware applications to generate task plans and interacts with the sensors and automatic system controls installed in the environment. The context-aware applications use different levels of context information, as they are able to adapt their behavior to the context within which they are executed. Upon consulting the data registered in the information system by the *LcA agent*, these applications can locate the services from each of the providers it may be interested in. The context-aware applications can obtain context data by asking the *Interface agent* or waiting for an event from the *Interface agent*. The *LcA* allows users and agents to locate the different context applications. The primary characteristics of the *LcA* include scalability, adaptability and multiple processing capabilities. The *LcA* controls large areas, located either in internal or external networks, where the context providers interact. Following the addition or elimination of physical sensors, or reconfigurations of the actual devices, the *LcA* searches for and adapts to changes in the environment. A new mechanism is also displayed to allow the context providers to inform the system of their functionality.

Figure 4.10 provides an example of monitoring a user in a context where the *Internal Provider Situation agent* sends signals to the *Interpreter agent* to inform on the situation of the user. The *Situation agent* also receives signals from the *Interpreter agent* to, for example, allow or prohibit user access to controlled areas. The *LcA*

agent, on the other hand, records all of the services that the *Situation agent* offers. The *LcA agent* continually updates the list of services provided by the *Situation agent* which it can then transfer to the active applications in the context. Additionally, the *Container agent* manages the context information according to the data provided by the *Interpreter agent*.

8.3.2 Description of the data model

The primary goal of the HoCCAC system is to improve the living conditions for a user in his or her context. To this end, a set of devices and context-aware applications capture context information about the user and the user's routine. This information is gathered by the agents defined in the system, and the *Container agent* ensures that the information related to the management of the context is stored in the system data model, as shown in Figure 4.11.

As shown in Figure 4.11, the domain model can store the preferences for various users on devices. The rules of action are specified in text format according to the attributes, methods and operators specific to each device. The rules of interaction are interpreted using a parser, similar to how a decision tree is interpreted by data mining software such as Weka. The actions can be specified manually or generated automatically using a deliberative model. Based on the preferences and context information stored in the HoCCAC information system, the *Interpreter agent* executes the available services to act on the context and the user. The *Interpreter agent* stores information in its knowledge base that indicates the situation and current state of the user in the context in order to respond quickly to the user by using the devices closest to the user. In the event that the user is located outside the context-aware environment, the *GPS agent* will be responsible for recording the location of the user periodically using GPS coordinates. Additionally, one of the bedrooms, the kitchen, bathroom and living room in the home will be equipped with smoke, gas, light, flood and temperature detectors. Each of the values picked up by the sensors is gathered by the corresponding agent and sent to the *Interpreter agent*, records stores the information.

The HoCCAC agents exchange information through ACL messages. For example, at a predetermined time, the *Interpreter agent* receives an ACL message with the

XML file that is shown in Figure 4.12. The information received by the *Interpreter agent* in the XML file shown in Figure 4.12 is stored in the system database by the *Container agent* for its subsequent use, and processed by the remaining system agents.

8.3.3 Solutions provided by the Interpreter agent.

The *Interpreter agent* is integrated in the HoCCAC system. The goal of this agent is to provide the user with efficient solutions in execution time within the context of the user and to improve his or her quality of life. The most important characteristics of the *Interpreter agent* is: (i) reasoning capabilities to analyze and reason out context data collected by the system to develop task plans and provide proactive solutions, (ii) to adapt easily to the context within which it functions, and (iii) to collect sensor data and messages from other agents to improve plans of action. One example of the functionality of the *Interpreter agent* applied to the user's preferred temperature conditions is when the system detects the presence of a user in a specific context using the RFID chip sensors that identify the user. The system has saved the user's temperature preferences within the context. It has also saved the case base for the user from other similar instances and also obtains the exterior ambient temperature. With these entries the *Interpreter agent* generates dynamic CBR-based action plans to stabilize the temperature in the environment while it detects the presence of the user. At the same time, the *Interpreter agent* is also responsible for planning tasks for the user and, consequently, planning the activities that the user must carry out for the day. Planned activities include the use of oxygen and any activities that can be carried out concurrently. These action plans are based on the CPM method. The *Interpreter agent* sends the action plans to the *Interface agent* who then interacts with the temperature control device. The system thus maintains the temperature desired by the user within a specific context. In order to achieve the results desired by the user, the *Interpreter agent* uses a CPM method to provide a CBR-based planning strategy.

CBR is a type of reasoning based on the use of past experiences to solve problems [Corchado *et al.,* 2008a]. The goal of CBR systems is to solve new problems by adapting solutions that have been previously used to solve similar problems in the past. A case can be defined as a past experience and is composed of three elements: a

description of the initial problem, a solution that provides a sequence of actions that must be carried out to solve the problem, and the final state that describes the end state once the solution has been applied. A CBR system generates cases, past experiences, to solve new problems. The way in which the cases are managed is known as the CBR cycle.

An example of a CBR cycle is shown in Figure 3.2. The cycle includes 4 sequential phases: retrieve, reuse, revise and retain. The retrieve phase begins when the description of a new problem is received. Similar algorithms are used in order to recover the cases with a problem description similar to the current problem. Once the majority of the similar cases have been retrieved, the reuse phase begins. In this phase, the solutions to the retrieved cases are adapted to obtain the best solution for the current case. The revise phase reviews the proposed solution. Finally, the retain phase allows the system to learn from the experiences obtained in the three previous phases and updates the data.

With regards to the CPM planning method, its primary goal is to determine the duration of a project or task plan, the latter being a sequence of interrelated activities, each of which has an estimated length of duration. The CPM method is widely applied in planning processes. CPM can control the execution of activities and determine which are the critical activities that must be carried out so that the global plan is not delayed [Kwak *et al.,* 2008]. CPM is applied in various fields such as manufacturing processes [Gupta *et al.,* 1991], industrial environments [Kwak *et al.,* 2008], task distribution tasks in grid computing [Lina *et al.,* 2006], or information processing in distributed mode in hospitals [Kwak *et al.,* 2008]. With the CPM method, the duration of all activities is known; that is, there is no uncertainty. This simplifies the CPM method, making it easy to use. With the CPM method, the length of the task plan is the same as the critical path. This is the largest path for the set of project activities. The CPM method is applicable and useful in any situation in which it is necessary to carry out a set of interrelated activities to reach a specific objective. The true value of the technique in the CPM method increases when it is applied dynamically. In the event of unanticipated events or circumstances, the critical path method provides the ideal means to identify or analyze the need to reexamine or reschedule the project, reducing the adverse effects of the unanticipated events to a minimum. Similarly,

when an opportunity arises to improve the project scheduling, the technique makes it possible to easily determine which activities should be accelerated in order to achieve the expected improvement. These scenarios are what make the CPM method the most adequate for the *Interpreter agent* to implement in the HoCCAC system. The application of the CPM method by the *Interpreter agent* allows the HoCCAC system to have the following information available: (i) which tasks should be carried out first, (ii) when to use system resources, (iii) how to schedule the use of devices, (iv) how to schedule the advancement of activities, (v) how many activities there are and which ones are carried out at any given time, (vi) the state of the current task plan in use with regards to the end date, (vii) which are the critical activities that, if delayed, will delay the length of the task plan, and (viii) which are the non-critical activities and their margin for delay; these are activities that can be delayed, without delaying the project, for a period of time referred to as a float. The CPM method essentially consists of two cycles:

- Scheduling and planning. In this cycle, the *Interpreter agent* defines the task plan with all its activities or main components. The relationship between the activities is then established to decide which activity should begin first and which will follow. The relationship among the activities is used to design a network connecting the different activities based on their levels of priority. Subsequently, the estimated cost and time is defined for each activity, and the feasibility of the task plan is evaluated. Finally the longest path for the project is identified, which will determine the length of the task plan, which in turn identifies the critical path.
- Execution and control. In this cycle the *Interpreter agent* approves the task plan, instructions are defined, advances are controlled, and finally corrective measures are implemented.

The following section describes the graphical representation of the task plans that the *Interpreter agent* carries out with the CPM method.

8.3.4 Task planning with the CPM method.

In order to develop the most optimal task plans, the *Interpreter agent* receives a series of information from the HoCCAC system and follows a set of standards for the CPM method. A task plan is composed of a set of activities that must be carried out in a specific order to achieve an objective. A graphical representation of a task plan using the CPM method is referred to as a network. Each activity can include one or more tasks. The activities include events that are instances of the activity and act as control points. The events describe the moment that the activity begins or ends. The activities are represented by arrows whose length is not indicative in any way of the duration of the activity. The events are represented by circles. When building a network it is necessary to keep in mind that each activity is represented by one and only one arrow and each activity should be identified by two nodes, as a single common node can have no more than two different activities.

Each arrow on the network represents an activity. Each activity is a task required by the project. A node represents an event, which is defined as the moment when the activities that arrive at that node finish. The points of the arrow initiate the sequence in which the events must take place. An event should precede the start of the activities that stem from that node. The node towards which all the activities lead is the event that corresponds to the conclusion of the task plan. The network can represent a task plan from its start or, if the task plan has already be initiated, it represents the plan for its completion. Thus, each node in the network represents the event that maintains the activity in progress or the event that initiates a new activity that can begin at any moment. Each arrow plays a dual role: represent an activity and assist in representing the relationship between the different priority activities. On occasion an arrow may be necessary to define the priority relationships, even though there is no real activity to represent. In this case, a fictitious activity is represented by a dotted line, indicating the priority relationship. The fictitious or phantom activity does not consume time and achieves two objectives: (i) to save a priority relationship, and (ii) to individualize each activity so that each one is identified by a unique pair of nodes. In order to create a network task plan, it is necessary to answer three basic questions about each arrow or specific activity:

- Which activities should be carried out immediately prior to executing a specific activity?
- Which activities should be carried out immediately after executing a specific activity?
- Which activities can be carried out while simultaneously executing a specific activity?

A common rule for building these types of network task plans is that no two nodes can be directly connected by more than two arrows. The fictitious activities can also be used to apply this rule when they have two or more concurrent activities. Additionally, the node numbering allows for identifying the different activites using events with start “i” and finish “j”. It is also possible to randomly number the nodes, as there is no reason that this cannot or should not be done. Nevertheless, experience has demonstrated that a systematic numbering of nodes simplify the arithmetic process. It is good practice to number the nodes in such a way that the number of the initial node for any arrow is always less than the number indicated in the node towards which the arrow is pointing. In other words, “i" should be less than “j”. Nor should two different nodes be assigned the same number.

As for the activities, two activities are said to be simultaneous when than can be completely or partially carried out within the same interval of time without slowing each other down. Two activities are also said to be connected when the beginning of one of them is contingent on another the completion of another activity or activities, as seen in Figure 3.3b with activities B and E. Additionally, concurrent activities are those ending in the same event, while divergent activities are those that begin in the same event, as seen in Figure 3.3a with activities A and B. Another representative example can be seen in Figure 3.3c whereby in order to carry out activity N, it is first necessary to finish activities L and M; and to initiate activity O, it is only necessary to finish activity L. Figure 3.3d also illustrates a network that includes fictitious activities, since in order to carry out activities M and N, it is necessary for activity L to have finished, and to carry out activity O, it is necessary to have completed activities M and N.

To date, the only existing restriction for correctly developing the network is to establish a logical sequence of activities. To perform a sequence of activities, the *Interpreter agent* takes into account the information contained in the HoCCAC system and its reasoning. Once the network has carried out the activities, a corresponding duration should be assigned to each one. This way, the *Interpreter agent* can calculate the total duration of the project and determine the early completion dates for each activity. To calculate these times, the *Interpreter agent* must take the following into account:

- When task plan begins.
- No activity is to be initiated without having previously completed all tasks whose execution is contingent on that activity.
- Each activity should be carried out as soon as possible.
- Once initiated, each activity is executed without interruption until its completion.

Just as it is possible to calculate early start dates or times for initiating and completing an activity, the *Interpreter agent* can also calculate late start dates or times for each activity for the total duration of the project. Calculating these times is quite simple: the earliest an activity can begin is the closest time or date that all of its preceding activities can finish. The earliest an activity can finish is the earliest required start date added to the time required for its completion.

The first calculation made by the *Interpreter agent* is of the early start times for each activity, as graphically represented in Figure 3.4. The procedure is as follows:

1. Assign a start date and time ES_i to the first event "i" of the first network activity.
2. Take the sum of the duration of each of the activities that precede event "j" and indicate the total as ES_j. ES_j is also the earliest start time for event "j" and can be defined as:

$$ES_j = \max_i \left\{ ES_i + t_{ij} \right\} \quad con \quad i < j$$

3. If two or more activities have the same final event, the maximum value obtained from the calculations performed in step 2 should be considered.

4. Steps 2 and 3 are repeated until the earliest time for carrying out ES_i is determined for all activities.
5. The final sum of the earliest start times constitutes the time required to carry out the Project.

The second calculation made by the *Interpreter agent* is for the late end times, as illustrated in Figure 3.4. The procedure is as follows:

1. The late end time for the last event "j" is equal to its earliest start date ES_j. The total duration of the project is used as the initial data, which is noted to the far right of the final event as LS_j.
2. The duration of each of the activities that finish in event "j" are subtracted from LS_j. The subtracted values are the late end times LS_i and can be defined as:

$$LS_i = \min_j \left\{ LS_j - t_{ij} \right\} \quad con \quad i < j$$

3. When two or more activities have the same starting event, the minimum value obtained from the calculations in step 2 should be considered. This minimum value is the late end time LS_i for the previous activities.

The *Interpreter agent* also calculates the margin for delay TF (Total Float) for an event such as the difference between the latest time and the earliest time to begin or end an activity. This is defined as:

$$TF_{ij} = LS_j - ES_i - t_{ij} = LS_i - ES_i \quad \forall_{ij}$$

The margin for delay of an event indicates the maximum delay allowed to complete the event without delaying the completion of the task plan. The margin for delay of an activity indicates the same with respect to a delay in the completion of that activity. The activities with a zero margin for delay are critical since any delay will postpone completion of the task plan. If an activity A is critical, it is designated as:

$$\underset{actividad\ \ crítica}{{}^{*}A_{ij}} \rightarrow \underset{tiempo\ \ crítico}{{}^{*}t_{ij}}$$

The critical path for a task plan is a path crossing the network in such a way that all of its activities have a zero margin for delay. It is the longest path in the network and corresponds to the minimum time for finalizing the task plan. The total duration of the

task plan (D) is the sum of the critical times of the critical activities and is represented as:

$$D = \sum {}^{*}t_{ij}$$

Critical paths are also characterized by a series of properties. A task plan always has a critical path, often more than one. All of the activities that have a zero margin for delay should be on a critical path, while no activity with a margin for delay greater than zero can be on a critical path. All of the events with a zero margin for delay should be on a critical path, while no event with a margin for delay greater than zero can be on a critical path. A path on a network in which the initial and final events have a zero margin for delay is not necessarily critical, because it may be that one or more activities along that path can have a margin for delay greater than zero.

All of the rules and procedures previously described are contained in the interpreter agent, specifically in a module referred to as CPM Context-Aware Plans, as explained in the next point and as shown in Figure 4.13.

8.3.5 Interpreter agent Design

As previously described in section 7.3.3, CBR is a paradigm based on the idea that similar problems have similar solutions. The agents that are designed and implemented using CBR systems have autonomous reasoning skills and adapt to changes in the environment. As such, the *Interpreter agent* fulfills two of its most important characteristics, as noted in the previous section: (i) reasoning capabilities, and (ii) ease in re-planning to adapt to the context. The other important characteristic is the ability to collect information from the environment using sensors and other agents. For this reason, when designing and implementing an *Interpreter agent* it is necessary to take into account the exchanges of information among the system agents. FIPA [7] specifications are currently considered the accepted standard for communication between agents. The *Interpreter agent* is designed with the Agent Unified Modeling Language (AUML[8]) methodology, which provides mechanisms to obtain a design detailed enough to greatly facilitate the implementation phase.

[7] www.fipa.org
[8] www.auml.org

On the other hand, the BDI [Rao *et al.,* 1991] model is a solid base for modeling and applying the internal behavior of the agents. With the BDI model, it is possible to view an agent as an entity searching for an objective and behaving rationally. CBR systems and BDI agents can interact if the cases are implemented as beliefs, intentions and desires that lead to the resolution of a problem. With the CBR-BDI agent, each state is equivalent to a belief. Furthermore, the objective to reach can also be a belief. The intentions will be plans that contain a specified set of actions that the agent must carry out in order to reach its objectives. The agent can move from one state to another upon executing the action or task. A desire will be one of the final states reached in the past. To produce a proper transition between the design phase and the implementation phase of the CBR-BDI paradigm, the agent must be supported in the implementation phase.

The Java Agent DEvelopment Framework (JADE) platform [Zhiming Zhao *et al.,* 2007] is a good option for developing agent based applications. JADE agents follow the BDI model, and the JADE platform can be used to implement CBR-BDI agents. JADE also complies with the FIPA standard for operating with intelligent multiagent systems. The JADE platform focuses on applying the FIPA reference model, providing the required communication infrastructure and a services (e.g., management agent) platform, as well as a set of tools for developing and debugging CBR-BDI agents. *Jadex* (JADE eXtension) [Pokahr *et al.,* 2003] is an implementation of a hybrid agent architecture (reactive and deliberative) to represent the various states of the JADE agents that are following the BDI model. *Jadex* is designed to be easily integrated within JADE, with the addition of a packet. The primary goal is to facilitate the use of reasoning and planning concepts during implementation.

Viewed externally, the *Interpreter agent* is a black box that receives and sends messages. This section will now attempt to provide greater detail regarding the functionality of the *Interpreter agent* via the use of a graphical representation. The *Interpreter agent*, as previously described, will be implemented as a *Jadex* agent. In order to implement the interpreter agent as a *Jadex* agent, some variations are introduced into the *Jadex* architecture, as shown in Figure 4.13.

Figure 4.13 provides a summary of the *Interpreter agent* architecture. The incoming messages, as well as the internal events and new objectives, serve as a

starting point for both the internal reactions and deliberative and reasoning mechanisms of the *Interpreter agent*. The primary novelty in the design of the *Interpreter agent*, as seen in Figure 4.13, is that it integrates a CBR reasoning engine and a reactive system that collects data from sensors and the control systems. This provides a unique attribute in the design of the agent with regards to its conception and reasoning capabilities. Based on the results of the CBR reasoning engine, the *Interpreter agent* develops task plans using the *CPM Context-Aware Plans*, taking into account the standards and procedures described in point 8.3.4. These plans can be executed immediately as events, or they can be stored in the library of context-aware plans to generate new plans to be executed at a future time. Executing the plans can modify the context-aware beliefs base, send message to other agents, create new context-aware objectives, or produce future internal events.

The *Interpreter agent* has a context-aware belief base, as shown in Figure 4.13, in which it stores the beliefs that constitute its knowledge base. These beliefs are related to the context-aware environment and to the user. These beliefs include the location of the user, the exterior temperature and that of the home and its rooms, or the lighting and smoke levels in the different rooms of the home. The beliefs are structured through Java objects that represent the beliefs, as seen in Figure 4.11. These objects have a name and attributes that have simple or multiple values. The knowledge base also incorporates the concept of databases oriented to objects. Object Constraint Language[9] (OCL) can retrieve subsets of context-aware beliefs. Another special characteristic of the context-aware belief base are the conditions, which represent an expression of a particular state, for example, or of one or various beliefs. Once the condition is satisfied, an internal event is generated. This event can activate a plan or allow the adoption of new objectives. With the *Interpreter agent*, beliefs represent changes in the state of the sensors installed in the context-aware environment. This makes it easy to add new types of sensors that assist in the daily tasks of the user, and for the task plans to add new states for a sensor at a future time. The *Interpreter agent* also includes specific plans for collecting data from the environment and the control systems. Furthermore, the task plans use OCL to inquire on the beliefs that meet

[9] http://www.omg.org/technology/documents/formal/ocl.htm

certain conditions. All of the task and action plans specific to the *Interpreter agent* contribute towards reaching the final objective.

The *Interpreter agent* can define three types of objectives: reaching an objective, maintaining an objective, and carrying out an objective. To reach an objective, it is necessary to have previously defined the state to be reached, but not the specific way to reach it. In this case, the *Interpreter agent* has several alternatives for reaching the objective. To maintain an objective, the *Interpreter agent* must monitor the state and execute the plans to re-establish the state when necessary. To carry out an objective, the *Interpreter agent* specifies the actions to be executed. The objectives of the *Interpreter agent* can also be represented by objects with various attributes. The plans for achieving the objective are explicitly stated through the conditions (e.g., using the beliefs). The name and properties of the objective facilitate the selection of a plan to initiate, and the parameters guide the tasks of the plans being executed. When an objective is carried out, the plan to execute is directly defined. The reasoning ability of the *Interpreter agent* can be implemented in one of two ways. One is through the activation and deactivation of conditions. In this case, the *Interpreter agent* applies rules according to the desired focus for activating or deactivating objectives that meet certain internal conditions, such as those conditions defined according to the knowledge base and user preferences. However, the objectives can also be activated and deactivated manually using task plans. The *Interpreter agent* maintains an objective (e.g., maintaining the desired temperature in a room), carries out an objective (e.g., generating daily task plans for a specific user), and reaches an objective when, for example, it activates the gas extractor prior to activating the smoke alarm.

The *Interpreter agent* has a library of context-aware plans based on the CPM method. With the library of plans it can generate solutions based on action plans that have been initiated in the past. These action plans can interact with the devices installed in the system to facilitate the user's daily tasks, thus making the user's stay in the context-aware environment more comfortable. One part of the implementation process of the *Interpreter agent* divides its functionality into separate plans composed of actions that are implemented in Java classes. We can say, then, that object orientation techniques are also used when carrying out the plans. Additionally, the

functionality that is implemented in Java classes can be incorporated into other similar or hereditary systems. The following section then proposes a low level AUML design for the *Interpreter agent* followed by the implementation using *Jadex*. The AUML design provides a class diagram for the *Interpreter agent*, as shown in Figure 4.17. It involves the most important agent within the HoCCAC architecture. This agent, as shown in Figure 4.17, has five capabilities and four services. The capabilities are: (i) P-Solution, (ii) C-Sensor, (iii) S-Plans, (iv) St-Data y (v) E-Result. The services are: (i) Information Provide, (ii) Plan Describe, (iii) Result Plan Provide and (iv) Component Task Assignment.

The execution of the *Interpreter agent* using the *Jadex* model is based on events. Anything that occurs within the *Interpreter agent* is represented as an event. The message events indicate when an ACL message has been received. The objective events announce when an objective has been reached, and the internal events inform, for example, of any changes in beliefs, timeouts, or conditions that have been met. To create and initiate an *Interpreter agent*, the system needs to know which of the *Interpreter agent's* properties will be instantiated. The state of the *Interpreter agent* is determined by the beliefs, objectives, plans of execution, and library of plans. All of these concepts (beliefs, objectives, plans, conditions, and filters) are defined in the Agent Definition File (ADF). The ADF applies a Java-based declaration of objects to define the initial beliefs and objectives. The plans are stated by specifying the way to create an instance in the Java class. The ADF file format is a file of properties with pairs of name/value that map out references to object statements. Figure 4.18 shows an excerpt of the definition for the *Interpreter agent*. The next section describes the execution model that the *Interpreter agent* follows.

The model for executing the *Interpreter agent* is based on the four behaviors of the JADE framework, and on the context-aware data collector, a new behavior that is added to the architecture for the *Interpreter agent*. These behaviors, which are executed within the *Jadex*-based interpreter agent, can be seen in Figure 4.19 and are denominated as follows: the scheduler, the dispatcher, the message receiver, the context-aware data collector, and the timing. These behaviors are executed concurrently inside the structure of the agent. The proper functioning and availability of the context-aware data collector is fundamental for carrying out the tasks related to

the *Interpreter agent's* remaining behaviors. The context-aware data collector is in charge of collecting the sensor and control system data, and generates corresponding notifications and events derived from the collected information. The behavior of the message receiver and the timing process is very simply to add new events to the event list. The message receiver listens to the ACL messages sent by other agents and generates corresponding event messages. The timing eliminates the planned events once they have been initiated, and adds new events to the list for their initiation. The dispatcher is responsible for adopting the objectives, queuing them for execution, and selecting the plans that manage the events needed to fulfill the objective. The selected plans are executed step by step by the scheduler, which also manages the supervision plan. These behaviors shut down on their own and restart themselves to keep the agent from consuming resources unnecessarily. Implementing the *Interpreter agent's* functionalities as separate behaviors provides a clean design and allows for the flexible substitution of the behaviors with personalized implementations such as alternative planning mechanisms and CBR-BDI implementations that can be carried out using modified versions of the behaviors.

8.4 Applying HoCCAC to plan task the COPD patient in context-aware environment

For this case study, the HoCCAC system was used to develop a prototype for improving a patient's quality of life. The system collects information from the sensors, which capture data and interact with the patient by means of task plans developed by the *Interpreter agent*. The primary information gathered by the installed sensors is the location-aware information for the user in the environment. The system also gathers information regarding the temperature throughout the patient's home and the lighting conditions in the patient's ambulatory areas. For this prototype, HoCCAC is based on the information collected within the environment to plan the tasks for an COPD patient at home, and improve the current living conditions.

HoCCAC was employed to develop a prototype multi-agent system aimed at enhancing the assistance and care for low dependence patients at their homes. The house is 65 m² with a single dependent occupant. As shown in Figure 5.2 the home is

installed with (i) Simon Vit@ 81800-30 passive infrared motion detectors for roof and (ii) mechanisms for automatic door opening. The detectors' movements and mechanisms for opening doors interact with the *JavaCard* & RFID microchip [Duc *et al.,* 2006] and with users to offer services in run time. Each dependent user is identified by a (iii) Sokymat ID bracelet Band Unique Q5 equipped with an antenna and a RFID chip-Java-Crypto-Card with 32K Module and Crypto-CoProzessor (1024 bit RSA) compatible to the SUN JavaCard 2.1.1 [Chen *et al.,* 2004a]. The home also includes (iv) Simon Vit@81915-38 light sensors that are used in an effort to manage lighting levels in the home by maintaining the levels within a predetermined range of changeable values, (v) a Simon Vit@ 81221-38 TFT surface screen that displays and manages the primary system components, (vi) Simon Vit@ 81860-39 flood detectors for detecting water presence, (vii) Simon Vit@ 81861-39 gas detector designed to detect the presence of toxic gases and explosives, (viii) Simon Vit@ 81915-38 optical smoke detector for early warning of fires in progress, and a telephone control system. The sensors or actuators are placed in strategic locations throughout the home. All of the devices are controlled by agents. This sensor network is responsible for generating alarms upon comparing the user's current state with the parameters of the user's daily routine, which have been stored by the system. The system can generate alarms if it determines that the parameters fall outside the normal range. For example if the user, on a non-working day, stands up prior to a certain hour, if the user spends more time than specified at the front door of the home without entering, if the user remains motionless in the hallway for an extended period of time, etc.

The information *Provider agents* are directly connected to the information gathering devices. The application in this case is composed of six modules: (i) one to control the location of the patient, (ii) one to control the interior lighting, (iii) one to control temperature, (iv) one to control gas leaks, (v) one to control flood detection, and (vi) one to control smoke detection. The *LcA* is responsible for identifying and accepting or rejecting the data submitted by the information providers. Its task is to oversee the information provider agents that incorporate into the system. All of the data are stored in the system and interpreted by the *Interpreter agent*.

The *Interface agent* provides the *Interpreter agent* with a series of basic activities that the COPD patient must perform at home. This activity list is defined by the

medical personnel monitoring the patient. Table 5.1 shows the basic list of activities for a COPD patient.

Once the activity list has been defined with the appropriate instructions as provided by medical personnel, the *Interpreter agent* generates an order to create a task plan following the CPM method. With the activity list, the prerequisites for each activity, and the time requirement list, it is possible to determine the duration for executing the task plan and developing the network for the task plan. Table 5.2 shows the activity list and prerequisites.

Figure 5.11 shows the network associated with the previous task plan, with each of the nodes numbered, and the earliest and latest possible times.

According to the network in Figure 5.11, the activities can be found between two nodes where $ES_i = LS_i$ are critical tasks and determine the critical path. As such, the activities that compose the critical path in the network in Figure 5.11 are A, C, E, F, G, H, L, M and N. The remaining activities are carried out in parallel to the previous ones. For this reason, it is possible to delay the start of the non-critical activities, or increase their margin for delay without increasing the overall duration of the task plan. Once the *Interpreter agent* has defined the task plan and its duration, it sends this information to the rest of the HoCCAC system agents to execute the plan. The *Provider agents* along with the *Interface agent* are responsible for overseeing the completion of the task plan. If at any time the plan is interrupted, for example, because the patient is choking and has to perform the activity specified for this situation, the HoCCAC system is notified by the *Interface agent* and the *Interpreter agent* immediately replans the task plan. Another special case can result from a visit to the doctor, which takes place monthly. When there is no need to visit the doctor, the node representing this activity on the network is not present. Depending on the type of interruption, the *Provider agent* may also generate a warning as, for example, in the event that an oxygen tank does not function. Figure 5.12 shows the Gantt diagram with the solution associated with the task plan shown in Figure 5.11. It is also possible to deduce from Figure 5.12 which activities are critical and which are not. The critical activities are shown in red, while those with a margin for delay are in blue. It is also possible to see which tasks have a temporal overlap. Thus, the activity

of putting on the oxygen cylinder overlaps with waking up, doing exercise, and having breakfast. It is also possible to deduce that the visit to the doctor takes place after the previous activities, as well as after taking the morning medication.

During the sequence of operations carried out by the HoCCAC agents, it is necesary to consider the transformation of the information produced in the system. First of all, low level data are collected from the patient's environment. These are subsequently stored in the information system as high level data so that they are more quickly interpreted and easier to use. The information *Provider agents* work with the *Interpreter agent* to carry out this task. Additionally, the patient can use context-aware applications to interact with the context at all times and establish the parameters that dictate how the HoCCAC system functions.

8.5 Results and conclusions

Para HoCCAC was used to develop a prototype for the home of a dependent patient. It incorporates *JavaCard* technology to identify and control access, which provides an added value to existing RFID technology. The integration of these technologies provides the system with the capability to perceive stimuli from within the environment automatically and in execution time. This makes it possible to personalize the system's functioning and adjust it to the characteristics and needs of the context for any specific situation.

The CPM planning mechanism is appropriate and simple to implement in a dynamic and changing environment in which information systems require the incorporation of mechanisms that can obtain high levels of learning and adaptation. The proposed planning system differs from other existing systems in that it optimizes the possibility of assigning new plans and dynamic replanning in execution time. To obtain results and data from the evaluation of the CPM method used in our case study, we created a performance graph (Figure 5.13) which allows us to observe the rate of development for the activities described in the case study. At the same time, this graph enables us to know which activities are nearing completion. In the graph shown in Figure 5.13, the y-axis lists the percentage of completion for the task plan, while the x-axis lists the number of hours used for the defined task plan. Our case study

presented a task plan with a total duration of 18 hours and 5 minutes. This duration was calculated by adding together the times associated to the tasks that compose the critical path. This graph indicates the final task plan, which is found on the 100% efficiency path and the final time coordinate for the plan of actions.

At this point, it is now possible to calculate the advances achieved in the project every hour, as shown in Figure 5.13. The advance in the task plan is the advance of the sum of the advances achieved for each of the activities in the plan. Table 5.3 shows the advance reports for each of the activities in the defined case study. Immediately upon receiving information on the actual advance of the task plan, the *Interpreter agent* fills in the data in Table 5.3 in the following manner:

- Document the hour that the information is received (*Hour* column)
- Document the activity carried out during that hour in the *Activity* column
- Finally, the *% advance* column lists the advances the activities have made during that hour. The letter "T" appearing in the column means that the activity has concluded.

Given the data defined in Table 5.3, it would be possible at any time to confirm the advance in the task plan by adding together the partial advances achieved by the activities. This sum represents the actual advance of the task plan. It is important to note that 1 hour time intervals were used for the present case study because it is a daily task plan. However, there can be more specific and short task plans where the time intervals can be measured in minutes. The diagonal line in the graph shown in Figure 5.13 is completely straight because during the execution of the activity plan for the case study, there was no delay or acceleration in any activity. When the task plan has accelerations or delays, the resulting graph are not represented with a straight line, but with various arcs as shown in Figure 5.14. The graph in Figure 5.14 shows that there is an acceleration-delay in progress with the tasks in zone (a) and there is also a delay-acceleration in progress with the tasks in zone (b) with regards to the correct execution of the task plan.

HoCCAC functions like a global system for context-aware environments. HoCCAC not only captures information from the environment and interacts with users and their requests; it also, assisted by the information system, constantly

evaluates the attributes for the user context and provides proactive solutions. The solutions provided by HoCCAC are supported by a vast knowledge base that the system stores and processes. HoCCAC is a novel system compared to other studies related to context-aware [Abowd *et al.,* 2007] [Agarwal *et al.,* 2007] [Bettini *et al.,* 2009]. These studies concentrate on collecting location data for the user. Other studies, such as [Moon *et al.,* 2006], in addition to locating users within the context, attempt to improve the communication in a hospital center between patients and medical personnel by capturing context attributes such as time, the patient's state, or the user's role. However, the new services offered by HoCCAC allow for a closer, more natural and implicit interaction with the user. At the same time, the system provides the patient with proactive task plans that attempt to improve the patient's quality of life. The user can perform daily tasks and receive support from the intelligent context without the need for explicit interaction. As a result, the user is not required to learn to use the system. This increases the level of user satisfaction with the HoCCAC system managing the environment. Furthermore, the HoCCAC system evaluates the use of resources in the environment, sending useful reports to the medical personnel. Information on the use of resources is supplied by the HoCCAC *Provider agents*, while the *Interpreter agent* is responsible for evaluating the supplied information. The *Interpreter agent* uses this evaluation to determine how the daily resources will be allocated to the activities requesting their use. The *Interpreter agent* also examines its own evaluations and can reassign activities to available resources to shorten the duration of a task plan. This allows the HoCCAC system to progressively improve the solutions offered over time.

Furthermore, the technology used is not conditioned by the services offered. The choice of *JavaCard* and RFID to locate and identify objects and users is independent of the specific implementation of the technology; that is, the services are valid for any type of sensors, antennas, readers and RFID tags. Thus, the system is valid for very different types of environments whose devices need not have a fixed technology base. The incorporation of new *JavaCard* and RFID sensors in the case study significantly improves a series of functions such as controlling medication and diet, and detecting anomalies in the patient's behavior, as shown in Table 5.6.

Table 5.6 shows the percentage of success in controlling the three variables associated with the case study. One can easily appreciate the efficiency of the HoCCAC system in controlling the intake of medication and food for each patient with a 100% efficiency rate. The system avoids errors in the administration and control of medication and special diets for the patients. Furthermore, the HoCCAC system can detect anomalous behavior in the patients and create behavior patterns for high risk situations. The *Interpreter agent* takes past experiences into account in order to organize the new task plans for the system. In this regard, the HoCCAC system takes advantage of past experiences and learns from them. This behavior provides a great adaptive ability, as shown in the results in Table 5.6.

While there is yet much work to be done, the system prototype developed for this study improves home security for dependent persons through the use of vigilance and alert devices. The *Interpreter agent* in the HoCCAC system is capable of reacting automatically when faced with dangerous or emergency situations, replanning any plans in progress and sending alert messages to the system. Thus HoCCAC creates a context-aware environment that facilitates the development of distributed intelligent systems and provides services to dependent individuals in their home. This makes it possible to automate certain tasks used to monitor dependent persons and improve their quality of life. The use of multiagent systems, automatic control and RFID systems, *JavaCard* and mobile devices provides a high level of interaction between the context, caregivers and patients. Furthermore, the proper use of mobile devices facilitates social interactions and transfer of knowledge. Our future work will focus on obtaining a model to define the context, as well as improving the current prototype and its testing with patients suffering from other diseases or deficiencies.

Bibliografía

Aaløkke Ballegaard, S., Bunde-Pedersen, J., Bardram, J.E. (2006). Where to, Roberta?: reflecting on the role of technology in assisted living. *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles*, vol. 189, pp. 373-376. ACM.

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

Abowd, G.D., Patel, S.N., Truong, K.N., Iachello, G., Hayes, G.R., Poole, E.S., Grimes, A. (2007). Physical, social, and experiential knowledge in pervasive computing environments. IEEE Pervasive Computing Vol. 6, no. 4, pp 56-63.

Agarwal, S., Joshi, A., Finin, T., Yesha, Y., Ganous, T. (2007). A Pervasive Computing System for the Operating Room of the Future, *Mobile Netw. App.* 12:215-228.

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

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*.

Anumba, C., Aziz, Z. (2006). Case Studies of Intelligent Context-Aware Services Delivery in AEC/FM. LNCS: *Intelligent Computing in Engineering and Architecture*, Vol. 4200/2006, pp. 23-31: Springer Berlin / Heidelberg.

Artikis, A., Kaponis, D. and Pitt. J., (2009). Multi-Agent Systems: Semantics and Dynamics of Organisational Models, chapter in Dynamic Specications of Norm-Governed Systems. IGI Globa.

AUML.org. (2010). *The FIPA Agent UML Web Site*, from http://www.auml.org

Bahadori, S., Iocchi, L., Leone, G. R., Nardi, D. and Scozzafava, L. (2007). Real-time people localization and tracking through fixed stereo vision, *Applied Intelligence*, vol. 26, no. 2, pp. 83-97.

Bai, Y., Yang, J., Qiu, Y. (2008). OntoCBR: Ontology-based CBR in Context-aware Applications. *International Conference on Multimedia and Ubiquitous Engineering*, pp. 164-16, IEEE Computer Society.

Bajo, J., De Paz, Y., De Paz, J.F., Martín, Q., Corchado, J.M. (2006). SMas: A Shopping Mall Multiagent Systems. *Proccedings of IDEAL'06, Lecture Notes in Artificial Intelligence (LNAI). 4224*, pp. 106-117. Springer-Verlag.

Bajo, J., De Luis, A., Tapia, D.I., Corchado, J.M. (2006b). 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., Tapia, D.I., De Luis, A., Corchado, J.M. (2007). Nature-Inspired Planner Agent for Health Care. *Proceedings of IWANN'07, Lecture Notes in Artificial Intelligence (LNAI)* (pp. 1090-1097). Springer-Verlag.

Bajo, J., Fraile, J.A., Pérez-Lancho, B. and Corchado, J.M. (2010). The THOMAS Architecture in Home Care Scenarios: A case study. *Expert Systems with Applications, 37 (5).* pp. 3986-3999.

Baldauf, M., Dustdar, S. (2004). A survey on context-aware systems. Technical Report TUV-1841-2004-24, Technical University of Vienna.

Baus, J., Cheverst, K., Kray, C. (2005). A Survey of Map-based Mobile Guides. En L. Meng, A. Zipf, T. Reinchenbacher (eds.) *Map-based Mobile Services*, pp. 193-209. Berlín: Springer.

Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomez-Sanz, J.J., Pavon, J. and Gonzalez-Perez, C. (2009) 'FAML: a generic meta-model for MAS development', *IEEE Trans. Softw. Eng.*, Vol. 35, No. 6, pp.841-863.

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

Benerecetti, M., Bouquet, P., and Ghidini, C., (2007). On the dimensions of context dependence. In P.Bouquet, L.Sera_ni, and R.H. Thomason, editors, Perspectives on Contexts, CSLI Lecture Notes, chapter 1, pages 1-18. Center for the Study of Language and Information/SRI.

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D. (2009). A Survey of Context Modelling and Reasoning Techniques. Pervasive and Mobile Computing.

Biegel, G. (2004). A Framework for Developing Mobile, Context-aware Applications, in Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04). IEEE Computer Society.

Boissier, O., and Gateau, B., (2007). Normative multi-agent organizations: Modeling, support and control. In Normative Multiagent Systems.

Bordini, R.H., Dastani, M., Dix, J. and El Fallah Seghrouchni, A. (2005). *Multi-Agent Programming: Languages, Platforms and Applications*. Number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer.

Bou, E., López-Sánchez, M., Rodríguez-Aguilar, J.A., (2008). Using Case-Based Reasoning in Autonomic Electronic Institutions. Coordination, Organizations, Institutions, and Norms in Agent Systems III, Lecture Notes in Computer Science. Editor Springer Berlin / Heidelberg. ISSN 0302-9743 (Print) 1611-3349 (Online). Volume 4870/2008. DOI 10.1007/978-3-540-79003-7. ISBN 978-3-540-79002-0. Páginas 125-138.

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.

Brenner, T. (2006). Agent learning representation: Advice on modelling economic learning, Handbook of computational economics – Elsevier.

Bricon-Souf, N., Newman, C.R. (2007). Context awareness in health care: A review. *International Journal of Medical Informatics*, 76, 2-12.

Brown, S. W., Griswold, W. G., Demchak, B., Lenert, L. (2006). Middleware for Reliable Mobile Medical Workflow Support in Disaster Settings, *AMIA'06: Proceedings of the American Medical Informatics Association Annual Fall Symposium*. pp. 309-13.

Bry, F., Hattori, T., Hiramatsu, K. (2005). Context Modeling in OWL for Smart BuildingServices. *http://dbs.informatik.unihalle.de/GvD2005/beitraege/gvd05_BrHaHiOkWiYa.pdf*

Büscher, M., Coulton, P., Efstratiou, C. et al., (2009). Intelligent mobility systems: some socio-technical challenges and opportunities, in *Communications Infrastructure: Systems and Applications in Europe*, R. Mehmood, E. Cerqueira, R. Piesiewicz, and I. Chlamtac, Eds., pp. 140-152.

Buttussi, F., Chittaro, L. (2008). MOPET: A *context-aware* and user-adaptive wearable system for fitness training. *Arificial Intelligence in Medicine.* Vol. 42, Is. 2, pp. 153-163.

Camarinha-Matos, L. M. and Afsarmanesh, H. (2008). Collaborative networks – Reference modeling. New York: Springer.

Carrascosa, C., Bajo, J., Julian, V., Corchado, J. M., Botti, V. (2008). Hybrid multi-agent architecture as a real-time problem-solving model. Expert Systems With Applications. Volumen 34. Numero 1. pp. 2-17.

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

Carriero, N., Gelernter, D. (1989). Linda in context. Communications of the ACM. 32(4): pp. 444-458.

Castro-Lacouture, D.; Süer, G. A.; Gonzalez-Joaqui, J.; and Yates, J. K. (2009). Construction project scheduling with time, cost, and material restrictions using fuzzy mathematical models and critical path method. Journal of Construction Engineering and Management, 135(10), 1096-1104.

Cesta, A., Cortellessa, G., Pecora, F. and Rasconi R. (2007). Supporting Interaction in the RoboCare Intelligent Assistive Environment. In *Proceedings of AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants* (pp. 18-25) Stanford: AAAI.

Chen, H. (2004a). An Intelligent Broker Architecture for Pervasive Context-Aware Systems. University of Maryland: Baltimore County.

Chen, H., Finin, T., Joshi, A. (2004b). An ontology for context-aware pervasive computing environments. Knowl. Eng. Rev. 18(3): pp. 197-207.

Cheung, R. (2005). An Adaptive Middleware Infrastructure Incorporating Fuzzy Logic for Mobile Computing. *Proc. of the International Conference on Next Generation Web Services Practices (NWeSP'05)*, pp. 3, IEEE Computer Society.

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.

Consolvo, S., Klasnja, P., McDonald, D.W., Avrahami, D., Froehlich, J., LeGrand, L., Libby, R., Mosher, K., and Landay, J.A. (2008). Flowers or a Robot Army? Encouraging Awareness and Activity with Personal, Mobile Displays, In Proceedings of the International Conference on Ubiquitous Computing (Seoul, Korea). pp. 54-63.

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. (2008a). Intelligent Environment for Monitoring Alzheimer Patients, Agent Technology for Health Care. *Decision Support Systems.* ISSN 0167-9236.Vol 34 (2) pp. 382-396.

Corchado, J.M., Bajo, J., Abraham, A. (2008b). GERAmI: Improving the delivery of health care. *IEEE Intelligent Systems. Special Issue on Ambient Intelligence - Mar/Apr 08*.

Corchado, J.M., Fraile, J.A., Bajo, J., Pérez-Lancho, B. (2011). HoCa: Multiagent Architecture for Developing Intelligent Home Care Environments. Applied Soft Computing. In press 2011.

Costa de, C.A., Yamin, A.C., Resin Geyer, C.F. (2008). Toward a General Software Infrastructure for Ubiquitous Computing, *Pervasive computing*, 2008, 1, p. 64-74

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

Dey, A. K. and Newberger, A. (2009). Support for context-aware intelligibility and control. *CHI 09*, 859-868.

Donghai, G., Weiwei, Y., Sungyoung, L., Young-Koo, L., (2007). Context Selection and Reasoning in Ubiquitous Computing, in Proceedings of the The 2007 International Conference on Intelligent Pervasive Computing. IEEE Computer Society.

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.

Dundas, G.R., Krentler, K.A. (1982). Critical path method for introducing an industrial product, Industrial Marketing Management. 11 (2) 125-131.

Emiliani, P.L., Stephanidis, C. (2005). Universal access to ambient intelligence environments: Opportunities and Challenges for people with disabilities. *IBM Systems Journal*, vol. 44, No. 3.

Ermi, L., Mäyrä, F. (2005). Challenges for Pervasive Mobile Game Design: Examining Players. Emotional Responses. *Proc. of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, pp. 371 – 372, Nueva York: ACM.

Fernández, F., Rodillo, D., del Pino, F., Bajo, J., Corchado, J. M. (2007). WirePET: Desarrollando Videojuegos para Dispositivos Móviles con Comunicación Bluetooth. 6th International Workshop on Practical Applications of Agents and Multiagent Systems. J. Bajo, V. Alonso, L. Joyanes and J.M. Corchado (Eds.)- Universidad Pontificia de Salamanca. Pág. 259-268.

Fernández, T., Rodas, J., Escudero, C., Iglesia, D. (2007b). Bluetooth Sensor Network Positioning System with Dynamic Calibration. *Proc. of the 4th International Symposium on Wireless Communication Systems,* pp. 45-49, IEEE Computer Society.

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

Fraile, J.A., Tapia, D.I., Sánchez, M.A. (2008). Hybrid Multi-Agent Architecture (HoCa) applied to the control and supervision of patients in their homes. Hybrid Artificial Intelligence Systems. Third International Workshop, HAIS 2008, Burgos, Spain, Proceedings. Lecture Notes in Artificial Intelligence (LNAI), Springer Verlag, Vol 5271. E. Corchado, A. Abraham, and W. Pedrycz (Eds.). pp. 54-61

Fraile, J.A., Bajo, J. and Corchado, J.M. (2008b). Hybrid Multi-Agent Architecture (HoCa) applied to the control and supervision of patients in their home. Advances in Artificial Intelligence - IBERAMIA 2008. 11th Ibero-American Conference on AI, Lisbon. Lecture Notes in Artificial Intelligence (LNAI), Springer Verlag, Vol 5290. Geffner, H.; Prada, R.; Machado Alexandre, I.; David, N. (Eds.) pp. 193-202

Fraile, J.A., Bajo, J., Corchado, J.M. (2009a). Multi-Agent Architecture for Dependent Environments. Providing Solutions for Home Care. Inteligencia Artificial. Special Issue 7th Ibero-American Workshop in Multi-Agent Systems (Iberagents 2008), 13(42). pp. 36-45.

Fraile, J.A., Bajo, J., Corchado, J.M. (2009b). Applying Context-Aware Computing in Dependent Environments. Bioinspired Applications in Artificial and Natural Computation Volume 5602/2009 IWINAC (2). Springer Berlin. pp. 85-94.

Fraile, J.A., Tapia, D.I., Rodríguez, S., and Corchado, J.M. (2009c). Agents in Home Care: A Case Study. Proceedings of IWANN'09. Lecture Notes in Artificial Intelligence (LNAI), Springer Verlag. J. Cabestany et al., Springer Verlag. Lecture Notes in Artificial Intelligence (LNAI), Springer Verlag. Vol 5517. pp. 1–8.

Fraile, J.A., Román, J.A., Pérez-Lancho, B. (2010a). Integration Mechanism in the HoCa Multi-Agent Architecture. Trends and strategies on agents and multiagent

systems: 8th International Conference on Practical Applications of agents and multiagent systems. Pawlewski, P.; Julián, V.J.; Fernández-Riverola, F.; Corchado, E.; Corchuelo, R.; Bajo, J.; Corchado, J.M.; Dignum, F.; Demazeau, Y.; Campbell, A. (Eds.). Vol.71/2010 pp. 303-310.

Fraile, J.A., Tapia, D.I., Román, J.A., García, O. (2010b). Agents to help Context-aware system in Home Care. Trends and strategies on agents and multiagent systems: 8th International Conference on Practical Applications of agents and multiagent systems. Pawlewski, P.; Julián, V.J.; Fernández-Riverola, F.; Corchado, E.; Corchuelo, R.; Bajo, J.; Corchado, J.M.; Dignum, F.; Demazeau, Y.; Campbell, A. (Eds.). Vol.71/2010 pp. 501-508.

Fraile, J.A., Zato, C., Gil, O., De la Prieta, F. (2010c). Multiagent systems and wearable devices: Helping people live healthier. Ambient Intelligence and Future Trends-International Symposium on Ambient Intelligence (ISAmI 2010). Springer Velag. Advances in Soft Computing Series. Vol. 72. pp. 11-18.

Fraile, J.A., Bajo, J., Corchado, J.M., Abraham, A. (2010d). Applying wearable solutions in dependent environments. IEEE Transactions on Information Technology in Biomedicine. Vol. 14(6). pp. 1459-1467.

Franklin, S. (2006). The LIDA architecture: Adding new modes of learning to an intelligent, autonomous, software agent. In *Proc. of the Int. Conf. on Integrated Design and Process Technology*. San Diego, CA: Society for Design and Process Science.

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

Garcia-Ojeda, J.C., DeLoach S.A., Robby, O.W.H., Valenzuela J. (2008). O-MaSE: a customizable approach to developing multiagent development processes. In: Luck M, editor. Agent-oriented software engineering VIII: the 8th International Workshop on Agent Oriented Software Engineering (AOSE 2007); Berlin. Springer-Verlag. pp. 1-15.

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

Gascueña, J.M. and Fernández-Caballero, A. (2008). Prometheus and INGENIAS agent methodologies: A complementary approach. In Michael Luck and Jorge J.

G_omez-Sanz, editors, AOSE, volume 5386 of Lecture Notes in Computer Science, pages 131-144. Springer.

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.

Giorgini, P., Mylopoulos, J., Perini, A. and Susi, A. (2011). The Tropos Methodology and Software Development Environment. In P. Giorgini, N. Maiden, J. Mylopoulos, and E. Yu, editors, Social Modelling for Requirements Engineering. MIT Press.

Griswold, W.G., Shanahan, P., Brown, S.W., Boyer, R.T., Ratto, M., Shapiro, R.B., and Truong, T.M. (2004). ActiveCampus - Experiments in Community-Oriented Ubiquitous Computing. *IEEE Computer*, 37(10), pp. 73-81.

Gupta, T. (1991) Applying the critical path method to manufacturing routing. Computers & Industrial Engineering. 21 (1-4) pp 519-523.

Gwizdka, J. and Cole, M. (2007). Finding it on Google, finding it on del.icio.us. In L. Kovács, N. Fuhr & C. Meghini (Eds.), *Research and advanced technology for digital libraries* (pp. 559-562). Berlin and Heidelberg: Springer-Verlag. (Lecture Notes in Computer Science LN4675).

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H. (2009). The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

Hasan, S.S., Isaac, R.K. (2011). An integrated approach of MAS-CommonKADS, Model–View–Controller and web application optimization strategies for web-based expert system development. *Expert Systems with Applications*, Vol. 38, No. 1, pp. 417-428.

Haya, P.A., Montoro, G. (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.

Henricksen, K., Indulska, J. (2006). Developing context-aware pervasive computing applications: Models and approach. Journal of Pervasive and Mobile Computing,

2(1):37–64.Hensel, B.K., Demiris, G., Courtney, K.L. (2006). Defining Obtrusiveness in Home Telehealth Technologies: A Conceptual Framework. *Journal of the American Medical Informatics Association*, 13(4): 428-431.

Hofer, T., et al., (2003). Context-Awareness on Mobile Devices - the Hydrogen Approach, in Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9 - Volume 9. IEEE Computer Society.

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.

IST Advisory Group (ISTAG) (2005). *Ambient Intelligence: from Vision to Reality*. In Riva, G., Vatalaro, F., Davide, F., Alcañiz, M. (Eds.) Chapter 4 in *Ambient Intelligence*. IOS Press, 45-68.

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

Iso.org. (2010). JTC 1 Information technology. ISO Standards. Committee Technology.

Jbara, S., Kuflik, T., Soffer, P., Stock, O. (2007). Context Aware Communication Services in "Active Museums". *Proc. IEEE International Conference on Software – Science, Technology and Engineering*, pp. 127-135, IEEE Computer Society.

Jiang, X., Chen, N.Y., Hong, J.I., Wang, K., Takayama, L., Landay, J.A. (2004). Siren: Context-aware Computing for Firefighting. *LNCS: Pervasive Computing*, Vol. 3001/2004, pp. 87-105, Springer.

Johanson, B., Fox, A., Winograd, T. (2002). The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. *IEEE Pervasive Computing*, Vol. 1, Is. 2, pp. 67-74.

Kleinberger, T., Becker, M., Ras, E., Holzinger, A., Müller, P. (2007). Ambient Intelligence in Assisted Living: Enable Elderly People to Handle Future Interfaces. En *LNCS 4555*, pp. 103-112.

Koskinen, K., Suomela, R. (2006). Rapid prototyping of *context-aware* games, IEEE *2nd International Conference on Intelligent Environments,* Vol. 1, pp. 135-142.

Kurkovsky, S., Zanev, V., Kurkovsky, A. (2005). SMMART, a Context-Aware Mobile Marketing Application: Experiences and Lessons. En *Embedded and Ubiquitous Computing*, LNCS Vol. 3823/2005, pp. 141-150, Springer.

Kwak,Y. H. and Anbari, F. T. (2008). Project management research trends of allied disciplines. Presented at the 2008 PMI Research Conference*,* Warsaw, Poland.

Lagoudakis, M.G.,Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., Koenig, S., Tovey, C., Meyerson, A. and Jain, S. (2005). Auction-Based Multi-Robot Routing, In Robotics: Science and Systems (RSS).

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

Lee, J.Y., Rhee, G.W., Kim, H., Lee, K-W., S. Y-H. (2006). Convergence of Context-Awareness and Augmented Reality for Ubiquitous Services and Immersive Interactions. *Computational Science and Its Applications - ICCSA 2006*, LNCS Vol. 3983/2006, Springer.

Lim, T., Lee, J., Shin, S. (2007). Context Aware System for Future Homes. En *Advances in Intelligent Web Mastering,* LNCS Vol. 43/2007, pp. 210-216, Springer.

Lina, M., Lin, Z. (2006). A cost-effective critical path approach for service priority selections in grid computing economy, Decision Support Systems. 42 (3) 1628-1640.

Lui, T. (2009). Automation in Home Appliances, en Springer Handbook of Automation, Springer, pp. 1469-1483.

Luyten, K., Winters, F., Coninx, K., Naudts, D., Moerman, I. (2006). A Situation-Aware Mobile System to Support Fire Brigades in Emergency Situations. En *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, LNCS Vol. 4278/2006. Springer.

Ma, T., Kim, Y., Ma, Q., Tang, M., Zhou, W. (2005). Context-Aware Implementation based on CBR for Smart Home. *IEEE Int. Conf. on Wireless and Mobile*

Computing, Networking and Communications (WiMob 2005), Vol. 4, pp. 112-115.

Malanowski, N., Özcivelek, R., Cabrera, M. (2008). Active Ageing and Independent Living Services: The Role of Information and Communication Technology. EUR 23246 EN-2008. European Commission, IPTS. Consultado en mayo 2008 en: *http://ftp.jrc.es/JRC41496.pdf*

Martinez-Ruiz, F.J., Vanderdonckt, J., and Martinez-Ruiz, J., (2008). Context-Aware Generation of User Interface Containers for Mobile Devices, ENC '08: Proceedings of the 2008 Mexican International Conference on Computer Science, 2008, 63-72, Washington, DC, USA.

Mas, A. (2005). Agentes software y Sistemas Multi-Agente: Conceptos, Arquitecturas y Aplicaciones. Pearson-Prentice Hall.

Meier, R., Harrington, A., Cahill, V. (2006). Towards Delivering Context-Aware Transportation User Services. *Proc. of the IEEE Intelligent Transportation Systems Conference*, pp. 369-376, IEEE Computer Society.

Miori, V., Tarrini, L., Manca, M., Tolomeo, G. (2005). An innovative, open-standards solution for Konnex interoperability with other domotic res. *Konnex Scienti* middlewa*fic Conference 2005.* Pisa, Italia 213.

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.

Moon, A., Kim, H. and Lee, S. (2006). Designing CAMUS based Context-Awareness for Pervasive Home Environments, International Conference on Hybrid Information Technology.

Moreno, A., Valls, A., Isern, D., Sanchez, D. (2006). Applying Agent Technology to Healthcare: The GruSMA Experience. *Intelligent Systems IEEE*, pp. 63-67.

Nasri Nazif, A., Davoodi, A. and Pasquier, P. (2011). *Multi-Agent Area Coverage Using a Single Query Roadmap: A Swarm Intelligence Approach*. ADVANCES IN PRACTICAL MULTI-AGENT SYSTEMS. Studies in Computational Intelligence, 2011, Volume 325/2011, pp. 95-112.

Nfc-forum.org (2010). NFC Forum Specifications.

Niemelä, M., González Fuentetaja, R., Kaasinen, E., Lorenzo Gallardo, J. (2007). Supporting Independent Living of the Elderly with Mobile-Centric Ambient Intelligence: User Evaluation of Three Scenarios. En *Ambient Intelligence,* LNCS Vol. 4794, pp. 91-107, Springer.

Ossowski, S. (2008). Coordination and agreement in multi-agent systems, in *Proceedings of Cooperative Information Agents XII, 12th International Workshop (CIA 2008)*, ser. Lecture Notes in Computer Science, M. Klusch, M. Pechoucek, and A. Polleres, Eds., vol. 5180. Springer, pp. 16-23.

Pascoe, J., Thomson, K., and Rodrigues, H. (2007). Context-Awareness in theWild: An Investigation into the Existing Uses of Context in Everyday Life. In OTM Workshops, pp. 193-202.

Pecora, F., Cesta, A. (2007). Dcop for smart homes: A case study. *Computational Intelligence , 23* (4), pp. 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)*, pp. 76-85.

Poole, I. (2004). What exactly is … ZigBee? *Communications Engineer , 2* (4), 44-45.

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. (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.

Rao, A.S., Georgeff, M.P. (1995). BDI agents: from theory to practice. *In Proceedings of the First International Conference on MultiAgents Systems (ICMAS'95).* San Francisco, CA, USA.

Rekimoto, J., Ayatsuka, Y., Hayashi, K., (1998). Augment-able Reality: Situated Communication through Physical and Digital Spaces, in Proceedings of the 2nd

IEEE International Symposium on Wearable Computers. IEEE Computer Society.

Reynolds, F. (2006). *The Ubiquitous Web, UPnP and Smart Homes*. Cambridge, UK: a Pervasive Computing Group. Nokia Reserch Center.

RoboCare. (2005). *The RoboCare Project. Institute of Cognitive Sciences and Technologies*. Retrieved from http://robocare.istc.cnr.it

Russell, S., Norvig, P. (1995). *Artificial Intelligence: A modern approach.* Englewood Cliffs, NJ, USA: Prentice-Hall Series in Artificial Intelligence.

Ryan, N., Pascoe, J., Morse, D. (1998). Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant. In Computer Applications in Archaeology, V.G.a.M.v.L.a.S. Exxon, Editor. Tempus Reparatum.

Sadeh, N., Gandon, F., Kwon, O. (2005). Ambient Intelligence: The MyCampus Experience, *Technical Report CMU-ISRI-05-123*, School of Computer Science, Carnegie Mellon University.

Salsbury, T. (2009). The Smart Building, en Springer Handbook of Automation, Springer, pp. 1079-1093.

Sánchez, D., Tentori, M., Favela, J. (2007). Hidden Markov Models for Activity Recognition in Ambient Intelligence Environments. *Proc. Eight IEEE.Mexican International Conference on Current Trends in Computer Science*, pp. 33-40, IEEE Computer Society.

Gómez-Sanz, J.J., Fuentes, R., Pavón, J., García-Magariño, I. (2008). INGENIAS development kit: a visual multi-agent system development environment. In: AAMAS (Demos), pp. 1675-1676.

Schilit, B., Adams, N. Want, R., (1994). Context-Aware Computing Applications, in 1st International Workshop on Mobile Computing Systems and Applications. pp. 85-90.

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.

SDSU, (2005). Radio Frequency Identification. *Masters of Homeland Security*. San Diego, CA, USA: San Diego State University.

Sheng, Q.Z., Pohlenz, S., Yu, J., Wong, H.S., Ngu, A.H.H., and Maamar, Z., (2009). ContextServ: A Platform for Rapid and Flexible Development of Context-AwareWeb Services, in *Proc. of IEEE International Conference on Software Engineering*.

Shokouhi, S. V., Aamodt, A., Skalle, P., Sørmo, F. (2009). Comparing two types of knowledgeintensive CBR for optimized oil well drilling, Proceedings of the 4th Indian International Conference on Artificial Intelligence (IICAI-09), Tumkur India.

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.

Susperregi, L., Maurtua, I., Tubío, C., 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.

Syvänen, A., Beale, R., Sharples, M., Ahonen, M., Lonsdale, P. (2005). Supporting Pervasive Learning Environments: Adaptability and Context Awareness in Mobile Learning. *Proc. Of the 2005 IEEE International Workshop on Wireless and Mobile Technologies in Education (WMTE'05)*, pp. 251-253, Washington: IEEE Computer Society.

SysML.org. (2010, Junio). OMG System Modelling Language (OMG SysML) V1.2. *OMG Available Specification*.

Tähti, M., Rautio, V-M., Arhippainen, L. (2004). Utilizing Context Awareness in OfficeType Working Life. *Proc. of the 3rd international conference on Mobile and ubiquitous multimedia*, Vol. 83, pp. 79-84, Nueva York: ACM.

Tapia, D.I., Cueli, J.R., García, O., Corchado, J.M., Bajo, J., Saavedra, A. (2007). Identificación por Radiofrecuencia: Fundamentos y Aplicaciones. *Proceedings de las primeras Jornadas Científicas sobre RFID.* Ciudad Real, España.

Tapia, D.I., Corchado, J.M. (2009). An Ambient Intelligence Based Multi-Agent System for Alzheimer Health Care. International Journal of Ambient Computing and Intelligence (IJACI), Vol. 1, núm 1, pp. 15-26.

Tapia, D.I., Fraile, J.A., Rodríguez, S., Alonso, R.S. and Corchado, J.M. (2011). Integrating Hardware Agents into an Enhanced Multi-Agent Architecture for

Ambient Intelligence Systems. Information Sciences ISSN: 0020-0255 Imprint: ELSEVIER. In press.

Tentori, M., Favela, J. (2008). Activity-aware computing for healthcare. IEEE Pervasive Computing, 7(2), pp. 51-57.

Troyk, P. R. (1999). Injectable Electronic Identification, Monitoring, and Stimulation Systems. *Annual Review of Biomedical Engineering , 1*, pp. 177-209.

UML.org (2010). *Unified Modelling Language Specification v2.0.* The Object Management Group http://www.uml.org

USDC, (2005). *Radio Frequency Identification. Opportunities and Challenges in Implementation.* Washington, DC, USA: U.S. Department of Commerce.

Vázquez, I., López de Ipiña, D. (2005). Inteligencia Ambiental: la presencia invisible. *Revista solo programadores* (127), pp. 16-19.

Vergara, M., Díaz-Hellín, P., Fontecha, J., Hervás, R., Sánchez Barba, C., Fuentes, C., Bravo, J. (2010). Mobile Prescription: An NFC-Based Proposal for AAL. In 2nd International Workshop on Near Field Communication. Monaco.

Verichip. (2006). VeriMed Patient Identification. Verichip Corporation.

Weiβenberg, N., Gartmann, R. (2004). Ontology Architecture for Semantic Geo Services for Olympia 2008. *Conference of GI-Days*, pp. 267-283.

Walton, C. A. (1983). *Patent No. 4384288.* Los Gatos, CA, USA.

Want, R. (2006). An introduction to RFID technology. *IEEE Pervasive Computing, 1* (5), pp. 25-33.

Weber, W., Rabbaey, J., Aarts, E. (eds.) (2005). *Ambient Intelligence*. Berlín: Springer-Verlag.

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), pp. 75-84.

Weyns, D., Omicini, A., Odell, J. (2007). Environment as a first class abstraction in multiagent systems. Autonomous Agents and Multi-Agent Systems 14(1) pp. 5–30.

Will, W. (2011). LinuxTuples. Available from: http://linuxtuples.sourceforge.net/.

Wooldridge, M., Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review , 10* (2), pp. 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 Sysems*), pp. 285-312.

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems.* Chichester, England: John Wiley y Sons.

Wu, C., Liao, C., Fu, L. (2007). Service-Oriented Smart-Home Architecture Based on OSGi and Mobile-Agent Technology, IEEE Trans. on Systems, Man and Cybernetics 37(2).

Yamabe, T., Takagi, A., Nakajima, T., (2005). Citron: A Context Information Acquisition Framework for Personal Devices, in Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications. IEEE Computer Society.

Yan, Z., Wei, C., Lu, A., and Ebert, D.S. (2009). Context-aware Volume Modeling of Skeletal Muscles. *Computer Graphics Forum,* Volume 28, Issue 3, pp. 887-894.

Yoerger D.M., Marcus F., Sherrill D., Calkins H., Towbin J.A., Zareba W., et al., (2005). *Echocardiographic findings in patients meeting task force criteria for arrhythmogenic right ventricular dysplasia: new insights from the multidisciplinary study of right ventricular dysplasia.* J Am Coll Cardiol 45, pp. 860-865.

Zhang, J., Zhan, S., Ji, Y., Zhang, P. (2006) Reasoning Mechanism Based on the Demspter-Shafer Evidence Theory for Pervasive Context Aware Computing Application. *Proc. of the 1st Int. IEEE Conf. on Communication and Networking in China*, pp. 1-3. doi:10.1109/CHINACOM.2006.344804.

Zhiming Zhao Belloum, A., de Laat, C., Adriaans, P., Hertzberger, B. (2007). Using Jade agent framework to prototype an e-Science workflow bus. In Cluster Computing and the Grid. Seventh IEEE International Symposium on, pp. 655-660.

ZigBee. (2006). *ZigBee Specification Document 053474r13.* ZigBee Standards Organization. ZigBee Alliance.

A

Tecnologías Inalámbricas

A.1 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] [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. [ZigBee, 2006].

Como se muestra en la Tabla A.1, 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 WiFi [ZigBee, 2006] [Kinney, 2003] [Poole, 2004]. La distancia de transmisión entre dispositivos varía desde los 10m hasta los 75m [Poole, 2004].

Tabla A.1 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 Wireless Personal Area Network (WPAN) 802.15.4 de la IEEE. IEEE 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.1 **[**ZigBee, 2006].

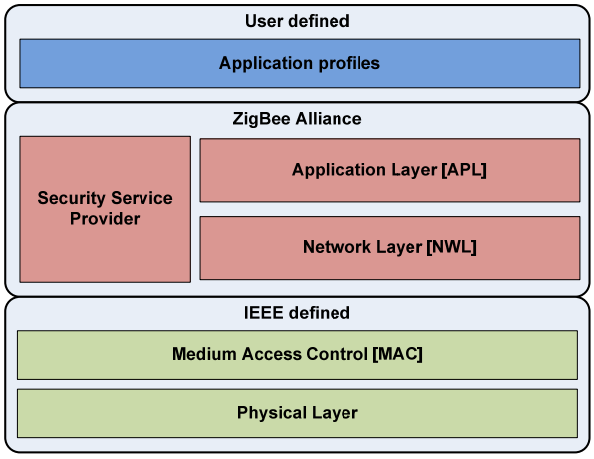


Figura A.1 Arquitectura en capas de ZigBee.

Este estándar define dos tipos de dispositivos [ZigBee, 2006] [Kinney, 2003]:

- ***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*).
  - ***Personal Area Network* (PAN) *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.

- ***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.2, ZigBee soporta las topologías de estrella, árbol y malla [Kinney, 2003] [ZigBee, 2006].

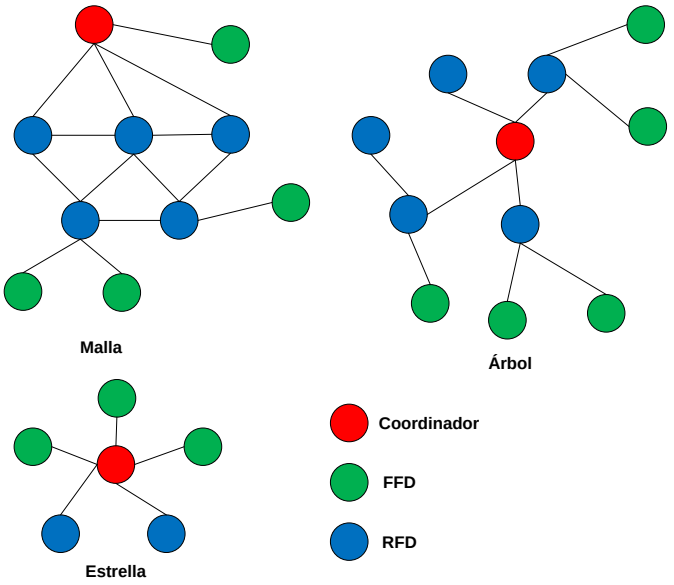


Figura A.2 Topologías soportadas por ZigBee.

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 la mínima 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]. 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 15 milisegundos 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 [Freescale, 2007].

A.2 WiFi

Las redes *Wireless Fidelity* (WiFi) 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 WiFi cuenten con ciertos requisitos de interoperabilidad y deben ser certificados por la organización WiFi Alliance, garantizando su compatibilidad con cualquier otro dispositivo certificado sin importar el fabricante [Hewlett-Packard, 2002]. Todos ellos funcionan bajo el estándar 802.11, publicado en 1999 por el comité de estándares *Local Area Network / Metropolitan Area Network* (LAN/MAN) de la *Institute of Electrical and Electronics Engineers* (IEEE). Este estándar especifica las características que deben cumplir la capa física y

la capa *Media Access Control* (MAC) del modelo de referencia *Open Systems Interconection* (OSI) de la *International Organization for Standarization* (ISO), descrito en la Tabla A.2, para redes inalámbricas de área local (WLAN). El principal objetivo de WiFi consiste en proporcionar conectividad inalámbrica a dispositivos que requieren cierta movilidad dentro de una LAN [IEEE, 2003a] [Hewlett-Packard, 2002].

Tabla A.2 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.

Para la capa física, se define una tasa de transferencia de datos de hasta 2Mbps, utilizando Radio Frecuencia (RF) en la banda de los 2.4GHz, ya sea mediante el método *Frequency Hopping Spread Spectrum* (FHSS) o *Direct Sequence Spread Spectrum* (DSSS). 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 *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) en lugar del protocolo *Carrier Sense Multiple Access with Collision Detect* (CSMA/CD) ya que en medios inalámbricos no hay manera práctica de transmitir y recibir datos al mismo tiempo [IEEE, 2003a].

A partir del estándar WiFi se han derivado otras especificaciones, algunas de las principales descritas en la Tabla A.3. Estas especificaciones mejoran diversos aspectos del estándar original, como el rendimiento, la seguridad y la confiabilidad de las transmisiones, entre otras [Hewlett-Packard, 2002] [Sun Microsystems, 2000].

El 802.11a define la capa física para tasas de transferencia entre 6 y 54Mbps a 5GHz [IEEE, 2003a], utilizando *Orthogonal Frequency - Division Multiplexing* (OFDM). 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 A.3 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]. 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 *Orthogonal Frequency-Division Multiplexing* (OFDM) [IEEE, 2003c]. Ofrece un menor coste y menor pérdida de trayectoria que el 802.11a, mayor seguridad, y es compatible con el 802.11b [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 *Basic Service Set* (BSS) está formada cuando dos o más estaciones se reconocen entre ellas y establecen una red. Esta red puede configurarse de dos maneras [Hewlett-Packard, 2002]:

- **Modo *Peer-to-peer* o *Ad Hoc*.** Conocido también como *Independent Basic Service Set* (IBSS), 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 [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 A.3 se muestran ejemplos de patrones de propagación de la señal en diferentes tipos de antenas.

Figura A.3 Tipos de antenas.

El crecimiento en la implementación de redes inalámbricas presenta nuevos retos para los administradores de redes, entre ellos la seguridad [Cisco Systems, 2005]. Una red inalámbrica debe ser tratada como insegura, por lo que es necesario implementar mecanismos de protección de información y de autenticación de usuarios.

A.3 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.

Un sistema RFID se compone principalmente de cuatro elementos, tal y como se muestra en la Figura A.4 [Garfinkel, *et al*., 2005] [Want, 2006]:

- **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.
- **Lectores.** Transmiten continuamente pulsos de energía mediante ondas de radio, que son recibidas 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.
- **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.
- **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.

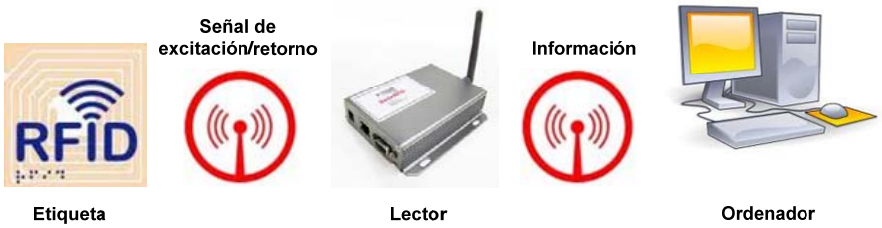


Figura A.4 Funcionamiento de RFID.

Las etiquetas RFID pueden clasificarse dependiendo de varios aspectos, por ejemplo por su forma o sus características físicas (adhesivas, tarjetas, discos, pulseras, implantes, 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 A.5, 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 implantes, en los que la estructura interna, mostrada en la Figura A.5, 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 *Food and Drug Administration* (FDA) 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 fabricación y es muy difícil de alterar, ya que tan solo el proceso 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.

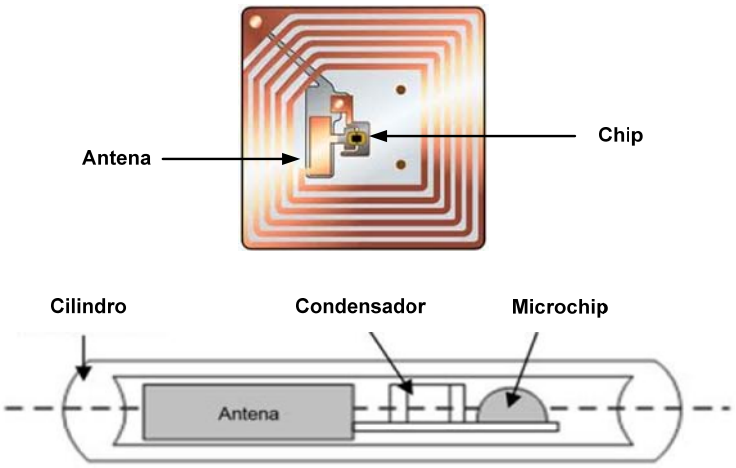


Figura A.5 Etiqueta RFID y *Transponder* implantable.

- **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 posible. Las etiquetas activas se utilizan principalmente cuando es necesario un mayor alcance de lectura y 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 *Federal Communications Comisión* (FCC); en Canadá, el *Department of Communication* (DOC); en Japón, el *Ministry of Internal Affairs and Communications* (SOUMU); en China, el *Ministry of Information Industry* (MII); y en Europa, los organismos *European Radiocommunications Office* (ERO), *Conférence Européenne des administrations des Postes et des Télécommunications* (CEPT), European Telecommunications Standards Institute (ETSI), 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 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 A.4. 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 A.4 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 *Global Positioning System* (GPS) que tienen poca precisión en entornos cerrados. 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.

A.4 Near Field Comunication (NFC)

NFC, es una tecnología de comunicación inalámbrica de corto alcance que permite el intercambio bidireccional de datos entre dispositivos a corta distancia, aproximadamente 10 cm. La idea de desarrollar esta tecnología surgió para crear un nuevo protocolo que preste compatibilidad con las tecnologías sin contacto de corto alcance ya existentes. Por esta razón NFC es una extensión simple del estándar ISO/IEC 14443 [Iso.org, 2010] de tarjetas de proximidad (tarjetas RFID sin contacto) que combina la interface de una tarjeta inteligente y de un lector dentro de un mismo dispositivo móvil.

Su desarrollo empieza en el año 2002 y sus promotores fueron Philips y Sony principalmente para conseguir compatibilidad con sus tecnologías, Mifare y FeliCa respectivamente, pero no fue hasta finales del año 2003 cuando se aprueba como el estándar ISO 18092 [Iso.org, 2010]. Un dispositivo NFC puede comunicarse con cualquier tarjeta inteligente o lector, que cumplan el estándar ISO/IEC 14443. Tan bien puede comunicarse con otros dispositivos NFC, y es por lo tanto compatible con la infraestructura sin contacto. NFC está destinado principalmente para el uso en teléfonos móviles ya que no está orientada para la transmisión masiva de datos como Wi-Fi por ejemplo.

NFC fue aprobado como un estándar ISO/IEC el 08 de diciembre del 2003 y posteriormente como un estándar ECMA. Al igual que la ISO/IEC 14443, se comunica vía inducción de campo magnético. En la Figura A.6 se muestran las dos maneras de comunicación posible, primero entre un móvil NFC y una etiqueta RFID (a) y después la comunicación entre dos móviles NFC (b). La comunicación NFC consta de cinco fases las cuales son importantes ya que tienen una función específica y siempre están presentes en el establecimiento de la comunicación. Estas etapas son:

- Descubrimiento: En esta fase los dispositivos inician la etapa de rastrearse el uno al otro y posteriormente su reconocimiento.
- Autenticación: En esta parte los dispositivos verifican si el otro dispositivo está autorizado o si deben establecer algún tipo de cifrado para la comunicación.
- Negociación: En esta parte del establecimiento, los dispositivos definen parámetros como la velocidad de transmisión, la identificación del dispositivo, el

tipo de aplicación, su tamaño, y si es el caso también definen la acción que va a ser solicitada.

- Transferencia: Una vez negociados los parámetros para la comunicación, se puede decir que ya está realizada exitosamente la comunicación y ya se puede realizar el intercambio de datos.
- Confirmación: El dispositivo receptor confirma el establecimiento de la comunicación y la transferencia de datos.

Cabe destacar que la tecnología NFC no está destinada para la transferencia masiva de datos, pero se puede utilizar para la configuración de otras tecnologías inalámbricas de mayor ancho de banda como Bluetooth o WiFi con la ventaja de que si se utiliza NFC, el tiempo de establecimiento de la comunicación es menor que si se utilizan otras tecnologías como WiFi para efectuar el enlace.

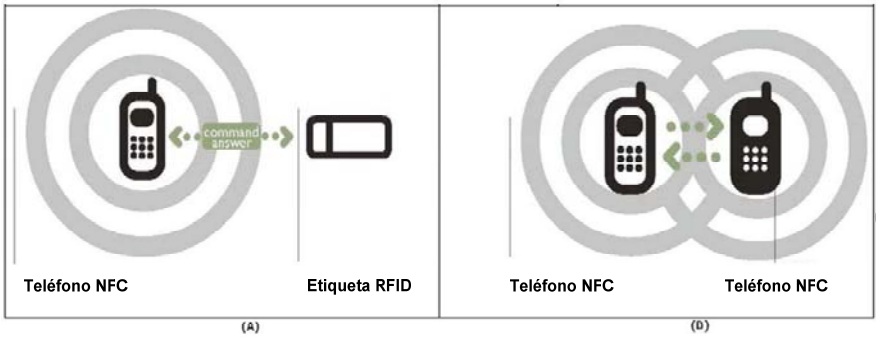


Figura A.6 Inducción del campo magnético de NFC [Nfc-forum.org, 2010].

Opera dentro de la banda *Industrial, Scientific and Medical* (ISM) de radio frecuencia de 13,56 MHz disponible globalmente sin restricción y sin necesidad de licencia para su uso, con un ancho de banda de casi 2 MHz. NFC es una tecnología estandarizada en la ISO/IEC 18092 y la ECMA-340. Estos estándares especifican los esquemas de modulación, codificación, velocidades de transferencia y formato de la trama de la interfaz RF de dispositivos NFC, así como los esquemas de inicialización y condiciones requeridas para el control de colisión de datos durante la inicialización para ambos modos de comunicación, activo y pasivo. También definen el protocolo de transporte, incluyendo los métodos de activación de protocolo y de intercambio de

datos. La interface de comunicación para NFC está estandarizada en: ISO/IEC 18092 / ECMA – 340: *Near Field Communication Interface and Protocol-1* (NFCIP-1) ISO/IEC 21481 / ECMA – 352: *Near Field Communication and Protocol-2* (NFCIP-2) [Iso.org, 2010].

NFC incorpora una variedad de estándares pre-existentes incluyendo ISO/IEC 14443 de ambos tipos, tipo A (normal) y tipo B (*banking/short range*), y FeliCa. Por lo tanto los teléfonos NFC muestran interoperabilidad básica con módulos que ya existentes como RFID, como se muestra en la parte (a) de la Figura A.6. La distancia de trabajo con antenas compactas estándar es aproximadamente 20 cm, aunque generalmente efectivo, es cercano a los 10 cm. Las velocidades de transmisión que soporta esta tecnología son de 106, 212, 424 u 848 kbits/s. La comunicación NFC es bidireccional, por lo tanto los dispositivos NFC son capaces de transmitir y recibir datos al mismo tiempo. De esta manera, los dispositivos pueden verificar el campo de Radio Frecuencia y detectar una colisión si la señal recibida no coincide con la señal transmitida.

A.4.1 Estándares de Comunicación NFC.

Dentro de los estándares de NFC se ha establecido un formato común de datos para que los dispositivos NFC puedan compartir información entre sí. Estos estándares señalan las especificaciones que permiten la comunicación y son propiedad del NFC Forum [Nfc-forum.org, 2010]. NFC Forum ha definido un formato de datos común llamado *NFC Data Exchange Format* (NDEF) [Nfc-forum.org, 2010], que puede ser usado para guardar y transportar diferentes tipos de elementos, que van desde cualquier objeto escrito *Multipurpose Internet Mail Extensions* (MIME) hasta documentos *Record Type Definition* (RTD), tales como *Uniform Resource Locator* (URL).

La especificación NDEF define un formato de encapsulación de mensaje para el intercambio de datos entre dispositivos NFC o de un dispositivo NFC a una etiqueta RFID, y las reglas para la construcción de un mensaje NDEF válido. Además también se puede crear una cadena ordenada de registros NDEF. La diferencia entre una etiqueta y un dispositivo NFC es que la primera no permite una interacción con el usuario y por sí sola no puede mostrar ninguna información al usuario, además es

pasiva es decir que no genera su propia energía de funcionamiento y necesita de un dispositivo activo para que funcione. En cambio un dispositivo NFC permite una interacción con el usuario y posee su propio generador de energía, para que a través de un campo de inducción pueda estimular y generar la energía para el funcionamiento de los elementos pasivos.

NDEF es un formato binario ligero que puede encapsular uno o más *payloads* de diferente tipo y tamaño dentro de la estructura de un solo mensaje. Un *payload* es la carga útil, es decir la información útil para el usuario del flujo de información transferido. El *payload* está identificado por un tipo, una longitud y un identificador opcional.

- **Longitud de la carga (*payload*):** Indica el número de octetos de *payload*, es decir, indica la longitud de *payload* encapsulada en un registro. Se encuentra dentro de los primeros 8 octetos de un registro. El Campo PAYLOAD_LENGTH es un octeto para registros pequeños y cuatro octetos para registros normales. Los registros pequeños están indicados estableciendo el bit de la bandera SR (Short Record) en 1.
- **Tipo de *Payload*:** Indica la clase de datos que está siendo transportado en el *payload* de ese registro. El tipo del primer registro, debe proveer el contexto de procesamiento no solo para el primer registro sino para todo el mensaje NDEF. Los tipos de identificadores pueden ser *Uniform Resource Identifier* (URI), MIME o tipos específicos NFC (*NFC-specific*). Al identificar el tipo de carga útil, es posible despachar la carga para la aplicación del usuario apropiada.
- **Identificador de *Payload*:** El *payload* puede dar un identificador opcional a través de una URI absoluta o relativa; esto permite a las cargas que soportan URI vincular tecnologías de referencia con otras cargas.

NDEF es simplemente un formato de mensaje, es decir que solo especifica la estructura del formato por lo que no se debe pensar que declara algún tipo de circuito o algún concepto de conexión o que pueda especificar el intercambio de información. El formato de datos de NDEF es el mismo tanto para un dispositivo NFC como para una etiqueta NFC/RFID, por lo que la información de NDEF es independiente de los

tipos de dispositivos que se estén comunicando. Dentro del formato de un mensaje NDEF se puede enviar un variado tipo de información como:

- Puede encapsular documentos *Extensible Markup Language* (XML), fragmentos XML, datos encriptados, e imágenes como archivos JPEG, GIF, etc.
- Encapsular cadenas de información.
- Agregar documentos múltiples y entidades que están asociadas lógicamente de alguna manera. Por ejemplo, se puede encapsular un mensaje *NFC-specific* y un conjunto de archivos adjuntos de tipos estandarizados que tienen referencia desde ese mensaje *NFC-specific*.
- Encapsulado compacto de pequeños payloads.

Los registros NDEF son de longitud variable pero todos tienen un formato común que se representa en la Figura A.7. La información de los registros NDEF se presenta en nivel de octetos. El orden de transmisión es de izquierda a derecha y de arriba hacia abajo; de esta manera el bit más significativo del octeto es el bit del extremo izquierdo y para una cadena de octetos es igual, el bit más significativo es el de la extrema izquierda de todo el campo de octetos y es el que se transmite primero. A continuación se detallan los campos que conforman el formato del registro NDEF:

- **Message Begin (MB):** Es una bandera de 1 bit que cuando se constituye indica el inicio de un mensaje NDEF.
- **Message End (ME):** Esta bandera es un campo de 1 bit que se establece en el caso de un payload fragmentado. Esta bandera solo se establece en el segmento de terminación de este payload fragmentado, indicando el final de un mensaje NDEF.
- **Chunk Flag (CF):** Es una bandera de 1 bit que si se establece indica, que es el primer segmento de registro o que es un segmento de registro del medio de un payload fragmentado.
- **Short Record (SR):** Está formado por 1 bit y al establecerse indica que el campo PAYLOAD_LENGTH es un solo octeto y no cuatro octetos como lo es para un registro NDEF normal. Este registro pequeño está destinado para una encapsulación compacta la cual permite que pequeños payloads sean parte de campos de payloads con un tamaño de entre 0 a 255 octetos. Un mismo

mensaje NDEF puede tener tanto registros NDEF normales como registros cortos.

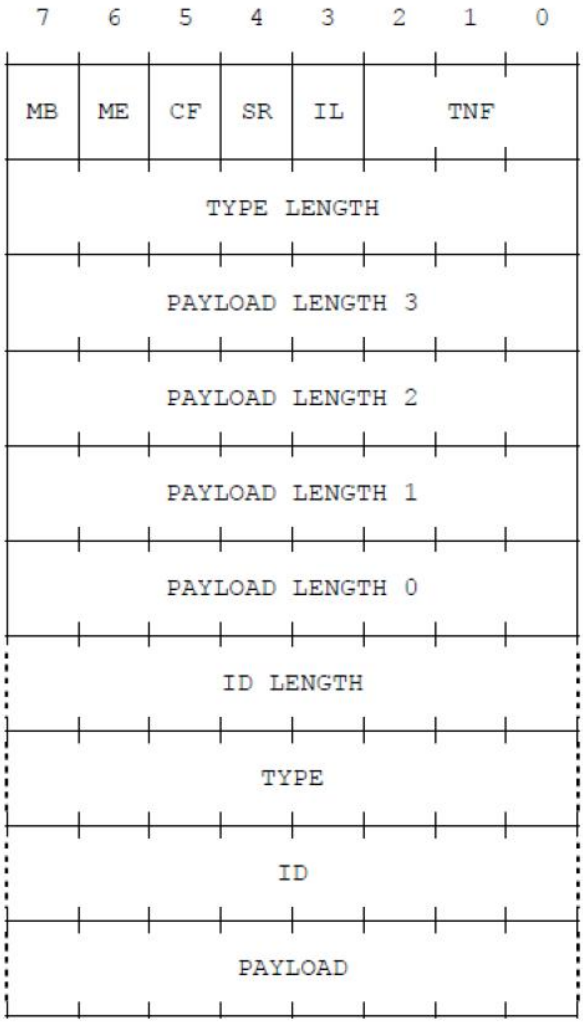


Figura A.7 Formato de un registro NDEF.

- **ID_LENGTH (IL):** La bandera IL es de 1 bit que si se establece indica que el campo ID_LENGTH está presente en la cabecera del registro como un octeto

pero si el campo IL es cero entonces éste es omitido de la cabecera y el campo ID también es omitido del registro.

- **TYPE NAME FORMAT (TNF):** Es un campo de 3 bits que indica la estructura del valor del campo TYPE. Estos valores se detallan a continuación y se muestran en la Figura A.8:

Type Name Format	Valor
Vacío	0x00
Tipo NFC Forum (NFC RTD)	0x01
Tipo de Medios	0x02
URI Absoluto	0x03
Tipo NFC Forum externo	0x04
Tipo Desconocido	0x05
Sin Cambio (Unchanged)	0x06
Reservado	0x07

Figura A.8 Valores de TNF.

EL valor 0x00 (vacío) significa que no hay ningún tipo o payload asociada al registro. De esta manera los campos TYPE_LENGTH, ID_LENGTH y PAYLOAD_LENGTH deben ser cero y por lo tanto los campos TYPE, ID y PAYLOAD respectivamente son omitidos del registro.

El valor 0x05 (*Unknown*) debe ser usado para indicar que el campo de payload es desconocido. Este valor de TNF ocasiona que el campo TYPE sea omitido del registro NDEF ya que el valor del campo TYPE_LENGTH debe ser cero. Se recomienda que cuando un analizador NDEF esté recibiendo un registro NDEF de este tipo, provea un mecanismo para guardar pero no para procesar el payload.

EL valor 0x06 (*Unchanged*) no debe usarse en otro registro que no sean los fragmentos de registro del medio y el fragmento del registro terminal que forman

payloads segmentados. Si se establece este valor TNF, el campo TYPE_LENGTH debe ser cero y el campo TYPE es omitido del registro NDEF. El valor 0x07 (Reservado) es para usos futuros y no debe ser usado.

- **TYPE_LENGTH:** Este campo es un entero no asignado de 8 bits que representa la longitud en octetos del campo TYPE. Al referirse a un entero no asignado, quiere decir que no es una constante sino que su valor depende de la longitud del campo TYPE.
- **ID_LENGTH:** Este campo también es un entero no asignado de 8 bits que especifica la longitud del campo ID en octetos y está presente sólo si la bandera IL en la cabecera del registro se establece en 1.
- **PAYLOAD_LENGTH:** Es un entero no asignado que representa la longitud en octetos del campo PAYLOAD y a su vez el tamaño del campo PAYLOAD_LENGTH depende del valor de la bandera SR. Si la bandera SR está establecida, el campo PAYLOAD_LENGTH representa un solo octeto; pero si esta bandera está vacía, el campo PAYLOAD_LENGTH es de 4 octetos representando un entero no asignado de 32 bits.
- **TYPE:** Este campo es un identificador que especifica el tipo de payload de la información transmitida. El valor de este campo debe seguir la codificación, la estructura y el formato implícito por el valor del campo TNF. El tamaño máximo de este campo es 255 octetos.
- **ID:** El valor de este campo es un identificador que tiene la forma de una referencia URI. Para NDEF, una URI es simplemente una cadena de texto que identifica un nombre, una localización o alguna característica de un determinado recurso. Los fragmentos finales y de la mitad de un payload segmentado no debe tener el campo ID ya que se trata del mismo campo de datos pero en diferentes fragmentos por lo que solamente basta con definir una vez la información completa acerca de todo el payload. Todos los demás tipos de registros pueden tener este campo ID. El tamaño máximo de este campo es 255 octetos.
- **PAYLOAD:** Dentro de este campo está la carga o información útil para las aplicaciones del usuario y la estructura interna de los datos llevados en este campo es oculta para NDEF. El tamaño máximo del campo PAYLOAD es 2^{32} -1

octetos para un diseño de registro NDEF normal y 255 octetos para un registro pequeño. Pero para tamaños de payload mayores a 2^{32} -1 se segmenta dicho payload para poder ser transmitida en fragmentos (*Payloads* fragmentados).

Un mensaje NDEF está compuesto por uno o varios Registros NDEF. El primer registro de un mensaje está marcado con la bandera *Message Begin* (MB) y el último registro lleva la bandera *Message End* (ME). Si un mensaje está compuesto por un solo registro, éste mismo lleva tanto la bandera MB como la bandera ME. El número de registros que un mensaje NDEF puede llevar es ilimitado. Los mensajes NDEF no deben superponerse, es decir, que las banderas MB y ME no deben ser utilizadas para anidar mensajes NDEF. Los mensajes NDEF pueden ser anidados llevando un mensaje completo como un payload en un registro NDEF.

Figura A.9 Mensaje NDEF con varios registros.

En realidad los mensajes NDEF no llevan números índices para indicar su orden, sino que está implícito en el orden en que los registros son serializados. Por ejemplo si los registros son empaquetados por una aplicación intermedia, ésta es la responsable de asegurar que el orden de los registros sea el mismo. Un registro es la unidad para llevar un *payload* en un mensaje NDEF y cada payload es descrito por sus propios parámetros.

Un registro (*record chunk*) lleva solo una parte de *payload*. Estas cargas en partes son utilizadas para particionar un contenido generado dinámicamente o mensajes demasiado largos en múltiples partes de registro ordenados en un mismo mensaje NDEF. Este agrupamiento sirve para reducir la necesidad de almacenamiento en el búffer de salida. Un mensaje NDEF puede contener cero o más *payloads* fragmentados. Cada *payload* fragmentado es codificado con las siguientes reglas:

- La parte de registro *inicial* tiene la bandera Chunk Flag (CF). El tipo de todo el *payload* fragmentado debe estar indicado en el campo de tipo. El campo ID puede ser usado para llevar un identificador de todo el *payload* fragmentado. El

campo de la longitud de *payload* indica solo el tamaño de los datos que lleva el registro actual y no el tamaño de todo el *payload*.

- El fragmento de registro *medio* tiene la bandera CF indicando que este registro contiene la próxima parte de datos del mismo tipo y con el mismo identificador como el fragmento de registro inicial. El valor del campo TYPE_LENGTH y del campo ID_LENGTH debe ser cero y el campo TNF debe ser 0x06.
- La parte de registro *final* es un registro NDEF con la bandera CF limpia indicando que este fragmento de registro es el que contiene el último fragmento de datos del mismo tipo y con el mismo identificador que el fragmento del registro *inicial.* El valor del campo TYPE_LENGTH y del campo ID_LENGTH debe ser cero y el campo TNF debe ser 0x06.

Un *payload* fragmentado debe ser encapsulado en un mensaje NDEF, o sea que no debe abarcar múltiples mensajes NDEF. Por lo tanto, ni un fragmento de registro inicial o medio puede tener una bandera ME establecida.

A.4.2 Definición de tipo de registro RTD.

La especificación RTD, provee las pautas para la especificación de los tipos de registros NFC bien conocidos (*well-known record types*) que puedan ser incluidos en mensajes NDEF transmitidos entre dispositivos NFC o entre un dispositivo NFC y una etiqueta NFC/RFID. Es decir que esta especificación permite soportar aplicaciones específicas NFC. En el campo de formato de los tipos de registros de un registro NDEF están contenidos los nombres del tipo de registro llamado *record type name*. Cada definición de tipo de registro es identificado por su *Record Type Name*. Los *Record Type Name* pueden ser especificados en distintos formatos llamados TNF como ya se vio anteriormente. El tipo bien conocido NFC (*NFC well-known*) es un formato diseñado para las etiquetas NFC y también para crear formatos primitivos, es decir un formato común, por ejemplo, se puede usar este tipo cuando no hay un equivalente URI o no hay disponible un tipo MIME o también cuando las limitaciones de un mensaje requieren un nombre muy pequeño.

Cuando un mensaje NDEF lleva un tipo *NFC well-known*, el campo TNF tiene el valor 0x01. Estos tipos de formatos en la especificación RTD son definidos como

Uniform Resource Name (URN) y tienen un identificador de nombre, "nfc", llamado *Namespace IDentifier* (NID). La estructura de un tipo de registro well-known viene de la siguiente manera:

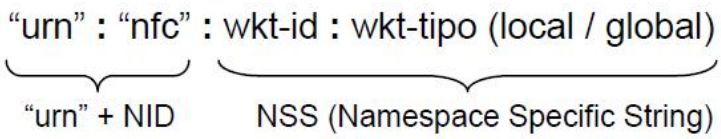


Figura A.10 Estructura de tipo de registro well-known.

Un ejemplo de cómo es un URN de tipo NFC well-known es "urn:nfc:wkt:a" en el cual:

- **urn** identifica a este tipo como un Nombre de Recurso Uniforme.
- **nfc** es el identificador de nombre NID.
- **wkt** es el wkt-id, un prefijo necesario antes del formato de tipo que lo identifica como un tipo well-known.

Pero cuando se codifica dentro de un mensaje NDEF, tanto el NID "nfc" como el prefijo "wkt" son omitidos. Dos ejemplos de tipos *well-known* son considerados iguales si y solo si sus representaciones binarias son equivalentes, es decir que se compara de carácter en carácter. Existen dos clases de este tipo *NFC well-known*:

- **Tipo *NFC Forum Global*:** Un tipo *NFC Global* comienza con una letra mayúscula, por ejemplo, "A", "Hello_world", etc.
- **Tipo *NFC Forum Local*:** La finalidad de los tipos *NFC Forum Locales* es para ser usados en el contexto de otro registro. Se utilizan cuando no hay la necesidad de definir el significado del formato fuera de un contexto Local.

La Definición de Tipo de Registro de Texto (*Texto RTD*) define un registro *NFC well-known* que contiene solamente datos de texto y que se usa sin que se requiera de mucho espacio. El nombre que se utiliza para representar este tipo de registro es "T".

La Definición de Tipo de Registro *URI* no permite acceder a recursos de internet. Es básicamente un contenedor compacto de URI's. El nombre de este registro para poder identificarlo es "U".

La Definición de Tipo de Registro *Smart Poster* especifica la manera de incorporar datos como *Short Message Service* (SMS), URI's o números de teléfono en etiquetas NFC o la manera de moverlos entre dispositivos. El objetivo de los *Smart Poster* es proveer una manera simple para acceder a un servicio remoto con un simple acercamiento de un dispositivo NFC. Un *Smart Poster* también puede contener acciones que desplieguen una aplicación en un dipositivo, por ejemplo iniciar un explorador de internet. El nombre que identifica el registro *Smart Poster* es "Sp". Dentro del *payload* de un *Smart Poster* hay algunos tipos de registros y puede haber uno o más de estos registros:

- **Registro Título:** Este registro es un ejemplo de registro Texto RTD. Nos permite dar información del contenido del *Smart Poster* y puede ser visto por el usuario. No hay un límite para el número de estos registros en un *Smart Poster*, pero dos o más registros de este tipo no pueden utilizar el mismo identificador de lenguaje. Este es un registro opcional.
- **Registro URI:** Este registro es el núcleo del *Smart Poster* y los demás registros son datos acerca de éste. Debe haber uno y solamente un registro URI en un *Smart Poster*.
- **Registro Icono:** Este registro permite colocar una imagen para ser mostrada al usuario. Un *Smart Poster* puede contener algunos registros de este tipo con sus imágenes, pero solo una de estas debe ser mostrada. Los tipos de imágenes compatibles son PNG o JPEG y también puede haber registros con íconos animados o videos de tipo MPEG. Este también es un registro opcional.
- **Registro Acción:** Es un registro de tipo local y sugiere el curso de una acción que un dispositivo puede seguir para realizar un proceso y como ser tratado el contenido. El nombre del tipo local es "act". Este registro es opcional. El contenido de este registro es un byte y cada acción tiene un valor como muestra la Tabla A.5:
- **Registro Tamaño:** Indica el tamaño del objeto al cual hace referencia el registro URI, pero solo debe ser utilizado para propósitos de información y como una guía. Está formado por 4 bytes de enteros no asignados (32 bytes). El nombre que representa este registro de tipo local es "s". Es un registro opcional.

Tabla A.5 Valores de las distintas acciones del Registro Acción.

Valor	Acción
0	Hace la acción predeterminada (iniciar un explorador, enviar un sms)
1	Guardar información (guarda un sms en la bandeja de entrada, un número de teléfono en contactos, etc)
2	Abre la información para ser editada.
3…FF	Valores reservados para el futuro.

- **Registro Tipo:** Es un registro opcional y puede ser usado para que el dispositivo sepa qué clase de objeto puede esperar antes de abrir una conexión y esto puede ser útil por ejemplo cuando un *Smart Poster* tiene un archivo multimedia que no es compatible o no es reconocido por un lector y éste último no tiene la posibilidad de reproducir al archivo. El nombre para este registro Local es "t".

B

Metodología SysML

System Modeling Language (SysML) es un robusto lenguaje de modelado para la ingeniería de sistemas basado en *Unified Modeling Language* (UML) [SysML.org, 2010]. Este lenguaje resuelve gran parte de los problemas derivados del diseño y desarrollo de sistemas multiagente, entre ellos la dependencia de las herramientas orientadas a objetos [Bauer, et al., 2000] [AUML.org, 2010]. 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 que 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.

En la fase de diseño SysML se tienen que definir los estados internos, los roles que desempeña y los servicios que ofrece cada agente dentro del sistema multiagente, representándolos a través de diagramas SysML. Además, 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. Los Diagramas de Secuencia SysML presentan gran nivel de detalle y agregan el 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:

- **Diagramas de Definición de Bloques.** Se utilizan para representar los agentes y sus roles dentro del sistema. SysML extiende los diagramas de clase UML, haciendo una diferencia entre agente y objeto, como se muestra en la Figura B.1, logrando representar los agentes junto con sus características y arquitectura [SysML.org, 2010] [AUML.org, 2010]. 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:
    - **Identificador.** Es el nombre o clave única que se le da a cada agente dentro del sistema.

- **Rol.** Un agente puede tener varios roles que definen su comportamiento dentro del sistema, siendo capaz de cambiar de rol en tiempo de ejecución.
- **Organización.** Un agente puede pertenecer a una o varias organizaciones, las cuales, definen los roles de los agentes y las relaciones entre esos roles.

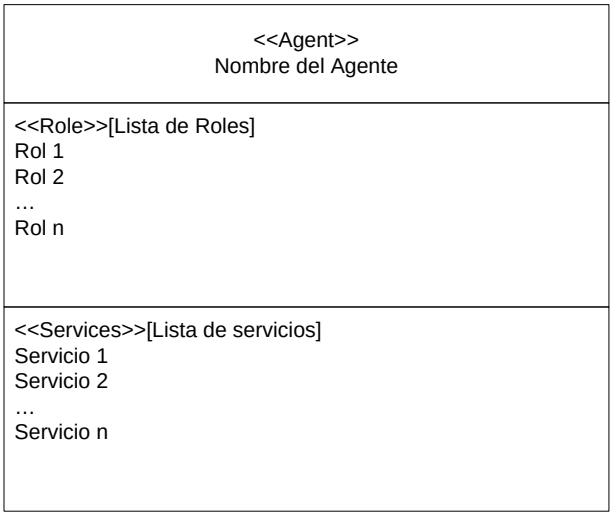


Figura B.1 Diagrama de Definición de Bloques para un agente en SysML.

A esta estructura básica se le añaden clases, entre ellas, las capacidades, los servicios y los protocolos de interacción. 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.

- **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 *Agent Communication Language* (ACL), 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 B.2 muestra la estructura general de un diagrama de secuencia SysML.

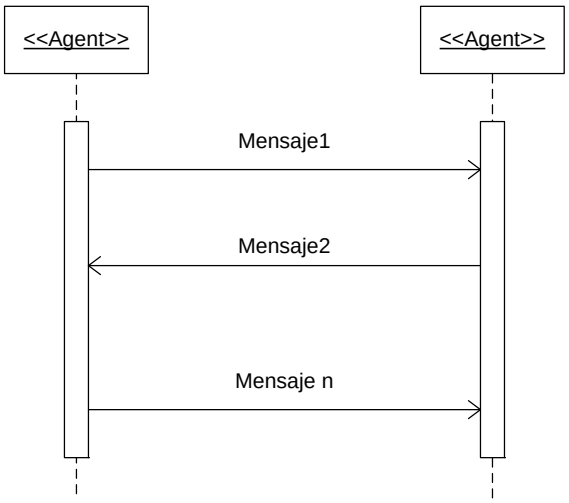


Figura B.2 Diagrama de secuencia en SysML.

- **Diagramas de Estados.** Tal y como se muestra en la Figura B.3, 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.

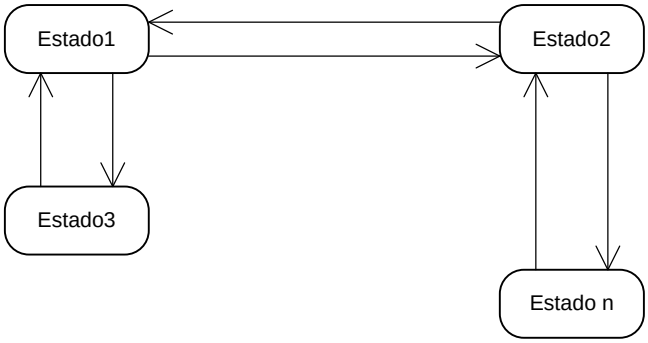


Figura B.3 Diagrama de estados en SysML.

- **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 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 B.4 muestra la estructura general de un diagrama de interacción en SysML.

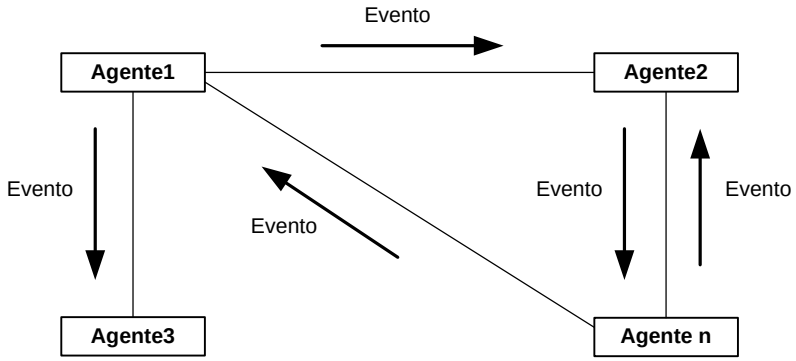


Figura B.4 Diagrama de interacción en SysML.