Universidad Carlos III de Madrid
Escuela Politécnica Superior

Departamento de Informática
Doctorado en Ciencia y Tecnología Informática

Tesis Doctoral

Sistema Multi-Agente basado en Contexto, Localización y Reputación para dominios de Inteligencia Ambiental

Verónica M. Venturini

Dirigida por

José Manuel Molina

Javier Carbó

Colmenarejo, 2012

Web page: http://www.giaa.inf.uc3m.es/miembros/reilly

E-mail: veronica.venturini@uc3m.es

Telephone: +34 91 856 1318

Address:

Grupo de Inteligencia Artificial Aplicada
Departamento de Informática
Universidad Carlos III de Madrid
Av. de la Universidad Carlos III, 22
Colmenarejo 28270 — Spain

Sistema Multi-Agente basado en Contexto, Localización y Reputación para dominios de Inteligencia Ambiental

Autor: Verónica M. Venturini
Directores: José Manuel Molina
Javier Carbó

Firma del Tribunal Calificador:

	Nombre y Apellidos	Firma
Presidente: D.	.....	.....
Vocal: D.	.....	.....
Vocal: D.	.....	.....
Vocal: D.	.....	.....
Secretario: D.	.....	.....

Calificación: .....

Colmenarejo, ... de ..... de 2012.

A mi hno. Matías y mis Lalitos que ya partieron.
A Marcelo y Diego, a mis padres.
A Bruno, por su paciencia y compañía.

Índice general

Índice de figuras

Índice de cuadros

Glosario

IA	Inteligencia Artificial
AmI	Ambient Intelligence - Inteligencia Ambiental
SMA	Sistema Multi-Agente
AG	Algoritmo Genético
BD	Base de Datos
RFID	Radio Frecuency Identification - Identificación por Radio Frecuencia
BDI	Beleife, Desires and Intentions - Creencias, Deseos e Intenciones
UA	User Agent - Agente Usuario
UP	Provider Agent - Agente Proveedor
LA	Locator Agent - Agente Localizador
BA	Broker Agent - Agente Intermediario
UMA	User Manager Agent - Agente Administrador de Usuarios
CALoR	Context-Aware and Location Reputation Model Modelo de Reputación basado en Contexto y Localización
ACL	Agent Communication Languaje - Lenguaje de Comunicación de Agentes

Abstract

RESEARCHES on Ambient Intelligent (also known as Ubiquitous Computing) using wireless technologies have increased over the last few years. *Ambient Intelligence* is a new concept adopted for referencing an intelligent environment, where the user is assist by the computer and his mobile device wherever he is. Several technologies are necessaries in these environments to provide personalized services. All immersed devices are invisibles to the user.

Ambient Intelligence is an ideal workspace for agents because of autonomous distributed and proactive agent nature. The main contribution of this Thesis consists of designing a methodological approach a Context Aware System (involving location services) using Agents that can be used in very different domains.

We review several scenarios to define a multi-agent architecture, that support the information needs of these new technologies, that will be used in heterogeneous domain. We use ontology to facilitate the communication between agents. Furthermore, this ontology allows agent's reasoning. In this manner, the use of geographic information allows to infer high level information about the real place where the agent is.

In Ambient Intelligence scenarios, user preferences allow to select the best services for each user. This personalization is particularly suitable in a multi-agent system with learning capabilities. Based on this, and on the improvement of the mobile technologies, we developed an algorithm that allows learning the user profile. A list of prioritized services would therefore be presented to the user that takes into account user interests. Also, the providers agents could make their offers.

Finally, users-providers interactions that take place in a dynamical domain needs a certain trust. Trust is build using trust and reputation systems. In this manner, we present CALoR reputation model that is a reputation model based-on context and location. This model allows the user agents of the systems to make recommendations from each other. CALoR take into account the past interactions of the agents and their spatial-temporal issues. We assume that a recommendation send by a user agent over a provider agent is more reliable when the agents interact closely and recently.

In summary, the reader will found in next pages a multi-agent system based-on context and location for heterogeneous domain in ambient intelligence. We explain the detail of the user profile acquisition using genetic algorithms. And we present CALoR system to consolidate the relationship between agents. For each item, we expose our experiments and the obtained results, that validates this approach.

Resumen

Las investigaciones en Inteligencia Ambiental (denominada también Computación Ubicua) utilizando tecnologías inalámbricas han crecido en los últimos años a pasos agigantados. El término *Inteligencia Ambiental* es un nuevo concepto adoptado para hacer referencia a entornos inteligentes, en donde la computadora y los dispositivos móviles asisten al usuario en sus actividades cotidianas. Son entornos con un gran despliegue de diferentes tecnologías, invisibles al usuario, que hacen posible la provisión de servicios personalizados.

Los agentes inteligentes son un paradigma especial dentro de la Inteligencia Ambiental. La principal contribución de este trabajo consiste en el diseño metodológico de un sistema basado en contexto (incluyendo servicios de localización) utilizando agentes inteligentes, que da soporte a las necesidades de información de estas nuevas tecnologías, para dominios heterogéneos. Los agentes facilitan la comunicación y las interacciones entre los usuarios del sistema. Para hacer factible este intercambio de mensajes, es necesario contar con una ontología, la cual posibilita, no tan solo la comunicación entre los agentes intervinientes, sino que además, permite a los agentes razonar sobre el contexto en el que se sitúan. En este sentido, el uso de información geográfica permite inferir información de alto nivel sobre dónde se encuentra un agente.

Las preferencias de los usuarios permiten seleccionar los mejores servicios en cada escenario de Inteligencia Ambiental. Esta personalización es particularmente adecuada en los sistemas multi-agentes con capacidades de aprendizaje. En base a esta concepción, se aborda la adquisición de perfiles de usuarios, de forma no intrusiva, a través de técnicas de computación evolutiva. La adquisición de las preferencias del usuario hacen posible que se le presente al usuario una lista de actividades priorizada por realizar, según el dominio en el que se encuentre a efectos de maximizar factores que le sean relevantes. Además, en base a estos perfiles, los agentes proveedores realizan sus ofertas.

Finalmente, las interacciones usuarios-proveedores en un dominio dinámico, requieren de cierto grado de confianza, el cual se adquiere a través del uso de sistemas de confianza y reputación. En este sentido, se presenta CALoR, un modelo de reputación basado en contexto y localización, que permite a los agentes usuarios del sistema realizar recomendaciones a otros agentes usuarios, considerando sus experiencias pasadas, y también aspectos espacio-temporales. Se asume que la recomendación enviada por un agente usuario sobre un agente proveedor es más fiable cuando han interactuado recientemente, y el agente usuario ha podido evaluar el servicio provisto a una distancia considerablemente corta.

Resumiendo, en las páginas siguientes, se presentará al lector, una arquitectura multi-agente basada en contexto y localización para dominios heterogéneos de inteligencia ambiental. Se explicará en detalle la construcción del perfil de usuario utilizando algoritmos genéticos para la provisión de servicios personalizados, y se presentará el modelo de reputación CALoR para afianzar las relaciones entre los agentes de la arquitectura. Por cada módulo, se presentan los

experimentos realizados y los resultados obtenidos, que demuestran la robustez de la propuesta global.

Agradecimientos

Es justamente la posibilidad de realizar un sueño, lo que hace que la vida sea interesante.
Paulo Coelho

En primer lugar, quiero agradecerles a mis padres, Vicky y Carlos, por no imponerme que camino seguir, y por lo contrario, regalarme la libertad para que pueda elegir quién y qué ser en la vida. A mis hermanos, Marce y Diegui, por darme las fuerzas para seguir siempre adelante y ser incondicionales.

Un agradecimiento muy especial a mis directores, José Manuel y Javier, por su infinita paciencia a lo largo de todos estos años, por haberme empujado a continuar con este trabajo, por el tiempo que me brindaron y por haber comprendido que ha pesar de la decisión de regresar, el doctorado era mi meta.

A mis compañeros del GIAA. En especial a Luis, Nayat, Rodri, Ani, Oscar y Lauri por su amistad. Cómo olvidar los momentos compartidos! A Vero por haber abierto las puertas de su hogar en Colmenarejo y haberme hecho sentir como en casa.

A mi mayor fuente de inspiración, Alicia Pérez, quien cuándo lea estas líneas se enterará de mi admiración. Siempre recuerdo el día que entró a mi primer clase de Inteligencia Artificial, y me abrió la mente a este fascinante mundo de la IA. Gracias a Alicia y a Beatriz por su ayuda para poder realizar el viaje hacia la madre patria.

Y si de ayuda se trata, agradecer a mis tíos Ana y Augusto que me facilitaron todo lo que hizo falta para poder emprender este camino, y a mis tías de Salta.

A mi compañero de vida, Bruno, que sin saber lo que el destino nos esperaba, un día me dijo "y si te vas a especializar afuera?". Y ahora está acompañándome en este final. Gracias por sus correcciones en inglés, por haberme enseñado a escuchar a los "gráficos", por su infinito apoyo y paciencia.

A mis amigos de Salta (Argentina) por estar siempre y brindarme su cariño. En particular Keto y Coco que me hicieron el aguante durante aquel año que decidí partir, quedando con todo el trabajo de GRIVA a cuestas. A mis amigos del Grupo de Control del centro atómico Bariloche (Argentina): Tincho, que siempre estaba para sacarme alguna duda de Java; Ale y Norbert, por escucharme y discutir conmigo algunas cuestiones de las que surgió CALoR (pese al frío patagónico); Gonza por transmitirme sus conocimientos de Latex. Gracias por escuchar mis delirios con los "agentes".

A todos, familia y amigos, gracias por ayudarme en cada instante que lo necesito. Los llevo siempre en mis pensamientos, a pesar de la distancia.

Vero
Colmenarejo, Septiembre del 2012.

I

Contexto de la Problemática

1

Introducción

1.1 Motivación

La producción de PCs en masa, la infraestructura de internet y la telefonía celular, impulsaron un nuevo mundo: un mundo inteligente. Es aquel mundo en el que los objetos que se encuentran a nuestro alrededor, se convierten en microsistemas inteligentes que interactúan con las personas y con otros microsistemas, a través de sensores y actuadores. Podríamos decir que nos encontramos en una etapa de transición hacia la gran revolución tecnológica. En el mundo de la telefonía móvil, en particular, se han vivido sucesivas revoluciones. En cada una de ellas se fueron incorporando nuevos servicios al usuario (radio, cámara de fotos, agenda), llegando a la gran revolución del 3G, donde el usuario puede manipular contenido multimedia, conectarse a internet con un buen ancho de banda, etc. Seguido a esto, surgió el fenómeno de los teléfonos inteligentes, que aunque aún no están en su gran auge puesto que no se puede explotar sus características sobresalientes, cada vez más personas en el mundo lo tienen. El tema actual es incorporar mayores servicios y más facilidades para asistir a sus usuarios. Desde la perspectiva de la Inteligencia Artificial, esta asistencia debería ser totalmente transparente, evitando la interacción constante usuario-dispositivo.

Los autores (Vázquez and López, 2003) hablan de la era del 4G. En este mundo del 4G, el teléfono móvil se transforma en una terminal capaz de recibir televisión, acceso a internet, bajar música, leer tarjetas RFIDs, sacar fotos, habilitar video-telefonía y mucho más. Es un mundo inteligente en donde la ropa que uno utiliza, las pinturas de la pared, las alfombras de su piso y en dinero de su billetera tiene capacidades computacionales de comunicarse. En los escenarios sugeridos por la Information Society Technologies Advisory Group (ISTAG 2010) en (Ducatel et al., 2001) se extiende, además, la tendencia de que los dispositivos móviles se conviertan en simples chips colocados en la muñequera de una persona o implantados en su cuerpo. Es la etapa más prometedora, en donde el terminal conoce al usuario (es su asistente personal) y facilita sus tareas de acuerdo a ese conocimiento.

Pero, ¿qué se necesita para que un entorno sea "verdaderamente" inteligente? Un requerimiento clave es que, para poder conocer el estado actual de las personas, deberían distribuirse, en todo el entorno, dispositivos transparentes con las siguientes propiedades:

1. Personalizados: de manera tal que sus funciones se adapten a las necesidades específicas del usuario.

2. Adaptativos: permitir el aprendizaje y el reconocimiento de aciertos y errores.

3. Anticipados: conocer los deseos del usuario previamente, sin necesidad de que haya explícitamente introducido datos para dicho fin.

En la actualidad, para referirse a estos nuevos entornos inteligentes, se utilizan distintos conceptos alrededor del mundo:

1. En Europa: Inteligencia Ambiental (AmI) por la Comisión Europea.

2. En América: Computación Ubicua u Omnipresente.

3. En Japón: se utiliza la frase Redes Ubicuas para referirse a la inteligencia integrada al entorno.

A lo largo del presente documento adoptaremos el término AmI. La AmI fue caracterizada en (Gaggioli, 2005) como aquel entorno físico que es sensible y que responde a la presencia de personas, haciendo hincapié en dos características claves:

1. Inteligencia: el entorno digital permite el análisis de contexto, se adapta a sí mismo para el usuario o para los objetos que residen en él, aprende de su comportamiento y eventualmente, reconoce expresiones emocionales.

2. Embebido: significa que el dispositivo, con poder de cómputo, puede estar integrado en segundo plano en las actividades de la gente, mientras que su funcionamiento e interacción social pasan a primer plano.

Por su parte, Van Houten define la AmI como un mundo en el cual la "inteligencia" se encuentra embebida de manera virtual en todo lo que nos rodea (van Houten, 2005). Más recientemente, Wright (Wright et al., 2008), en su libro Safeguards in a World of AmI, habla de la AmI como "el internet de las cosas" o el mundo de las "partículas inteligentes".

Ahora bien, es necesaria una gran infraestructura tecnológica que de soporte a las interacciones de diferentes sistemas (Fig. 1.1). Su arquitectura dependerá del dominio y de las necesidades de aplicación, pero normalmente incluirá: dispositivos de redes, proxy, dispositivos móviles (PDA, teléfonos inteligentes), sensores y actuadores (cámaras, escáner, micrófonos, lectores de huella digital, detectores de presencia), repositorio de comportamientos, servidores de aplicaciones, dispositivos de localización, interfaces de percepción (reconocimiento de voz, reconocimiento de gestos, reconocimiento de la mirada), entre otras tecnologías implicadas.

En este nuevo paradigma móvil, cambiamos la perspectiva y en vez de enfocarnos en el usuario estacionario, que se encuentra fijo en un sitio y conectado todo el tiempo, hacemos foco en el usuario móvil, un usuario que ocasionalmente tiene conexión y que, de hecho, tiene diferentes necesidades de información. Esta información se encuentra estrechamente ligada a su ubicación geográfica. Es en este punto en donde surge la necesidad de otro tipo de tecnologías, emergiendo entonces los Sistemas de Localización y los Servicios Basados en Localización (Bernardos, 2008).

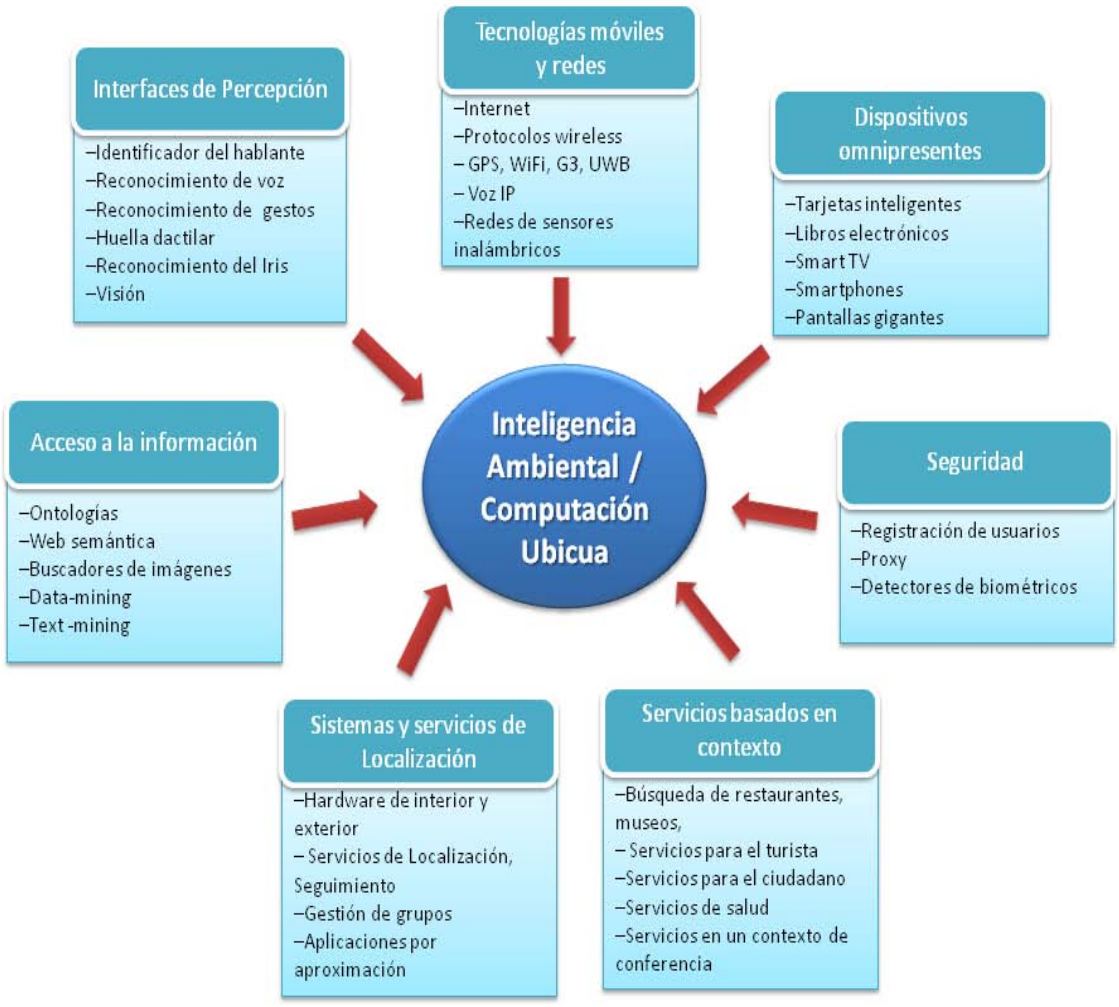

Figura 1.1: Tecnologías implicadas en AmI.

En cuanto a los sistemas de localización se refiere, existen distintos sistemas de tipo ultrasónicos, de radiofrecuencia y ópticos, entre otros. Los mas nombrados son GPS, RFID, GSM, WiFi, Bluetooth, Wimax, WAP, Aruba, UWB. Existen varios sistemas que implementan servicios móviles haciendo uso de dichas tecnologías, por ejemplo: Loop, Presence, Nokia Sport Traker, los cuales muestran el estado del usuario en los móviles de sus contactos. Por otro lado, existen múltiples investigaciones con respecto a los servicios que se pueden ofrecer una vez conocida la posición del usuario. Ejemplos de estos últimos son servicios de:

- previsión meteorológica,
- de seguimiento: de un vehículo, mascota, o niño,
- de navegación: recorrido de un punto a otro en vehículo,
- de localización: sirven para localizarse a uno mismo, a otros individuos u objetos. Son los que más abundan. Ejemplos:

  - *Teens Arrive Alive* basado en A-GPS y provisto por Nextel. Usa redes celulares.
  - El operador NTTDoCoMo que proporciona en Yokohama un servicio de localización para niños basado en WiFi.

- suministro de contenidos basados en localización: encontrar restaurantes, museos o cualquier otro tipo de punto de interés en el mapa del dispositivo móvil,
- redes sociales: los servicios de red social han evolucionado a la movilidad. Permiten concretar reuniones y citas basadas en proximidad, intereses y disponibilidad.
- aplicaciones de proximidad: las aplicaciones habilitadas por las comunicaciones de campo cercano. Por ejemplo NTTDoCoMo proporciona servicios de información comercial o cupones de descuento que se activan al acercar el teléfono a un lector NFC (servicio ToruCa).

Además de estos servicios basados en localización, existen los servicios basados en las características del entorno. A estos sistemas que proveen servicios de forma personalizada se los denomina sistemas basados en contexto, o en inglés context-aware systems.

Resumiendo, con el crecimiento del uso de los teléfonos inteligentes, y los servicios asociados, nos enfrentamos a usuarios más exigentes con nuevas necesidades de información relacionadas a sus preferencias y ubicación, apuntando a ese "mundo inteligente" tan deseado. Los sistemas actuales son complejos de desarrollar y adaptar a diferentes contextos.

1.2 Objetivos

Dado el análisis anterior y de cara a necesidades futuras para organizar o gestionar todo este tipo de información, se infiere la necesidad de contar con un sistema que cumpla las siguientes características:

- abierto: en el sentido de que permita la conexión hacia y desde otras plataformas (por ejemplo, un sistema de localización);
- adaptativo: como capacidad de realizar tareas y objetivos en distintos dominios en el que exista un conjunto de usuarios que requieren servicios y un conjunto de proveedores que los ofrezcan, adaptándose automáticamente a condiciones o requerimientos cambiantes; y
- dinámico: el sistema debe adquirir conocimiento del dominio, y ese conocimiento que evoluciona en el tiempo debe ser incorporado en el sistema, para luego adaptarse a la situación.

Este sistema además deberá dar soporte a la provisión inteligente de servicios de manera personalizada, basándose en la ubicación geográfica del usuario. Además, será el resultado de un conjunto de herramientas provistas para aquellos usuarios que interactuarán en un escenario inteligente utilizando algún dispositivo móvil que los mantenga conectados.

Delineadas las motivaciones del trabajo y luego de un exhaustivo estudio del estado del arte sobre las tecnologías implicadas (resumido en el Capitulo 2), a continuación se enumeran los objetivos de este trabajo de investigación:

- Objetivo 1:

Contar con una metodología ágil, clara y eficiente para el desarrollo de sistemas para AmI, que permitan el uso de agentes inteligentes, a través de la cual se obtenga una arquitectura multi-agente distribuida adaptable a cualquier dominio. La motivación del uso de agentes viene dada por su naturaleza autónoma (operan sin intervención directa del hombre y llevan un control de sus acciones y estados internos), social (interactúan con otros agentes y humanos) y pro-activa (son capaces de exhibir una conducta dirigida hacia objetivos al tomar la iniciativa). Asimismo, los agentes tienen la capacidad de razonar y de adaptarse a entornos dinámicos. La arquitectura que se construya deberá dar soporte a distinto tipos de interacciones entre los usuarios: localización, negociación, recomendaciones, coaliciones, gestión de perfil de usuario y provisión de servicios personalizados.

- Objetivo 2:

La construcción de una ontología genérica (a alto nivel) para dar soporte a las necesidades de información del sistema de agentes, y que pueda ser utilizada en cada dominio de inteligencia ambiental que requiera su uso. Pues las entidades del sistema deben compartir una representación común de la información conceptual. Por otro lado, una ontología permite el razonamiento sobre el contexto, haciendo posible la composición de información conceptual compleja.

- Objetivo 3:

La adquisición dinámica de perfiles de usuario, sin necesidad de que estos divulguen sus preferencias, sino infiriendo sus gustos a través de sus comportamientos. Es decir, la adquisición de las preferencias de forma implícita y no intrusiva. La motivación de este objetivo viene plasmada en la premisa de que el usuario es reticente a dar información privada. Para ello se construirá un algoritmo genético capaz de evolucionar las preferencias de los usuarios sobre cada característica de las tareas que puede realizar en el dominio en el que se encuentre, y según el valor de preferencia total, asignarle un orden de ejecución a las tareas.

- Objetivo 4:

La construcción de un sistema de reputación para facilitar la interacción entre los individuos del ambiente inteligente, aumentando la confianza en sus relaciones. Para ello se propone un modelo matemático que incluya en el cálculo de la reputación información respecto de la localización desde la que se consumieron los servicios. Lo que justifica introducir la localización es la suposición de que el consumo de servicios se valora de forma más detallada y precisa cuando la distancia es menor.

1.3 Contribuciones

Se propone el uso de técnicas de IA, específicamente la construcción de agentes inteligentes, dado el comportamiento activo de los mismos y, en particular, las características deliberativas de

los agentes tipo BDI (ver más detalle en el Capítulo 2). Dada su naturaleza social, se construye una arquitectura multi-agente adaptativa a cualquier dominio de AmI publicada en (Venturini et al., 2010).

Dicha arquitectura surge como fruto del diseño del sistema, haciendo una hibridación de las metodologías GAIA y AUML, especificas para el desarrollo de agentes inteligentes expuesta en (Venturini et al., 2012b). Se procede a la evaluación de la arquitectura a través de la técnica de conteo de los mensajes intercambiados por los agentes durante sus interacciones, sustentadas en protocolos FIPA predefinidos en la teoría de agentes. Una ontología creada en Protégé hace posible la comunicación de los agentes. Dicha ontología abarca aspectos espacio-temporales como así también conceptos sobre las preferencias de los usuarios.

Luego, se implementa un algoritmo para la adquisición de perfiles de usuario sin la interacción del mismo, utilizando técnicas de inteligencia computacional, en este caso algoritmos genéticos. La obtención del perfil de usuario se debe al estudio de las características de las preferencias del usuario sobre las tareas que puede llevar adelante. Este trabajo fue publicado en (Venturini et al., 2008a)

Finalmente, se formaliza un modelo de reputación, presentado en (Venturini et al., 2012a), para brindar a los agentes usuarios del sistema una herramienta que les permita evaluar la confianza social sobre aquellos agentes proveedores de servicios.

1.4 Estructura del documento

El documento de esta tesis se divide en tres partes. La primer parte contiene la introducción, el estado de la cuestión y el planteamiento del problema.

La segunda parte contiene la propuesta detallada dividida en cuatro capítulos:

- Capítulo 4: Diseño metodológico del SMA para entornos heterogéneos de AmI.
- Capítulo 5: Estructuración de la Información.
- Capítulo 6: Adquisición de perfiles.
- Capítulo 7: Modelo de Reputación basado en localización.

Cada capítulo contiene el detalle de implementación, los experimentos y resultados obtenidos.

En el capítulo 4 se analiza un escenario turístico como ejemplo de entorno de inteligencia ambiental. A partir de ese ejemplo se realiza el análisis y diseño de un sistema multi-agente para dominios heterogéneos. Se describen los agentes de tipo BDI para posteriormente mostrar el diseño de los agentes utilizando una hibridación de las metodologías GAIA y AUML. Se detallan los modelos que llevarán a la creación de una nueva arquitectura multi-agente para entornos de inteligencia ambiental y a la implementación de la misma. La evaluación del sistema se hace mediante un método de conteo de mensajes pesados según la importancia de la información transmitida, y a través de una evaluación subjetiva.

El capítulo 5 sobre estructuración de la información, se enfoca al desarrollo de una ontología para AmI, con explicación de los conceptos que abarca. En el anexo B el lector podrá encontrar

el detalle de los atributos de cada concepto. Además se muestra el uso de bases de datos espacio-temporales en sistemas basados en contexto con ejemplos explicativos, sobre el escenario turístico.

En el capítulo de Adquisición de Perfiles (capítulo 6), se hace uso de heurísticas de algoritmos genéticos para la adquisición de las preferencias del usuario, a través de la cual se puede realizar una lista priorizada de tareas que se le presenta al usuario para brindarle asistencia en sus actividades cotidianas. Se muestra el algoritmo, un ejemplo ilustrativo paso a paso (sobre el escenario turístico) y su programación utilizando la herramienta GOAL. Finalmente se explican los experimentos realizados y los resultados obtenidos. El algoritmo construido se evaluó considerando aspectos de la cantidad de tareas disponibles al usuario y la cantidad de características que el usuario valora para poder darle un peso a la tarea.

El último capítulo de esta parte es el capítulo 7 que trata sobre el sistema de reputación CALoR, siglas en inglés de *Context-Aware and Location Reputation Model*. El modelo contiene información de localización para calcular la importancia de la reputación dada por un agente. En primer lugar se hace una ampliación de la ontología propuesta en el capítulo 5 para dar soporte al sistema de reputación. Seguidamente se detalla el protocolo del sistema y la formulación del modelo. La validación de dicho modelo se realiza mediante un caso de uso. El dominio elegido para validar el modelo es el de los servicios turísticos en una ciudad. Se explica la implementación de los agentes utilizando JADE. Y, finalmente, se realiza la evaluación del modelo a través de simulaciones de tipo Montecarlo, comparando el modelo de reputación considerando e ignorando el concepto de distancia entre los agentes.

Luego en la tercer parte de la tesis se concluye el trabajo, se mencionan las contribuciones realizadas y los trabajos futuros.

Por último, existen, al final del documento, anexos con características técnicas de la implementación de los agentes y la ontología obtenida, y la adaptación de los diferentes módulos a distintos escenarios.

2

Estado de la Cuestión

"The best way to predict the future is to invent it". Alan Kay

PARA introducirnos en el trabajo primero definimos el concepto de Inteligencia Ambiental (AmI) como una inteligencia imperceptible en el entorno, la cual ofrece asistencia en las actividades e interacciones de los humanos. Los cimientos de la AmI se encuentran en la combinación de otras tecnologías: la computación ubicua y los sistemas basados en contexto. Desde este punto de vista, la computación ubicua intenta integrar dispositivos (sensores, dispositivos de cálculo, dispositivos de procesamiento y de aprendizaje) al entorno, haciendo posible al usuario obtener servicios en cualquier lugar, mientras que a través de sistemas basados en contexto se obtiene información acerca de la situación del usuario (localización y actividad que está realizando). Por lo tanto, definimos el contexto del usuario para caracterizar al usuario y su situación (Dey, 2001),(Schilit et al., 1994). El contexto de usuario puede incluir una gran variedad de información a través de sensores tales como la localización temporal y espacial, y el almacenamiento de información como las preferencias del usuario y su perfil.

Este entorno ubicuo es ideal para usar el paradigma de agentes dada su naturaleza autónoma, y otras características especiales propias de los agentes: son proactivos, reactivos, cooperativos y sociales. Los sistemas que se modelan como organizaciones de agentes o sistemas de varios agentes son llamados sistemas multi-agente (Ferber, 1999). Los sistemas multi-agentes soportan interacciones complejas entre distintas entidades, usando un lenguaje de alto nivel. Esta característica es esencial en un Ambiente Inteligente dado que hace posible la manipulación de información heterogénea de las preferencias de los usuarios y de los sensores físicos, lo cual es posible solo si todos los tipos de información se expresan semánticamente en un mismo nivel alto.

Los entornos ubicuos que se mencionan al inicio del párrafo, abarcan desde el hogar, una oficina hasta cualquier espacio público, donde la gente obtenga asistencia en la ejecución de sus tareas de acuerdo a sus preferencias. Asumiendo la existencia de un entorno y la posibilidad de obtener las preferencias del usuario, podemos hacerle sugerencias acerca de ciertas actividades para satisfacer sus intereses. Es en este punto en donde los agentes cobran verdaderamente relevancia, debido a que cada usuario puede poseer un agente ejecutándose en su dispositivo móvil, que en cierta manera actúe por su cuenta, y negocie con otros agentes personales o proveedores de servicios.

En el presente capítulo se realiza un análisis exhaustivo sobre las temáticas expuestas anteriormente, a los fines de delinear un marco de trabajo y comprender en que consiste el trabajo llevado adelante en esta tesis, que abarca en la construcción de un sistema multi-agente que adapta la información del contexto para entornos dinámicos, basándose en la localización y las preferencias o perfiles de usuario.

En las secciones a continuación se estudiará en profundidad cada tecnología implicada para abordar la problemática de esta tesis, haciendo una revisión de los trabajos relacionados a cada una de ellas.

En la sección 2.1 se revisan los distintos términos adquiridos alrededor del mundo para hacer referencia a las nuevas tecnologías de entornos inteligentes; se exponen algunos escenarios propuestos por la comunidad científica para comprender la riqueza de los distintos contextos que deberán contemplar las nuevas tecnologías; se hace un breve resumen de las metodologías y arquitecturas existentes para el desarrollo de sistemas de inteligencia ambiental. Posteriormente se hace una revisión de los sistemas contextuales, revisando definiciones de distintos autores y tipificando el contexto, para finalmente establecer la relación entre inteligencia ambiental y sistemas basados en contexto.

La sección 2.2 contiene la caracterización de los agentes, la manera de comunicarse y los tipos de arquitecturas de agentes existentes; además se hace una diferenciación entre los sistemas referentes a comunidades de agentes: sistemas basados en agentes vs. sistemas multi-agentes; posteriormente se revisan arquitecturas multi-agentes presentadas por diferentes autores y las metodologías existentes que hacen posible el desarrollo de las mismas. Puntualmente, se revisa la arquitectura que sirve de base a la propuesta de esta tesis. Para finalizar esta sección, se revisan las ontologías que dan soporte a las arquitecturas de agentes, facilitando el intercambio de mensajes entre los agentes, incorporando conceptos sobre contexto, localización y preferencias de usuarios. Se realiza una diferenciación entre ontologías para dominios específicos y para dominios genéricos. En particular, se hace hincapié en las ontologías que sirvieron como referencia para la propuesta en el trabajo del tesista.

Finalmente, la sección 2.3 se revisan conceptos relacionados a la adquisición de perfiles, se revisan distintos modelos de confianza y reputación, y específicamente, se exponen los dos trabajos que sirvieron de inspiración para presentar el modelo de reputación de esta tesis: un modelo de reputación que incorpora distancias geográficas y un modelo de reputación basado en cuán reciente fueron las interacciones de los usuarios del sistema. Se concluye el capítulo en la sección 2.4.

2.1 AmI

La Inteligencia Ambiental (AmI, por sus siglas en inglés Ambient Intelligence) (Haya et al., 2005) surge en 1999 a partir de un informe del Comité de Expertos del European Community's Information Society, Technology Programme (ISTAG). "Inteligencia Ambiental" es una frase utilizada para describir un mundo en el cual la inteligencia está embebida en todo lo que nos rodea virtualmente. También se la denomina "el Internet de las cosas", donde la identificación por radio frecuencia (RFID) se incorpora a todos los productos. El mundo del aprendizaje automático y el software inteligente es aquel en donde las computadoras monitorean nuestras rutinas para predecir que haríamos o querríamos en el siguiente instante de tiempo (Wright

et al., 2008). Para referirse a estos nuevos entornos inteligentes, se utilizan distintos conceptos alrededor del mundo:

- En Europa se ha adoptado el término Inteligencia Ambiental (AmI) por la Comisión Europea.
- En América su equivalente es la Computación Ubicua u Omnipresente.
- Mientras que en Japón se utiliza la frase Redes Ubicuas para referirse a la inteligencia integrada al entorno.

Otros autores definen la AmI:

- Gaggioli:
  La Inteligencia Ambiental fue caracterizada por Gaggioli (Gaggioli, 2005) como aquel entorno físico que es sensible y que responde a la presencia de personas. Sus características claves son la inteligencia y lo embebido. El término "Inteligencia" hace referencia al factor de que el entorno digital permite el análisis de contexto, se adapta a si mismo para el usuario o para los objetos que residen en él, aprende de su comportamiento y eventualmente, reconoce expresiones emocionales. Por su parte, el término "embebido", significa que el dispositivo, con poder de cómputo, puede estar integrado en segundo plano en las actividades de las personas, mientras que su funcionamiento e interacción social pasan a primer plano.
- Van Houten:
  "Inteligencia Ambiental" (Ambient Intelligent - AmI) es una frase utilizada para describir un mundo en el cual lo "inteligente" se encuentra embebido prácticamente en todo lo que nos rodea (van Houten, 2005). En este nuevo mundo del software inteligente las computadoras monitorean nuestras actividades, rutinas y comportamientos para predecir lo que nos gustaría realizar en un instante de tiempo con el objetivo de facilitarnos las tareas cotidianas. Es decir, son entornos digitales que son sensibles y que reaccionan a la presencia de personas (van Houten, 2005).

No se trata de una visión puramente técnica, sino de una visión orientada a los humanos. La dimensión emocional es crucial. En este sentido puede ser visto como un espejo del mundo de la computación discreta de Mark Weiser (1993), y de la visión sociológica de la interacción humano-medio de Nass y Reeves (1998).

En (Wright et al., 2008), para dar relevancia al concepto de AmI, hacen referencia a los aportes que da en distintos aspectos de la vida: social, económico, legal y tecnológico, estudiando puntualmente las debilidades de los sistemas que pueden surgir a futuro en este campo. Los autores en este libro analizan con profundidad escenarios en donde la seguridad es un punto crítico.

Los escenarios sugeridos por Information Society Technologies Advisory Group (ISTAG) en (Ducatel et al., 2001) incluyen un conjunto de "factores socio-políticos críticos" los cuales se consideran cruciales para el desarrollo del ambiente inteligente, incluyendo factores de seguridad y confianza. A continuación se describen dos escenarios típicos extraídos de ese trabajo:

María y su P-Com

"María arriba al aeropuerto de un país. Ella sabe que ahora puede viajar más ligera que una década atrás, pues al menos la información personal es totalmente digital, ya no le hace falta cargar con su pasaporte, papeles de visado, ni tampoco con esa cantidad de aparatos electrónicos (portátiles, teléfono móvil, agendas electrónicas, etc). Su sistema de computación para este viaje se reduce tan solo a una unidad de comunicación altamente personalizada, el P-Com que lleva en su muñeca. María ha alquilado un vehículo. Solo tras pulsar un botón de su P-Com, el coche se abre a medida que ella se acerca, ya no es necesaria una llave. Ella tiene una reserva en el estacionamiento del hotel. Esto normalmente tiene un costo adicional, pero su agente se encarga de negociar con el agente de la agencia de alquiler y con la cadena hotelera. Una vez que baja del coche estacionado, un conserje la recibe y la acompaña a su habitación. Cuando ingresa en ella, las condiciones del lugar se personalizan de acuerdo a, por ejemplo, cual es la temperatura habitual, las luces y la música que le gustan, y, hasta, puede elegir algún video en la pantalla. Pero decide llamar a su hija, para lo cual utiliza el control remoto e inicia una video conferencia a través de dicha pantalla."

Dimitiros y el Digital-Me

"Son las cinco de la tarde. Dimitiros, un empleado de 32 años de antigüedad del mejor restaurante multinacional, se encuentra en la cafetería con su jefe y algunos colegas. No quiere excederse en esa pausa, por lo tanto se encuentra constantemente recibiendo llamadas y mails. El se siente 'orgulloso' de estar en comunicación con 'la humanidad', al igual que muchos de sus amigos. Dimitrios se está agotando, embebidos en la ropa (o en su propio cuerpo), una voz activa su *gateway* o avatar digital, familiarmente conocido como 'D-Me' o '*Digital Me*'. Un D-Me es un dispositivo de aprendizaje, aprendizaje acerca de Dimitiros y su interacción con el medio ambiente. A la vez este es un dispositivo que actúa ofreciendo otras funcionalidades, como ser comunicación, procesamiento y la toma de decisiones. Dimitiros ha tenido que configurar algunos detalles, pero solo al principio. Sabe que hoy en día puede actualizar esos datos iniciales periódicamente, pero no lo hizo. Pues confía plenamente en que su D-Me lo ha hecho por él, se basa en las reacciones inteligentes de este dispositivo. En un momento el D-Me de Dimitiros recibe una llamada de su esposa, y le responde explicando el porqué de su retraso. Mientras que también llega un mensaje del D-Me de una persona anciana que se encuentra en la estación de metro y ha olvidado en casa su medicina, y desea saber cual es el camino más corto para encontrar una farmacia que disponga de la misma. Como el D-Me de Dimitiros sabe que es la misma que utiliza su dueño, se contacta con el D-Me de este hombre para aconsejarle."

Se observa en ambos escenarios como la tendencia del futuro es una nueva sociedad de la información totalmente digitalizada e inteligente, que asiste al usuario mejorando su estilo de vida.

2.1.1 Metodologías de AmI

En (Jae-Hyung et al., 2005) se describe una metodología para desarrollar servicios de computación ubicua, la cual consiste en los siguientes pasos: (1) analizar el estilo de vida actual del usuario, (2) analizar la conexión de las actividades básicas, (3) buscar nuevos objetos y (4) finalizar la nueva actividad básica. Las características de los servicios de computación ubicua se engloban en los siguientes conceptos: proveer la información correcta, mediante interfaces naturales, maximizando los beneficios del usuario.

En el artículo (Fuentes and Jimenez, 2005) se propone una 'AmI Aspect Oriented Programming' algo así como 'Programación orientada a aspectos (o apariencias) para AmI'. Se definen los siguientes pasos para la programación dentro del entorno de la AmI: 1º gestión de la arquitectura de aplicación, 2º gestión de componentes, 3º administración de aspectos del usuario, 4º administración de aspectos del sistema y por último, gestión de la evaluación de los aspectos.

Dado que el concepto de AmI es novedoso, aún no han surgido muchas metodologías específicas para estructurar el trabajo al momento de diseñar sistemas de este tipo.

2.1.2 Sistemas Context-Aware

Según (Dey, 2001) se define el "contexto" como cualquier información que puede ser usada para caracterizar la situación de una entidad. Una entidad es la representación de una persona, objeto o lugar que se considera importante para la interacción entre el usuario y la aplicación, incluyendo al mismo usuario e inclusive a la aplicación. Existen varios tipos de contextos tales como el tiempo, la localización y los dispositivos de computación. Asimismo, un sistema es basado en contexto (context-aware system) si utiliza el contexto para proveer al usuario de información relevante y/o servicios, donde la relevancia de lo que provee depende de la tarea del usuario. Por su parte (Schilit et al., 1994) expresa que para realizar un correcto análisis de contexto, es fundamental responder a las siguientes preguntas:

- *Who* ("quién", *identity awareness*). Podemos hablar de perfiles de usuario y la medida en la que el contexto los diferencia para lograr el comportamiento adecuado.
- *What* ("qué", *task awareness*). Se centra en lo que el usuario está haciendo, qué tarea realiza y que quiere conseguir. Constituye por tanto los servicios que el sistema le ofrece.
- *Where* ("dónde", *location awareness*). Conocimiento de la localización física, es decir ubicación de personas y objetos que realizarán las tareas.
- *When* ("cuando", *time awareness*). Adquisición y mantenimiento de información sobre tiempo y fecha, horarios estáticos y dinamismo de la agenda de cada usuario.
- *Why* ("por qué" del comportamiento de un dispositivo). Para conseguir comunicar fácilmente con la computadora en la realización de tareas cotidianas, a ser posible, de manera implícita, es decir, sin intervención por parte del usuario.

A continuación se describen los tipos de contextos considerados normalmente y expuestos en (Lee, 2007). Cada uno de nosotros juega roles diferentes a través de la vida, los cuales pueden reducirse a: un rol individual privado, uno profesional o la participación pública. Cada uno de los roles mencionados anteriormente, demandan cierta privacidad, seguridad y confianza acorde a las situaciones en el transcurso de un día. Dentro de este entorno inteligente, es relevante la información de contexto. A continuación se realiza una diferenciación entre contexto de localización, contexto de dispositivos y contexto de usuarios.

- Contexto de Localización

Un servicio basado en localización puede describirse como una aplicación que depende de una cierta localización geográfica. Es decir, se usa la localización de un usuario móvil como parámetro para la provisión de servicio. La información de localización puede ser utilizada en si misma o integrarse a otras fuentes de información para proveer de ventajas en los negocios de las compañías.

- Contexto de Dispositivos

  La mayoría de los Servicios de Internet fueron diseñados para las computadoras de escritorio, pero posteriormente, con el avance en la tecnología, surgieron distintos dispositivos móviles y con múltiples capacidades para acceder a la red de servicios. Es por ello que tanto a nivel hardware como software, tuvieron que realizarse nuevos diseños, tales como el tamaño de la pantalla, la memoria, o la velocidad de procesamiento. Esto muestra la necesidad de adaptar los servicios y el contexto a distintos dispositivos.

- Contexto de Usuarios

  Otro contexto interesante es el que concierne a los factores personales. Para recolectar la información del usuario, un camino típico es el de los cuestionarios. A través de este método el usuario puede expresar sus opiniones o preferencias. Una alternativa a esta técnica es que el sistema puede observar y almacenar de manera implícita el comportamiento del usuario cuando éste se encuentra utilizando un servicio, y luego analizarlo para encontrar patrones de consumo. Esto nos permite ofrecerle servicios de manera personalizada.

2.1.3 Sistemas Context-Aware en Escenarios de Inteligencia Ambiental

Un gran número de sistemas han sido desarrollados para demostrar el uso de esta nueva tecnología que combina los conceptos de los sistemas context-aware y la inteligencia ambiental (o computación ubicua), a los fines de proveer servicios al usuario de manera inteligente. ALZ-MAS (Corchado et al., 2008) es un sistema multi-agente desarrollado para el cuidado de la salud y el monitoreo de ancianos en una residencia, donde por cada doctor o enfermera, se cuenta con un agente, lo que permite el control y seguimiento de los pacientes a través de dispositivos móviles.

E-Tourism (Paganelli et al., 2006) es un sistema basado en localización que utiliza un sistema basado en contexto para ofrecer servicios de acuerdo a la reputación del visitante y, además, soporta la toma de decisiones del usuario. Otro ejemplo de implementación de AmI se muestra en (Bombara et al., 2003) donde la persona que asiste a un museo puede aprender sobre las diferentes piezas de arte con solo colocarse delante suyo, puesto que cada pieza está provista de un sensor que detecta la presencia del visitante, y a la vez, el visitante, puede enviarle información sobre dicho ítem a otro visitante.

Otra instancia de AmI puede observarse en (Chen et al., 2004c) donde el sistema asiste al usuario en su vida diaria, mediante planes y advertencias sobre sus actividades y aprendiendo según se comporta. En el dominio de conferencias también se aplica AmI, tal como podemos observar en (Bravo et al., 2006), donde los usuarios reciben información sobre la sesión de pósters y conferencias en su móvil de manera tal de no perderse ningún seminario relacionado a sus preferencias.

En (Jih et al., 2003) se muestra un entorno de inteligencia ambiental para el cuidado de personas ancianas que viven solas. El sistema los ayuda en la realización de compras de alimentos a través de un agente helpdesk shopping del supermercado. Este agente se comunica con el asistente de cocina, quien provee de un razonable menú dietético para efectuar una compra adecuada de alimentos.

El sistema contex-aware de (Kjeldskov and Paay., 2005) utilizan el concepto de grupo de personas. Cuando dos o mas amigos se encuentran juntos (o cercanos a algún otro), se muestra en el mapa el grupo de personas. En situaciones en las que una persona desea reunirse con otra en un determinado bar, a una hora estipulada, se genera un chat automático. Asimismo, se le provee al usuario de una lista de posible tareas que podría realizar y distintas recomendaciones, como por ejemplo de algunos restaurantes en base a su perfil.

El trabajo de (Räck et al., 2006) propone la arquitectura Amaya que consiste en un sistema de recomendación basado en la sinergia entre el comportamiento del usuario, el contexto de datos y la información semántica, a los fines de habilitar ciertos servicios de acuerdo a la situación del usuario. Amaya está construida en base a cuatro componentes funcionales: el adaptador de datos, el cual provee de una interfaz de usuarios unificada y personalizada; un administrador de perfiles, encargado del manejo de usuarios y grupos y de la personalización de los datos de contexto; el intermediario de perfiles, que es usado por el administrador de perfiles para generar una lista de objetos de perfiles, los cuales pueden ser entidades, tratando de realizar un match entre el perfil del usuario y dicha entidad; y por último, el componente recomendador, que permite el aprendizaje de las preferencias del usuario o del grupo y predice sus comportamientos.

2.2 Agentes

2.2.1 Agentes Inteligentes

Podemos definir a los agentes como entidades interactivas y autónomas que poseen un objetivo y mecanismos para la toma de decisiones. Son entidades que perciben y actúan sobre un entorno. Shoham (Shoham, 1993) propone la definición de un agente como una entidad formada por componentes mentales (típicos humanos) como creencias, capacidades, elecciones y compromisos.

Una de las definiciones más citadas sobre el concepto de Agente, es la de (Wooldridge et al., 2000) extraída de (Mas, 2004): "un agente es un sistema informático situado en un entorno y que es capaz de realizar acciones de forma autónoma para conseguir sus objetivos de diseño".

Los Agentes poseen las siguientes características:

- Adaptables: tienen habilidad para aprender y mejorar.
- Autónomos: toman decisiones en base a sus objetivos, sin interacción humana.
- Colaborativos: trabajan en grupos para conseguir un objetivo común.
- Sociables o comunicativos: pueden establecer comunicación con otros agentes.

- Móviles: poseen la habilidad de migrar a otra plataforma por decisión propia.
- Reactivos: actúan de acuerdo a las percepciones del entorno y deben reaccionar.
- Temporalmente continuos: porque mantienen su identidad y estado en largos periodos de tiempo.
- Personalizados: contienen atributos que muestran su comportamiento mas humano.
- Pro-activos: deben cumplir sus propios objetivos, con iniciativa propia.

Ahora bien, los agentes se diseñan, construyen y comunican bajo los siguientes aspectos:

- Ontología: construye el glosario semántico de conceptos usados por los agentes para comunicarse.
- Protocolo de comunicación: define el lenguaje de comunicación entre agentes y el formato del mensaje.
- Infraestructura de comunicación: hace referencia a los canales de comunicación entre agentes.
- Protocolo de interacción: describe las convenciones de interacción entre agentes.

En los siguientes apartados profundizaremos el tema de la comunicación entre agentes, pero antes se hace una revisión a la problemática de los sistemas en tiempo real.

2.2.1.1 Comunicación entre Agentes

Los agentes pueden comunicarse de dos maneras, una forma procedural y otra declarativa. En la Forma Procedural, un agente ejecuta un código que lo haga dejar datos que mas tarde sean leídos por otro agente. Por su parte, en la Forma Declarativa, un agente se comunica con otros directamente a través de sentencias declarativas. De acuerdo al camino que ha de seguir esta investigación, haremos hincapié en la forma declarativa. En este tipo de comunicación se construyen protocolos de comunicación y una estructura del mensaje. Algunos ejemplos de especificaciones son KQM (Knowledge Query and Manipulation Lenguaje) (Finin et al., 1993) y ACL (Agent Communication Languaje) (FIPA, 2002).

ACL es un lenguaje que permite la comunicación entre Agentes Distribuidos. Un mensaje en ACL es una expresión SQML que consiste en una directiva de comunicación y un contenido semántico en KIF (Knowledge Interchange Format, o sea que los argumentos de expresiones son términos expresados en KIF). Los tres componentes principales de ACL son:

- Vocabulario - Ontologías: El vocabulario del diccionario ACL tiene palabras escritas en lenguaje natural y una anotación formal (escrita en KIF que se describe a continuación). El término ontología se utiliza para definir la especificación de una conceptualización, permite hacer una descripción de los conceptos y relaciones que pueden formar parte del conocimiento de un agente o de una comunidad de agentes. Un compromiso ontológico entre agentes es un acuerdo para usar un vocabulario respecto a la teoría especificada en la ontología.

- KIF (Knowledge Interchange Format): KIF (Formato de intercambio de conocimiento o lenguaje interno) es una versión en notación prefija del cálculo de predicados de primer orden. KIF permite expresar datos simple e información complicada mediante el uso de términos complejos, para lo cual incluye una variedad de operadores lógicos y los operadores (? y ,).
- KQML (Knowledge Query Manipulation Data.): KQML (Finin et al., 1995) define el formato de los mensajes y el protocolo que los maneja para permitir que los agentes intercambien información con otros agentes. KQML cumple con los estándares de FIPA.

Finalmente, en cuanto a los mensajes ACL, dentro de su estructura, requieren de los siguientes conceptos: *performative*, indica el tipo de mensaje (Inform, Query, Propose, etc); *addressing*, indica cual es el agente que recibirá el mensaje y aquel que lo enviará; *content*, este es el contenido principal del mensaje, es la información sobre la que la *perfomative* expresa una actitud; *conversation ID* usado para seguir una conversación; *lenguaje* es el nombre del lenguaje de representación del parámetro 'content'; *ontology* es el nombre de la ontología usada en el parámetro 'content'; *reply with*, si el emisor espera respuesta, es la etiqueta para dicha respuesta; *in-reply-to* es la etiqueta para la respuesta a un mensaje anterior; *sender* es el nombre del emisor del mensaje; *receiver* es el receptor del mensaje.

2.2.1.2 Clasificación de Agentes

Existen distintas clasificaciones de agentes expuestas por ejemplo en (Molina and Corchado, 2001). Para entablar la comunicación entre agentes, se necesita de agentes intermediarios especializados en conocer las posiciones de los otros y los servicios que ofrecen, nos basaremos nosotros en esta clasificación de agentes extraída de (Mas, 2004):

- Facilitadores: coordinan otros agentes subordinándolos a cambios de sus servicios.
- Mediadores: utilizan servicios de varios agentes para construir servicios de más alto nivel.
- Broker: reciben y resuelven peticiones usando servicios de otros.
- Buscador de pareja: si recibe una petición, devuelve el ID del agente que tenga el servicio mas adecuado a sus necesidades.
- Páginas amarillas: contiene el listado de los servicios y los agentes que los proveen.
- Pizarra: recibe peticiones que serán entregadas a otros agentes.

2.2.1.3 Arquitecturas Internas de los Agentes

A continuación se describen los distintos tipos de arquitecturas de agentes, de acuerdo al modelo de razonamiento que utilizan, extraídas de (Wooldridge and Jennings, 1995):

Deliberativas: usan modelos de representación simbólica del conocimiento. Las decisiones se toman usando mecanismos de razonamiento lógico. Al elegir esta arquitectura, hay que buscar la descripción simbólica del sistema y luego integrarla al agente para que razone y ejecute las

tareas en los tiempos establecidos. El estado interno de los agentes se clasifica en tres estados mentales: creencias, deseos e intenciones según (Rao and Georgeff, 1991).

Reactivas: se caracterizan por no tener un modelo simbólico como elemento de razonamiento. Suelen estar organizados en capas de menor a mayor abstracción. La mayor aplicación está en la robótica, dado que al actuar en un entorno impredecible, deben re-planificar y adaptarse continuamente.

Híbridas: combinan la arquitectura deliberativa con la reactiva. Surge una propuesta en la que un agente posee dos subsistemas:

- Uno deliberativo, que utiliza un modelo simbólico y que genera planes.
- Uno reactivo, que reacciona a los eventos que tengan lugar y que no requiera de un mecanismo de razonamiento complejo.

Esta arquitectura puede estructurarse en capas. La estructura vertical, en la que una capa tiene acceso a los sensores y actuadores, y la estructura horizontal, en donde todas las capas tienen acceso a los dispositivos. A su vez, cada capa se organiza con diferentes niveles de abstracción:

- Reactivo (el más bajo), toman decisiones en base a los estímulos recibidos en tiempo real.
- Conocimiento (intermedio), se olvida de los datos que recopila el agente y se centra en el conocimiento que posee del medio.
- Social (el más alto), maneja aspectos sociales del medio (información de otros agentes, deseos e intenciones).

Dado que el enfoque de esta investigación es el uso de los agentes deliberativos, a continuación vamos a explicar el modelo deliberativo BDI, en base a la arquitectura deliberativa. Dado que los agentes que se implementarán necesitan razonar sí o sí sobre el contexto en el que se encuentran, no se opta por otro tipo de arquitectura (reactiva o híbrida).

2.2.1.4 Modelo Deliberativo BDI

En la arquitectura BDI (Belief-Desire-Intention) los agentes se construyen con los estados mentales: creencias, deseos e intenciones. Desde el punto de vista de la Inteligencia Artificial, las creencias (en inglés *belief*) representan el conocimiento que se tiene del entorno, mientras que desde el punto de vista de la informática en general, son la forma de representar el estado del entorno (por ejemplo, el valor de una variable). Por su parte, los deseos (*desires*) son los objetivos. En términos informáticos es el valor de alguna variable dentro de una expresión lógica. El software convencional está orientado a tareas no a objetivos, es decir, solo ejecuta procedimientos, pero no recuerda, por lo tanto no puede recuperarse ante fallos. Los objetivos representan algún estado final deseado. Y finalmente, las intenciones (*intentions*) son un conjunto de caminos de ejecución (*threads*) que pueden ser interrumpidos si hay cambios en el entorno. Es decir que para seguir un plan de acción es necesario además de tener creencias y deseos, tener intenciones para poder re-planificar ante los cambios del entorno. El sistema

almacena sus intenciones. Las intenciones conducen a acciones. Si una intención es una acción simple, el agente la ejecuta. Por lo tanto, se dice que los agentes BDI son aquellos preparados para trabajar con creencias, deseos, intenciones y planes, para enfrentar al mundo real.

En el modelo BDI de Rao & Georgeff (Rao and Georgeff, 1991), quienes consideran necesarias y suficientes a las creencias, deseos e intenciones para modelar un sistema se definen:

- *Creencias*: componente del sistema que almacena la información del entorno en el que está el agente inmerso, a los fines de modificarlo en caso de que el entorno cambie.
- *Deseos*: en un agente existen varios objetivos que pueden ser incompatibles entre sí, pero deben ser priorizados.
- *Intenciones*: es el componente que representa el camino elegido por el agente para alcanzar los objetivos. Como el entorno puede cambiar, puede ser necesario tener que cambiar el curso de las acciones.

2.2.2 Sistemas Multi-Agentes

De acuerdo a la teoría de agentes, se puede definir a una organización de agentes como un sistema basado en agentes o un sistema multi-agentes. En los sistemas basados en agentes, el sistema se modela usando el concepto de agentes, pero su implementación no necesariamente se hace con agentes. Es decir, se usa el concepto de agentes como método de abstracción. Por su parte, un sistema multi-agentes se diseña e implementa bajo el concepto de agentes, y los mismos se comunican entre sí para alcanzar la funcionalidad deseada. La misma puede estar basada en modelos de cooperación, coordinación o negociación.

La definición de Gasser (Gasser, 2000) propone a los sistemas multi-agentes como aquellos Sistemas que coordinan de manera inteligente una colección de "agentes" autónomos. Coordinan los conocimientos, metas, propiedades y planes para tomar una decisión y resolver un problema.

En (Mas, 2004) se define un sistema multi-agentes como aquel sistema que permite la gestión inteligente de un sistema complejo, coordinando los distintos subsistemas que lo componen e integrando los objetivos particulares de cada subsistema en un objetivo común.

El término "sistemas multi-agente" (SMA) (Ferber, 1999) se aplica a los sistemas que comprenden los siguientes elementos:

1. Un entorno, E, es un espacio el cual generalmente tiene un volumen.
2. Un conjunto de objetos O. Estos objetos están ubicados en un momento dado para asociar cualquier objeto con una posición en E. Estos objetos son pasivos, es decir, ellos pueden ser percibidos, creados, destruidos y modificados por los agentes.
3. Una asamblea de agentes, A, que son específicas de objetos, representando las entidades activas del sistema.
4. Una asamblea de relaciones R, que crean un enlace desde un objeto a otro.
5. Una asamblea de operaciones, Op, que hacen posible al agente A la percepción, producción, consumo, transformación y manipulación de objetos desde O.

6. Los operadores con la misión de representar la aplicación de esas operaciones y de las reacciones del mundo en ese intento de modificaciones.

El corazón del sistema multi-agente es el entorno. El agente se define asimismo como la imagen de este entorno, en base a la percepción del espacio físico y de la comunicación directa que recibe.

2.2.2.1 Arquitecturas de los Sistemas de Agentes

Efectivamente, los beneficios de la AmI en los últimos años se han visto plasmados en diversos ámbitos, realizando aportes relevantes en todos los aspectos de la vida, como ser entornos sociales (Moreno et al., 2003), (Kjeldskov and Paay., 2005), educativos (Gu et al., 2004a), culturales (Bombara et al., 2003), (Sashima et al., 2004a), turísticos (Paganelli et al., 2006), en la medicina (Corchado et al., 2008), entre tantos otros. Por otro lado, cada uno de estos entornos, nos ha permitido realizar distintos enfoques: público (un turista que llega de visita a una ciudad (Paganelli et al., 2006)), privado (un enfermo que requiere de cuidados especiales (Corchado et al., 2008)) o profesional (una persona que asiste a una conferencia de su interés (Bravo et al., 2005)). Dichos ambientes son propicios para la aplicación del paradigma de agentes inteligentes utilizado ampliamente en la AmI.

De acuerdo a (Molina and Corchado, 2001) un sistema de agentes cooperante debe estar formado por un conjunto de agentes con habilidades propias con el fin de cumplir con el objetivo común del sistema multi-agente. Las habilidades de un agente consisten en que sea capaz de adquirir datos, comunicarse con otros agentes o servicios, planificar y actuar. Asimismo, cada agente del sistema multi-agente, tiene conocimiento limitado en cuanto al entorno o al objetivo general del sistema, y por otro lado, tienen una especialidad. Es decir, un agente es especialista en realizar una tarea determinada, tanto en capacidad de proceso como en habilidad. De esta manera se descentraliza la toma de decisiones, dado que también la información está distribuida.

Como consecuencia de ello, es sumamente necesaria la coordinación en la comunicación entre los agentes, de tal manera de llevar a cabo las tareas en forma coordinada de acuerdo al/los objetivo/s del sistema. Para ello también es relevante saber distinguir cuales son las funciones que realiza el agente y qué funciones debe llevar a cabo el sistema (Molina and Corchado, 2001).

En (Sashima et al., 2004b) se construye un sistema multi-agente (CONSORTS) para entornos de inteligencia ambiental. La arquitectura CONSORTS se encuentra compuesta por un agente personal, en cada dispositivo del usuario, el cual se comunica con otros dispositivos, por un agente de razonamiento espacio-temporal y por un agente de servicios. El agente de servicios administra los servicios web. Mediante una base de datos espacial y una ontología, se manipula la información de localización. Como escenarios de aplicación en el mundo real, se describen hogares inteligentes y espacios de eventos tales como conferencias, exposiciones, conciertos o fiestas.

La arquitectura SOAR nos permite instanciar agentes, los cuales poseen acciones predefinidas. Cada template de agente trae una parte ya codificada para facilitar la tarea al programador, sin embargo se concluye diciendo que el entorno de Eclipse para programar Agentes es mucho mas flexible que tomar un framework existente como SOAR. Se puede seleccionar entre una lista de

comportamientos, los cuales son evaluados antes de asignarlos al agente, dado que puede darse que un agente ya posea ese comportamiento.

Otra propuesta de arquitectura multi-agente para sistemas basados en contexto se expone en (Robles et al., 2007). Los agentes que la componen son el agente tipo usuario y el agente administrador. El tipo de agente usuario puede cumplir tres roles o tener tres comportamientos de acuerdo a su conocimiento: provider, consumer o partner. A su vez los agentes de tipo administrador se encargan de tareas tales como registrar un agente en un sistema, registrar la frecuencia con la cual el agente contribuye, llevar la cuenta del número de veces que el agente realiza un feedback a otros agentes y, por último, registrar la interacción entre agentes. Para finalizar, se describe al agente Creador, el cual tiene los roles de "initiator" y de "creator", para cuando el sistema arranca inicialmente o cuando se crea una nueva comunidad, respectivamente.

Dentro del escenario de Dimitiros y el D-Me expuesto en los escenarios de ISTAG 2010 (Ducatel et al., 2001), se ha montado una arquitectura multi-agentes en (Cozzolongo et al., 2004) para dar soporte al entorno inteligente, que consiste en una organización de un grupo de agentes especializados, encargados de proveer servicios a otros agentes y manejar datos sobre el contexto de interacción dentro del entorno.

En (Corchado et al., 2008) se describe el agente AGALZ desarrollado para la monitorización de pacientes con Alzheimer. Este agente se incorporó al sistema multi-agente ALZ-MAS, el cual gestiona diferentes aspectos dentro de una residencia geriátrica.

2.2.2.2 Contradicciones en el Diseño de SMA

Hay al menos tres contradicciones en los sistemas multi-agentes (Fernández et al., 2007). Primero, hay contradicciones inherentes al sistema bajo estudio, por ejemplo el hecho que los agentes compiten para lograr sus objetivos. Segundo, existe una tensión que tiene su origen en la naturaleza dinámica del sistema multi-agente y su evolución. Por ejemplo, el caso de comercio entre las capacidades de aprendizaje del agente y sus tiempos de respuesta en un entorno cambiante. Tercero, hay errores de análisis causados por desentendidos entre los clientes y los desarrolladores, o por el número y la variedad de puntos de vista y de las entidades del sistema que tienen los desarrolladores.

La Teoría de la Actividad (TA) es un framework para el estudio de las diferentes maneras de las prácticas del hombre y sus evoluciones en un contexto social e histórico (Fernández et al., 2007). Este estudio está enfocado en los conflictos entre individuos y su entorno, lo cual incluye la sociedad. El comportamiento de los individuos no puede estudiarse fuera del contexto, pero al mismo tiempo las actividades de las personas modifican el entorno. Por lo tanto, la Teoría de la Actividad nos provee de una serie de conceptos para representar la situación social con sus cambios, contradicciones y soluciones. La intención de usar la Teoría de la Actividad en la Ingeniería de Software orientada a Agentes es el hecho de que nos provee de un lenguaje entendible y aplicable para los desarrolladores. Con este propósito se utiliza luego la notación UML para representar los conceptos de la TA.

Una contradicción es una desviación de las normas comunes de la sociedad sobre una actividad, o una tensión entre sus componentes, lo que acarrea nuevas formas de actividades. Una contradicción puede tener diferentes pares de patrones relacionados que corresponden a diferentes configuraciones en fuentes originales textuales donde surge la contradicción. La

descripción textual de una contradicción trata de explicar a los desarrolladores que significa la contradicción en el contexto del SMA.

En (Fernández et al., 2007) toman el ejemplo de una situación en donde un agente efectúa una tarea (actividad) que genera una solicitud (una herramienta) a otro agente. Las actividades del usuario y del productor persiguen diferentes objetivos sobre la herramienta, y no debe considerarse el rol que juega esta herramienta en otra actividad. Sin esta consideración, no existe una real colaboración en la red de actividades y ese es el origen de la contradicción. Una solución posible propuesta por la TA en la relación productor-usuario, es crear un contexto en las actividades límites. Son necesarios nuevos elementos relacionados a los objetivos que no se han sido incluidos. Debemos resaltar que las soluciones a las contradicciones proveen indirectamente una posible improvisación. El desarrollador necesita refinar las soluciones para implementarlas en un problema en particular. Para el manejo de contradicciones se mencionan los siguientes pasos. Primero el experto en TA y paradigma de agentes debe desarrollar una librería de contradicciones TA que puedan usarse en diferentes métodos y proyectos. Luego el experto junto a una metodología orientada a agentes debe realizar la transformación de la metodología elegida a los diagramas UML-AT. Esto permite detectar las contradicciones. Posteriormente se chequean las contradicciones y se las personaliza, puesto que los desarrolladores pueden modificar variables en el diseño. Luego, se analizan las entidades y relaciones respecto de la contradicción que se está estudiando. Si es necesario, se piensa en una solución en la cual posiblemente deban incluirse nuevos elementos. Finalmente, se traducen estas especificaciones UML-AT a la metodología orientada a agentes. El ejemplo concreto que utilizan es el de un agente que cumple dos roles en el SMA: *advisor* y *suggester*, y la contradicción es cuándo es mala una información y no debe ser enviada.

2.2.2.3 Metodologías para el desarrollo de Sistemas Multi-Agentes

Con el auge del paradigma de agentes, se desarrollaron diferentes metodologías para el análisis y diseño de sistemas multi-agentes. Asimismo, además de las metodologías en sí, algunos grupos de investigación profundizaron sus estudios en la construcción de meta-metodologías, que recogen diferentes aspectos de las metodologías propuestas por otros autores. En otros trabajos, se propone el uso de metodologías de manera combinada, es decir, toman algunos modelos de unas y diagramas de otras. A continuación se exponen las diferentes metodologías encontradas en la literatura.

ZEUS

Zeus es un conjunto de herramientas para construir aplicaciones multi-agentes distribuidas, descrito en (Nwana et al., 1998). Esta especie de metodología apunta a la construcción de agentes colaborativos. Brinda agilidad en cuanto a tiempo de desarrollo. Los pasos que se plantean en esta metodología son: la configuración de los agentes, la definición de las tareas en las que están involucrados los agentes, la organización de los agentes desde el punto de vista social y la coordinación de los agentes mediante protocolos de interacción. En cuanto a estos últimos, incluye diferentes protocolos que pueden ser elegidos por el diseñador: master-esclavo, contract-net, estrategias de subasta, tiempo para la negociación, reducciones de precio de negociación, entre otros.

De acuerdo a (Gómez Sanz, 2003) esta metodología combina los distintos resultados de

investigaciones de agentes: planificación, ontologías, asignación de responsabilidades, relaciones sociales entre agentes.

PASSI MAS

Si hablamos de PASSI (Process for Agent Societies Specification and Implementation) (Bernon et al., 2003) estamos frente al uso de los estándares UML y FIPA. PASSI cubre todos los pasos del diseño del software desde el análisis hasta las pruebas. Además tiene desarrollado un conjunto de herramientas PTK (PASSI ToolKit).

El meta modelo de PASSI MAS está organizado en diferentes dominios: el dominio del problema (qué se desea capturar), el dominio de la agencia (lo que representa la transición de los conceptos relacionados al problema y la solución de agente correspondiente) y el dominio de la solución (donde se desplegará el sistema implementado). La comunicación en PASSI es expresada con mensajes FIPA a través del protocolo de interacción de agentes (AIP).

RICA

El enfoque de RICA (Role/Interaction/Communicative Action) (Bernon et al., 2003) integra aspectos de ACLs y modelos de organización de agentes. El modelo de organización está especificado en términos de entidades (agentes, roles e interacciones) mientras que las especificaciones de ACL se basan en la definición de acciones de comunicación y protocolos.

CoMoMAS

Este trabajo de (Glaser, 1997) es una extensión de la metodología CommonKADS, para el dominio de sistemas multi-agentes. CommonKAS fue introducida como una metodología para el desarrollo de sistemas basados en conocimiento. Por su parte CoMoMAS es una metodología para el modelado de SMA, cuya base se encuentra en el modelo de agentes. Dicho modelo se construye de acuerdo a las diferentes estructuras de conocimiento: social, cooperativo, cognitivo o reactivo. Para el modelado conceptual del SMA se siguen cinco pasos durante la etapa de análisis: especificación, construcción de modelos, validación, implementación y pruebas. Los modelos que se obtienen como resultado al aplicar esta metodología, y que finalmente nos lleva al modelado de agentes, son el modelo de cooperación que cubre aspectos de comunicación y especificaciones de métodos de cooperación, el modelo de sistema que integra el modelo organizacional y la parte arquitectónica del modelo de diseño. El modelo de tarea se utiliza sin modificaciones. Es decir, los modelos de agente se construyen a partir de componentes que son obtenidos de los cinco pasos de análisis. De cada paso de análisis, se obtiene un modelo conceptual que describe el SMA desde distintos puntos de vista.

MaSE

En MaSE según (Gómez Sanz, 2003), los agente pueden, o no, tener inteligencia. Contempla tanto el análisis como el diseño del sistema. En cuanto al análisis, se realizan los diagramas de objetivos, diagramas de roles y casos de uso. Por su parte el diseño se lleva a cabo mediante la creación de diagramas de clases de agentes, descomposición del sistema en subsistemas e interconexión de los mismos mediante diagramas UML. AgentTool es la herramienta que permite el uso de esta metodología.

INGENIAS

La metodología MESSAGE extiende del paradigma de objetos de la ingeniería de software conceptos para el área de agentes. Define entonces cinco meta modelos (Gómez Sanz, 2003):

agentes, organización, dominio, tareas-objetivos e interacciones. Adopta el Proceso Unificado de Desarrollo de software para realizar el proceso de instanciación de los meta-modelos.

Según Bernon, (Bernon et al., 2003), la metodología INGENIAS también provee de un conjunto de herramientas para el desarrollo del sistema multi-agente (SMA). Es una refinación de la metodología MESSAGE. Sus métodos y herramientas son definidas en términos de meta-modelos, con lo cual un cambio en el meta-modelo es fácilmente adaptable a las herramientas y métodos. El meta-modelo está estructurado desde cinco puntos de vista: organización, agente, interacción, tareas/objetivos y entorno. Para el modelo de interacción, se propone el uso de los diagramas de interacción de GAIA o de AUML.

Una ventaja que salta a la vista de esta metodología es que se dispone de una herramienta visual para el diseño, denominada GRASIA. Además es posible generar código para distintas plataformas de agentes como ser JADE, Robocode, entre otras.

INGENIAS es una metodología mas representativa para un modelo que necesite de la cooperación de agentes para realizar una tarea específica, está mas orientada a procesos industriales o robótica. Parece de aplicación a grandes proyectos. Sus esquemas son mucho mas complejos, quizá robustos para desarrollos de gran tamaño. Alguno autores (Gómez Sanz, 2003) hacen referencia a que a pesar de ser una metodología probada, queda por demostrar su viabilidad en proyectos reales.

GAIA

GAIA es una de las primeras y únicas metodologías puras de agentes. En esta metodología descrita en (Wooldridge et al., 2000) se considera el sistema multi-agente como una organización de entidades que interactúan entre sí. Los pasos a seguir se estructuran en dos fases: fase de análisis y fase de diseño. En la fase de análisis se realizan las especificaciones del sistema mediante los modelos de interacción y de roles en un alto nivel de abstracción. Los roles agrupan cuatro aspectos (Gómez Sanz, 2003): responsabilidades del agente, los recursos que se le permite utilizar, las tareas asociadas e interacciones. A partir de la documentación obtenida en esta etapa, es posible empezar a caminar por la fase de diseño en la cual se construyen los modelos de agente, servicios y conocimiento. Y siempre es posible regresar a la fase de análisis para refinar los modelos obtenidos, de acuerdo a las necesidades en el diseño.

En (Gómez Sanz, 2003) sintetizan a GAIA como una metodología para el diseño de sistemas basados en agentes cuyo objetivo es obtener un sistema que maximice alguna medida de calidad global.

Según (Bernon et al., 2003) un agente en GAIA es una entidad que puede jugar diferentes roles; un rol es un comportamiento específico que será jugado por un agente, definido en términos de permisos, responsabilidades, actividades e interacciones con otros roles. Lo que se ha logrado con GAIA es una buena representación de la organización de agentes, en la cual se muestran la estructura y las reglas de la organización, estando ambas estrictamente relacionadas.

GAIA presenta dos inconvenientes: en primer lugar, a pesar que permite un alto nivel de abstracción, no se guarda relación entre los diferentes diagramas del modelado. Por otro lado, no lleva directamente a la implementación del sistema, lo cual es costo en cuanto a tiempo. Según (Gómez Sanz, 2003), una desventaja de la metodología es la falta de una herramienta de soporte.

En (Fuentes et al., 2007) se detalla la aplicación de esta metodología para un sistema multi-agentes basado en contexto que utiliza agentes deliberativos, y fue modelado con BDI. Se realiza el análisis y diseño completo, utilizando GAIA para desarrollar un framework genérico dentro de los sistemas basados en contexto.

AML

AML (Agent Modelling Language), descrito en (Bernon et al., 2003), está basado en las especificaciones de UML 2.0. AML está diseñado para soportar modelos de negocio, especificaciones de requerimientos, análisis y diseño de sistemas de software que utilizan conceptos y principios de SMA. AML tiene una estructura de capas para proveer una sintaxis abstracta, una semántica y una notación para el desarrollo de agentes.

AUML

Agent UML (Huget, 2003) es un lenguaje de modelado gráfico basado en UML. Provee de distintos tipos de representación cubriendo la descripción del sistema, los componentes, la dinámica del sistema y el desarrollo. En (Huhns, 2004) se presentan las herramientas necesarias para representar los agentes en distintos diagramas: protocolos de interacción, modelo de agentes, entre otros. Se expone en detalle la representación de cada agente del sistema, haciendo analogía entre los agentes reactivos y los deliberativos, incluyendo conceptos como los servicios y protocolos que soportan, sus objetivos, conocimientos, normas, deseos, intenciones, planes y acciones, como así también el lenguaje de comunicación que utiliza y con que ontologías trabaja cada servicio. Los diseñadores de sistemas multi-agentes utilizan actualmente Agent UML para representar los protocolos de interacción (Huget and J., 2004). En (Huget, 2003) se presenta una extensión de los diagramas de clases de UML orientados a objetos, para representar agentes.

De acuerdo a (Bernon et al., 2003) si hablamos de agentes móviles, un diagrama de actividades AUML debe expresar cuando el agente tiene que moverse. En UML estas expresiones son posibles usando el diagrama de secuencias.

Una gran ventaja de esta metodología, es que nos lleva directamente a la implementación, mientras que quizá la construcción de los diagramas sea costosa debido al grado detalle que requiere.

En el caso del diseño de la arquitectura multi-agentes SIGMA (Tiba and Capretz, 2006), se ha realizado una combinación de las metodologías GAIA y AUML. Se han implementado los modelos de entorno, organización, roles y servicios de GAIA, a la vez que se han utilizado los diagramas de interacción de AUML para extender el modelo de interacción de GAIA. Estos últimos diagramas están siguiendo los estándares de FIPA, cosa que facilita la implementación de la comunicación de agentes.

Aunque algunos autores consideran esta metodología, como un lenguaje de modelado y no como una metodología propiamente dicha, la autora de esta tesis, asume a AUML como una metodología más.

Meta-Metodologías

En (Bernon et al., 2003) el AOSE-TFG (Agent-Oriented Software Engineering Technical Forum Group) se intenta construir un meta modelo para hacer que todas las metodologías orientadas a agentes existentes sean interoperables entre ellas. La definición de un meta-modelo significa la construcción de un modelo de conceptos que pueda ser usado para describir y

diseñar el sistema actual (Fernández et al., 2007). Las entidades del sistema modelado son instancias de las entidades del meta-modelo. Los autores de este trabajo no apuntan a lograr una unificación de los diseños orientados a agentes, pero suponen que la definición de un único meta-modelo, con un conjunto de definiciones de esos conceptos, puede ayudar a entender las diferencias entre todas las metodologías e integrarlas.

Ahora bien, los componentes básicos identificados para la unificación de los MMM (MAS Meta Model) son: agente, rol, tarea, entorno y organización. La relación que guardan estos conceptos entre sí es que un agente, situado en un entorno, juega uno o mas Roles, con lo cual cada agente tiene ciertas tareas. A su vez una Organización está compuesta por agentes.

(Bernon et al., 2003) analiza las siguientes metodologías para formar la meta-metodología: ADELFE, Gaia, INGENIAS, PASSI, RICA y TROPOS.

Para finalizar esta sección dedicada a las metodologías de sistemas multi-agentes, se destaca la referencia de (Gómez Sanz, 2003) para la elección de una metodología. Si la idea es trabajar con sistemas basados en conocimientos, se debería optar por MAS-CommonKADS. MaSe se utiliza en torno a la experiencia de objetos. Por su parte, Zeus, Ingenias o GAIA son las orientadas a agentes. Finalmente, si en lo que se piensa es en las que tengan herramientas que den soporte, se recomienda optar por Zeus, MaSE o Ingenias.

2.2.2.4 Negociación entre Agentes

Además de la comunicación entre agentes,en una organización de agentes, es necesario analizar la forma de distribución de las tareas desde el punto de vista de la colaboración, la coordinación y la negociación.

La colaboración entre los agentes se entiende por la función de distribuir el "trabajo" (las tareas, los datos y los recursos) entre los diversos agentes que constituyen el sistema multi-agente (Ferber, 1999). En un sistema multi-agente la distribución se realiza en función de los contratos que se acuerda entre los agentes y dicha distribución se mantiene porque el agente se compromete a realizar una determinada tarea. Ese compromiso pasa a ser un objetivo que debe cumplir el agente.

La coordinación de acciones se puede definir según (Malone, 1988) como el conjunto de acciones suplementarias que deben realizarse en un sistema multi-agente y que deben alcanzar unos objetivos. Suponiendo que sólo se tiene un agente con los mismos objetivos (que el sistema multi-agente), éste no podría alcanzarlos. Desde el punto de vista de la cooperación, la coordinación es la manera en que se acomodan las acciones realizadas por cada agente para que lleven a cabo una acción conjunta. La coordinación se produce cuando agentes tienen objetivos diferentes pero dependen los unos de los otros.

Por su parte, la negociación entre agentes sirve de forma mas específica para la definición o identificación de tareas y recursos, reconocimiento de conflictos, resolución de objetivos dispares. Para que los agentes negocien y puedan llevar a cabo procesos de coordinación y resolución de conflictos, es necesario determinar qué lenguaje de comunicación se utilizará, qué directrices seguirá el proceso de negociación y cuál es el comportamiento que se espera de los agentes. Por tanto, en función del problema que se quiera resolver se puede definir la negociación de una forma u otra.

La negociación de acuerdo a los autores (Molina and Corchado, 2001) se puede ver como un principio organizacional utilizado para relacionar adecuadamente tareas y métodos de resolución de problemas. Por negociación se puede entender la discusión en la que las partes interesadas intercambian información con el objetivo de alcanzar un acuerdo. En este marco, la negociación viene definida por tres aspectos importantes: (a) el flujo de información es biridecional, (b) cada negociador evalúa la información desde su propia perspectiva y (c) la decisión final se toma de mutuo acuerdo.

Otra forma de negociar entre agentes es a través de la formación de coaliciones. En el trabajo de (Andriatrimoson et al., 2012) se presenta Coalaa (Coalitions for Ambient Assisted Living Applications) un sistema multi-agente basado en el protocolo de formación de coaliciones. Un agente puede asumir el rol de creador o candidato de una coalición. Las coaliciones las conforman aquellos agentes con intereses en común que pertenecen a un mismo vecindario. El creador de la coalición deberá negociaron con los candidatos la pertenencia o no a la coalición. El creador de la coalición intercambia mensajes con otros agentes candidatos que son potenciales miembros de la coalición, y una vez resuelto el problema que da origen a la coalición, les comunica la resolución.

Aplicaciones de *Pervasive and Persuasive Negotiation*.

En (Khedr and Karmouch, 2004), han desarrollado un protocolo de negociación en el nivel de contexto para facilitar el desarrollo de sistemas context-aware, con aplicaciones personalizadas y la definición de una ontología para representar la información de negociación. La arquitectura distribuida multi-agentes que proponen está compuesta por:

- agentes de servicio y agentes usuarios, dispuestos a negociar,
- un agente administrador de contexto que negocia con los agentes usuarios para lograr acuerdos en el nivel de contexto, y con los agentes proveedores de contexto para plantificar y garantizar los requerimientos de contexto,
- un agente interfaz, integra el razonamiento, resolución de conflictos y coordinación, capturando las señales de los sensores,
- el agente de ontología, que provee de las funciones semánticas necesarias que los otros agentes pueden utilizar para representar y compartir el contexto en el sistema
- los agentes proveedores de contexto, responsable de capturar los datos de la fuente de contexto e interpretarlo si no éste no fuere entendible por otros agentes, utilizando el servicio del agente de ontología.

La negociación se implementa a través de lógica difusa y el paquete de negociación en lenguaje ACL, provisto por FIPA en JADE.

En (Ramchurn et al., 2004) se realiza una propuesta sobre negociación en entornos de computación ubicua y los autores se enfocan en la búsqueda de una solución a la privacidad del usuario, es decir, que el sistema no sea intrusivo. La negociación en los entornos ubicuos, es a lo que denominan '*pervasive negotiation*'. El sistema debería adaptarse a los intereses y contexto actual del usuario en diferentes circunstancias como ser cuando el usuario se encuentra en una reunión o cuando está solo. Pues debería detectar que se trata de una reunión de trabajo y

en lo posible solo enviar mensajes relacionados a la misma para no distraer la atención de los presentes. El usuario debería poder administrar desde su Agente cuándo, cómo y dónde recibir las notificaciones de tal manera que no se convierta en algo molesto o intrusivo. Este artículo está ligado al problema de las interrupciones al dispositivo en donde correrá el Agente Usuario: un beeper, una pda o un portátil. Para esto definen dos conceptos: intrusión e interrupción. A su vez muestran cuatro casos de interrupción en un entorno de computación ubicua.

En (Kwon et al., 2006) se propone una arquitectura de '*pervasive negotiation*' que permita la ejecución del sistema de manera no intrusivo. En este proyecto se ha implementado el prototipo SmartGuide, que es un sistema recomendador que está disponible de acuerdo a la ubicación física del usuario, para efectuar las recomendaciones de acuerdo a su situación actual. A nivel de negociación, permite a sus usuarios y comerciantes comprender y monitorear la estrategia de negociación que están utilizando en la transacción. La estrategia de negociación supone que los agentes proveedores sean proactivos en cuanto a las sugerencias a los agentes usuarios de acuerdo a quién es el cliente y en qué contexto se encuentra, para personalizar las promociones. Para ello, el agente cliente da a conocer cuales son sus preferencias a priori. La estrategia a seguir en este trabajo para realizar una negociación entre agentes, está basada en ciertas reglas de negocio. Entre ellas, se debe tener en cuenta por ejemplo, que cantidad de rounds es lo máximo que el usuario propone para negociar, la cantidad de proveedores invitados a la negociación, o de los cuales quiere el cliente recibir propuestas, y el nivel de delegación, es decir, que anatomía tiene el agente para comprar él directamente un producto sin que el humano intervenga.

Por su parte, la negociación omnipresente (*pervasive*) sucede cuando agentes autónomos intercambian sus propósitos, de las cuales se mantienen copias de respaldo en los argumentos retóricos (tales como las amenazas, recompensas o apelaciones) (Ramchurn et al., 2003). El rol de cada uno de estos argumentos retóricos es persuadir al oponente en la negociación para que acepte la propuesta. Se entiende por 'retórico' que quien hace la propuesta cree que la divulgación o validación del contenido de esos argumentos, que podrían, en algún punto, ser influenciados por la evaluación del oponente. En este trabajo se analiza la negociación desde el punto de vista de la definición de argumentos, basándose en los estados mentales de los agentes y en las relaciones de confianza entre ellos. Se modelan los conceptos mencionados al principio, como posibles acciones: amenazas, recompensas y apelaciones.

2.2.3 Arquitecturas de Sistemas de Agentes para AmI

En (Sashima et al., 2004b) se construye un sistema multi-agente (CONSORTS) para entornos de inteligencia ambiental. La arquitectura CONSORTS se encuentra compuesta por un agente personal, en cada dispositivo del usuario, el cual se comunica con otros dispositivos, con un agente de razonamiento espacio-temporal y con un agente de servicios. El agente de servicios administra los servicios web. Mediante una base de datos espacial y una ontología, se manipula la información de localización. Como escenarios de aplicación en el mundo real, se describen hogares inteligentes y espacios de eventos tales como conferencias, exposiciones, conciertos o fiestas. Captura la posición del cliente a través de los sensores de la red y el razonador espacio-temporal gestiona las distancias. Cuando se satisface una relación geométrica, el agente razonador advierte al agente de servicios, y éste envía un mensaje al usuario proveyendo los servicios adecuadamente. El principal problema que se detecta es que los servicios se ofertan

ignorando cualquier tipo de preferencia del cliente.

En (Berger et al., 2007) se describen diferentes escenarios de ambientes inteligentes: una guía personal de la ciudad, un hogar inteligente, el cuidado de la salud e infraestructuras de megaciudades. Se propone la arquitectura AmIRA (AmI Referente Architecture). Es una arquitectura de múltiples capas, en donde cada capa depende de los servicios que proveen las capas inferiores, y no tiene conocimiento sobre las capas superiores. Asimismo, se realiza una distinción de diferentes modelos: modelo del problema, modelo de situación y modelo del mundo. Dentro de la arquitectura se menciona que un SMA sería una herramienta relevante dadas las condiciones de coordinación necesarias en los sistemas distribuidos. Se muestra como instanciar dicha arquitectura en diferentes escenarios. Dentro de las capas de esta arquitectura se distinguen dos en especial. La primera encargada del razonamiento y actos (comportamiento basado en reglas, planificación, optimización lineal, eventos y enrutamiento), y la segunda, de la percepción y comprensión (razonamiento, control de los RDF, Marco de Descripción de Recursos (del inglés Resource Description Framework, RDF)). En ambos casos se trabaja directamente con el uso de ontologías.

OAA (Open Agent Architecture) (Martin et al., 1995) es un framework para la construcción de sistemas multi-agentes. Dicha arquitectura está compuesta por los siguientes agentes:

- Agente facilitador: es una especie de agente Servidor encargado de la coordinación en la comunicación y cooperación entre agentes.
- Agentes de aplicación: son los agentes encargados de proveer servicios independientes del dominio tecnológico o dependientes del usuario y contexto. A su vez, cada agente de aplicación puede trabajar con un API o librería específica a los fines de otorgar un servicio en particular.
- Meta agente: son aquellos cuyo rol es asistir al agente facilitador en su tarea de coordinar las actividades de los otros agentes. Son los encargados de dar soporte a dicho proceso mediante el conocimiento o razonamiento sobre un dominio u aplicación específica (reglas, aprendizaje automático, planificación, etc).
- El agente interfaz de usuario juega un rol muy importante. Se implementa como una colección de micro-agentes, cada uno de los cuales monitorea una modalidad de entrada de datos diferentes (punto y clic, escritura manual, gestos con el lápiz, voz), y realiza la interpretación de esos datos.
- Todo otro agente que se encuentre dentro del sistema, es considerado un agente cliente. Que tras una invocación, se conecta al agente facilitador, el cual es conocido como facilitador padre.

El ejemplo que utilizan para hacer uso de esta arquitectura de agentes consiste en un entorno de oficina virtual. Lo que falla en esta arquitectura es el hecho de que el agente cliente es como externo al sistema a pesar que se le asigna el agente facilitador. Esto trae como consecuencia la pérdida de interés sobre las preferencias del cliente, lo cual es relevante a la hora de ofrecer servicios.

En (Dogac et al., 2003) se propone una arquitectura para ambientes inteligentes basada en agentes. Se describe un framework en el cual si un usuario solicita un servicio web, un

agente realiza la consulta al servicio de registros para encontrar explícitamente quien cumple con dichos requisitos.

Actualmente, existen numerosas arquitecturas multi-agentes lo que hace casi imposible poder innovar sobre ellas. Es por ello que se trata de adaptar arquitecturas existentes a las necesidades del problema que se aborda. La arquitectura multi-agentes de (Fuentes et al., 2007) es la que se propone como base de la arquitectura de esta tesis, principalmente por pertenecer a una colega del grupo de investigación de la tesista. La arquitectura que proponen, surge del estudio de un caso de uso del aeropuerto de Barajas, pero se adapta a cualquier otro dominio con características contextuales predominantes. Son tres los tipos de agentes involucrados (ver figura 2.1):

- Agente Cliente: tiene como objetivos la negociación con agentes proveedores, la recomendación de servicios a otros agentes clientes, confiar en agentes clientes y manipular y mejorar su perfil interno, de acuerdo a los servicios recibidos.
- Agente Proveedor: la funcionalidad de este tipo de agentes es establecer diálogos con los agentes clientes, realizando tratos con ellos a los fines de brindarles servicios de acuerdo a sus preferencias y perfil de usuario. La información contextual se distribuye entre los proveedores.
- Agente Central: es el encargado de detectar, registrar y quitar del registro a los usuarios del sistema, mejorando los perfiles de usuario, filtrando y notificando a los proveedores que tienen cerca a un potencial cliente. El agente central maneja las principales actividades del sistema.

Sobre este último agente es en el qué más se trabaja en el capítulo 4, tratando de disgregar sus funciones en otros agentes, dada la cantidad de usuarios concurrentes, que simultáneamente pueden estar solicitando o recibiendo servicios personalizados, además de la necesidad de constantemente ir chequeando la posición de ellos en el entorno. Se considera que en el agente central, podría darse un cuello de botella ocasionando, por ejemplo, retardos en la entrega de información a los demás agentes, lo que derivaría en una pérdida de confianza en el sistema, por fallas espacio-temporales. Es decir, dependiendo del tamaño geográfico del contexto, un mensaje recibido con segundos de retrazos, lleva al usuario del sistema a ignorar la información recibida, dado que por ejemplo, si la persona se encuentra en la calle, y le llega un mensaje sobre un negocio con ofertas, pero esto sucede dos cuadras después, seguramente el usuario optará por seguir adelante y no ir en busca de la oferta recibida en su móvil. En cambio, si el entorno es dentro del hogar del usuario, posiblemente le dará igual ir de un lado al otro.

2.2.4 Arquitecturas de Sistemas de Agentes Context-Aware

Muchos son los frameworks de servicios context-aware que se desarrollan en la actualidad como los resumidos en (Miraoui et al., 2008) y (Baldauf et al., 2007).

(Plaza., 2004) diferencia a las aplicaciones comunes de aquellas basadas en contexto (context-aware): las primeras se basan en un circuito de entrada/salida, en cambio las segundas deben ejecutarse de manera continua, sin un final, deben tener una identidad persistente en el

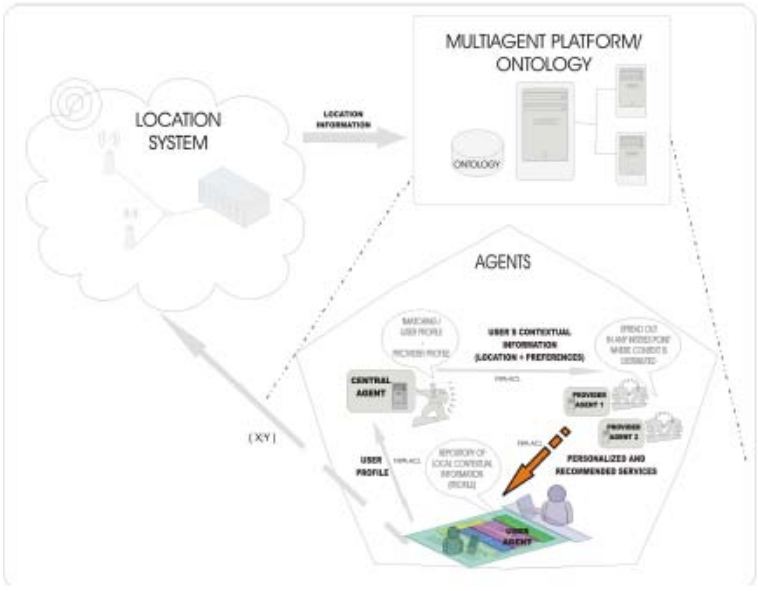

Figura 2.1: Arquitectura multi-agente para AMI.
(Sánchez-Pi et al., 2008)

tiempo y una memoria individual, deben adaptarse y producir respuestas adecuadas cuando algo cambia en el entorno. Las aplicaciones context-aware tienen propiedades que las identifican: ser persistentes, sensibles/receptivas y autónomas, lo cual se refiere a la Computación Continua.

En el trabajo (Smailagic et al., 2007) se estudia un framework de actividades para computación context-aware, como novedad, incorpora características espacio-temporales e interfaces de usuario tanto visuales como auditivas. La información espacial se considera en relación a la posición absoluta y a la orientación, mientras que la información temporal se representa en la planificación de eventos públicos y privados. El objetivo principal del trabajo es minimizar el grado de distracción del usuario en el entorno. Por ejemplo, apagar o silenciar el móvil cuando ingresa a una reunión, al cine o a un hospital. El sistema necesita saber el estado del usuario para adecuarse a sus necesidades. Los investigadores han plasmado en una matriz cuáles son los factores de distracción y considerando dichos aspectos, pudieron clasificar el tiempo de acuerdo al momento en el que se puede interrumpir una tarea. El framework pone a disposición del usuario cuatro aplicaciones:

- *Portable Help Desk*: permite construir mapas según el contexto, por lo tanto, si estamos viajando, se puede visualizar una ruta, si estamos recorriendo la ciudad muestra los distintos restaurants, o, en caso del ámbito laboral, puede mostrar las impresoras disponibles en el área de trabajo.
- *Matchmaker*: entre sus funciones, la principal es encontrar un experto que sea capaz de solucionar un problema en particular reduciendo el tiempo de búsqueda de ayuda. Asimismo, es quien gestiona los grupos de usuario. Ambas aplicaciones mencionadas anteriormente necesitaron previamente de la construcción de dos servicios fundamentales: el servicio de localización y el servicio de planificación.
- *Privacy Guard*: protege la información de usuario e implementa todas las políticas de

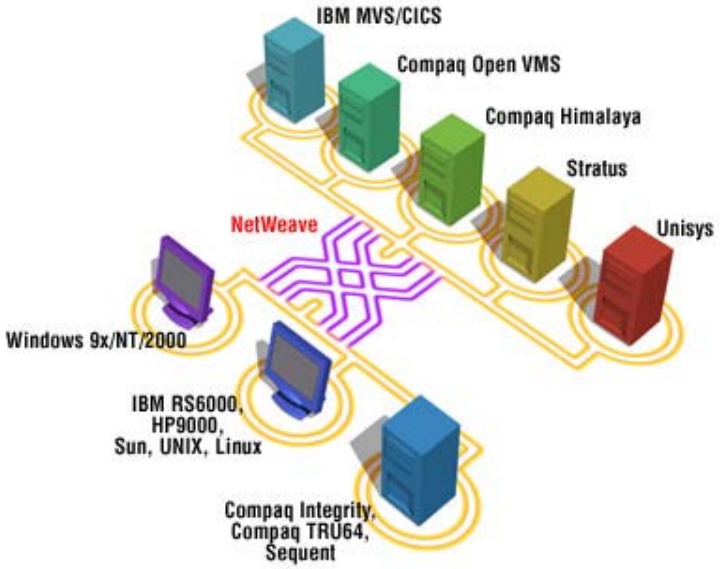

Figura 2.2: Arquitectura para la provisión de servicios en AmI.
(Soldatos et al., 2006)

seguridad del sistema.

- *Agentes Context-Aware*: manejan la información de contexto relevante (leen mails, calendarios, boletines), resuelven conflictos con calendarios de otros usuarios a fines de planificar encuentros. En este framework existen tres tipos de agentes context-aware: agentes de notificación, de recordatorios y de recomendaciones.

La arquitectura de este trabajo es una mejora a una arquitectura propuesta por IBM, a la cual denominaron Idealink Handy Andy.

En (Soldatos et al., 2006) se especifica una infraestructura para la provisión de servicios en ambientes inteligentes, se detallan mecanismos para controlar sensores y actuadores del entorno, se propone la registración automática a los servicios que brinda el sistema y se modela la información contextual a través de tuplas entidad-atributo, sobre la cual se realiza un proceso de razonamiento aplicando funciones de predicado. Está planteado sobre una arquitectura multi-agente distribuida, funcionando como una capa de abstracción de software, siendo un puente entre el software de bajo nivel y los componentes de hardware como puede verse en la Fig. 2.2.

Dicha arquitectura se compone de los siguientes tipos de agentes:

- *Core Agents*: provee mecanismos de comunicación para entidades distribuidas y se encarga del control de los sensores distribuidos en el entorno. Estos agentes son:
  - *Device Desktop Agent*: implementa la interfaz de usuario
  - *Device Agent*: permite la comunicación entre los dispositivos y el framework
  - *Personal Agent*: hace de proxy entre el usuario y el agente

  - *Agent Manager*: permite que se agreguen servicios al sistema

- *Basic Services Agents*: incorporan la lógica de servicio en los cuartos inteligentes, componen situaciones y controlan también sensores y actuadores. Estos agentes son: *Situation Watching Agent, Smart Room Agent, Knowledge Base Agent* (es quien permite a los agentes del framework acceder a información ubicua (sensores, actuadores, etc)) y *Knowledge Base Server* (quien gestiona la ontología).

- *Ubiquitous Service Agent*: implementan de forma no-obstrusiva la lógica de servicio de los diferentes servicios del contexto. Algunos de estos agentes fueron tomados del proyecto CHILD: *Memory Jog Agent (MJA), Connector Agent, Socially Supportive Workspaces Agent y Attention Cockpit Agent.*

La arquitectura la componen un middleware encargado del control de los servicios y actividades, control de acceso a dichos servicios y modelado de contexto en base a las percepciones del entorno. Por otra parte, está compuesta por sistemas de identificación del hablante, detección de rostro y de la actividad que el usuario está realizando. Esta plataforma está implementada en JADE siguiendo el protocolo FIPA.

El servicio de Localización mediado propuesto en (Sashima et al., 2004a) posee una arquitectura multi-agente, cuya principal función es la coordinación de servicios. Está compuesta por los siguientes tipos de agentes:

- Service Provider: responsable de los web services, registra las capacidades y áreas de los servicios físicos.

- Service Requester (Agente Personal): solicita la registración en el sistema

- Location Aware Middle Agent: maneja el modelo espacial y el razonamiento de las relaciones espaciales de objetos en el entorno. Actúa como mediador entre los proveedores de servicios y quienes requieren servicios.

- Device Wrapper Agent: este tipo de agente capta a aquellos sensores que tienen una interfaz de servicio web, proveyendo información acerca de los mismos.

La Figura 2.3 muestra la coordinación entre los distintos tipos de agentes. Los agentes proveedores publicitan sus capacidades y reglas de servicios, por su parte los *Middle Agents* almacenan estas publicidades y sus reglas de servicios además de monitorear la información del sensor recibida del sensor a través de los *Device Wrapper Agents*. Si detectan tal solicitud, enfrentan la solicitud accediendo al servicio. Si el solicitador de servicio acepta la oferta, pide a los proveedores de servicio que suministren los servicios solicitados.

Por su parte los *Middle Agents* coordinan varios formatos de información espacial usando meta modelos de información. Gestiona la información espacial basada en: un modelo global, un razonador espacial, un repositorio de información espacial y un intérprete de meta información . Solo manipula información de localización indoor (incluye conceptos como segmento, piso, etc). El prototipo de esta arquitectura fue desarrollado usando FIPA y JADE, e implementado en un museo, donde se reflejan las ventajas de coordinar información de localización. Los agentes son conscientes en este entorno de la distancia entre el usuario y cada obra de arte.

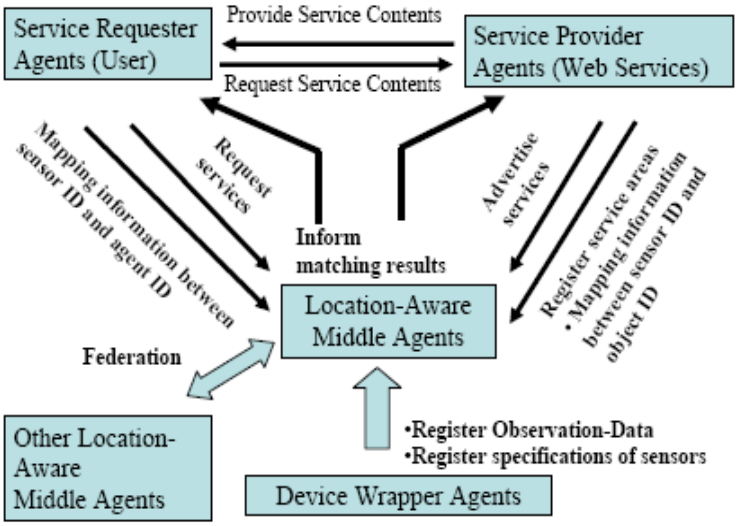

Figura 2.3: Servicios de localización.
(Sashima et al., 2004a)

2.2.4.1 Ejemplos de Sistemas de Agentes para AmI

Un sistema basado en agentes para inteligencia ambiental, es el que se describe en (Masthoff et al., 2007). En este trabajo se muestra cómo combinar las arquitecturas de agentes de inteligencia ambiental con los trabajos de adaptación de grupos para obtener entornos responsables que tengan en cuenta la satisfacción de esos usuarios. Se presenta la manera de modelar el contagio y la conformidad entre los usuarios vecinos, dependiendo del dominio de aplicación. Por ejemplo, para adaptar la música, el contagio y la conformidad del grupo debería ser más alta en ciertos entornos como los pub, pero mas baja en una librería. Para reducir la comunicación, debería ser suficiente para el agente usuario comunicar su satisfacción solo a los otros agentes que representan aquellos usuarios con los que tiene excelente relación, intentando estructurar la coherencia social en un grupo. En síntesis, se modelan las preferencias del usuario en base a grupos y de acuerdo al dominio en el que se encuentren. Cada usuario está representado por un agente como así también cada dispositivo o sensor del entorno. Los agentes de los sensores deberán ejecutar una acción en base al rango de preferencias del grupo de usuarios presentes.

En el trabajo propuesto por (Tewari et al., 2003) se estudia el proyecto MARI (Multi-Attribute Resource Intermediary) y el Wherehoo, para los mismos se intenta describir una arquitectura basada en agentes de manera tal que sea posible dar respuesta al problema de las fuentes de localización específicas utilizando una arquitectura intermediaria en un contexto de contradicciones geográficas. En especial, se analiza el caso de cómo ofrecer recomendaciones de restaurantes de acuerdo a la posición del usuario, y siendo esta misma recibida en sus móviles. En esta arquitectura son relevantes los agentes comprador y vendedor, y se hace hincapié en cómo obtener las preferencias del usuario.

Un clásico ejemplo de arquitectura multi-agente es la descripta en (Bombara et al., 2003).

Esta arquitectura describe el software Kore, el cual consiste en un sistema multi-agentes para asistir a los visitantes del museo corriendo sobre una arquitectura distribuida. Utilizan una combinación de tecnología infrarrojo y wireless. Los problemas que exponen son: la dificultad para conseguir la posición exacta del usuario, la adaptación de la información de acuerdo a las preferencias del usuario, el hecho que el sistema deba adaptar la información de acuerdo al dispositivo del usuario y sus características y que la información debe filtrarse de acuerdo al lenguaje, edad y nivel del usuario.

En (Moreno et al., 2003) se muestra la arquitectura para un sistema multi-agente el cual permite solicitar a los usuarios un taxi en la ciudad. Para ello se construye un agente personal que corre sobre las PDA y se comunica con el resto de los agentes de manera inalámbrica via bluetooth con el resto de los agentes del sistema, a los efectos de encontrar un taxi apropiado para servir al usuario. En la estación de taxi corre un agente denominado centralita, que recibe las solicitudes de los usuarios y se las re-envía a los agentes de taxis. Por otro lado un agente de tráfico les envía a los agentes de taxi la información sobre el estado de las calles (por ej. Si una está bloqueada o si hubo un accidente). Y finalmente el agente visualizador, que recibe información de la estación central, informa sobre las posibles rutas que debería realizar un taxi y muestra esa información gráficamente. Los agentes fueron desarrollados usando JADE-LEAP con las especificaciones de FIPA.

La idea del artículo (Masaud-Wahaishi et al., 2003) es desarrollar un modelo basado en agentes para la integración basada en capacidades (servicios de mediación o intermediario) como una capa de intermediación con la capacidad de soportar diferentes grados de privacidad. La separación en capas de esta arquitectura, implica una adecuada separación de responsabilidades en las aplicaciones para entornos abiertos y heterogéneos. Por lo tanto, un aspecto de diseño importante en este tipo de entorno abierto, es el grado de privacidad que se les otorga a los agentes solicitantes y a los agentes proveedores. Aquí la privacidad se mide entorno a la identidad de la entidad, sus capacidades y sus objetivos. Se construyen entonces varios Agentes Intermediarios con diferentes roles, de acuerdo al grado de privacidad de los demás Agentes del sistema.

AMICO (Susperregi et al., 2004) es una arquitectura multi-agente que provee de movilidad y capacidad de adaptación al usuario, para entornos de aplicación distribuidos. La arquitectura se basa en JADE con especificaciones FIPA y trabaja tanto sobre WiFi como sobre Bluetooth. AMICO asiste al usuario dentro de un laboratorio, brindándole la información que necesita en el momento adecuado y en el dispositivo mas adecuado para cada caso. Para ello cuenta con un agente Broker encargado de atender la solicitud de los otros agentes, buscando a los receptores adecuados para transferibles el mensaje. Mientras tanto, el agente Register, registra cada uno de los dispositivos del laboratorio en la plataforma, permitiendo al resto de los agentes, acceder a los mismos. El sistema de identificación por radio frecuencias, es manejado por el agente Tracker, quien se encarga de realizar el seguimiento de los usuarios y el uso de los dispositivos. El agente Broker gestiona el entorno AMICO del laboratorio, y en base a las preferencias y localización del usuario, envía el agente móvil al dispositivo más adecuado en cada momento. El agente móvil muestra información de producción o mantenimiento correspondiente al perfil del usuario.

SIGMA (Tiba and Capretz, 2006) es una arquitectura multi-agente conceptual que intenta mejorar la integración y coordinación de herramientas y técnicas. Fue diseñado para distintas fuentes de conocimiento, tales como sistemas de soporte, aplicaciones comerciales, sistemas

expertos, técnicas de inteligencia artificial y otras aplicaciones que surjan a futuro. Las principales funciones que provee esta arquitectura son: monitoreo y diagnóstico, interfaz de usuario y almacenamiento y recuperación de datos. Utiliza también FIPA-ACL como lenguaje de comunicación entre agentes.

En (Corchado et al., 2008) se describe el agente AGALZ desarrollado para la monitorización de pacientes con Alzheimer. Este agente se incorporó al sistema multi-agente ALZ-MAS, el cual gestiona diferentes aspectos dentro de una residencia geriátrica. El agente AGALZ es un agente deliberativo basado en una arquitectura de razonamiento (BDI) y además es basado en planificación (CBP), ambas características hacen que tenga una gran capacidad de aprendizaje y adaptación para trabajar en entornos dinámicos. ALZ-MAS utiliza microchips montados en unas pulseras sobre las muñecas o tobillos de los pacientes y sensores colocados en zonas protegidas. El microchip captura información sobre el paciente, la cual puede consultarse a través del agente AGALZ instalado en la PDA del médico o la enfermera.Los agentes AGALZ se asignan a los doctores o enfermeras de la residencia y proveen de información acerca de la localización de un paciente, datos históricos y alarmas. El sistema multi-agentes ALZ-MAS está compuesto por 4 agentes: agente paciente, gestiona los datos y comportamientos de los agentes; agente administrador, este juega dos roles, un rol de seguridad que controla la ubicación del paciente y gestiona las alarmas, y un rol de administrador que maneja la base de datos médica y las citas de los pacientes con los doctores; agente doctor AGALZ encargado de tratar a los pacientes; agente AGALZ que planifica el trabajo diario de las enfermeras dependiendo de las necesidades del día de cada paciente, es el encargado de administrar los perfiles de las enfermeras, las tareas, la disponibilidad horaria, y genera los planes asegurando el cuidado de cada paciente asignado por enfermera.

Parte del proyecto ASK-IT (Ambient Intelligentce System of Agents for Knowledge-based and Integrated Services dor Mobility Imparied users) fue publicado en (Spanoudakis and Moraitis, 2006). Se propone el desarrollo de un asistente personal de viaje basado en las especificaciones de FIPA en [FIPA Personal Travel Assistence Specification]. Algunos de los agentes que participan en esta arquitectura multi-agentes son: un agente de servicios de viajes, un agente intermediario de viajes, un agente asistente de viaje, el agente interfaz, el agente proveedor de servicios, entre otros. El proyecto utiliza JADE para la implementación de las especificaciones FIAP.

2.2.5 Ontologías

Dentro de la Inteligencia Artificial (IA) existen distintas definiciones de ontologías, y algunas de ellas se contradicen. En este ámbito de la informática, se utiliza ontologías a los fines de representar y compartir el conocimiento o estado del mundo actual, a través de conceptos.

En (Noy and McGuinness, 2001) se define una ontología como una descripción explícita y formal de conceptos en un dominio de discurso (clases o conceptos; propiedades de cada concepto y atributos de cada concepto, también llamados slots, roles o propiedades). Una ontología junto con un conjunto de individuos de clases constituye una base de conocimiento.

El trabajo de (Uschold and Gruninger, 1996) incluye la definición de ontologías según Gruber: "Las ontologías son acuerdos sobre conceptualizaciones compartidas, que incluyen marcos conceptuales para modelar el dominio de conocimiento así como protocolos de contenido específico para la comunicación entre agentes y los acuerdos sobre las teorías de representación

de un dominio particular. En el contexto del conocimiento compartido, las ontologías son especificadas en forma de definiciones del vocabulario representativo. Un caso muy simple podría ser un tipo de jerarquía, especificando las clases y la relación entre ellas. Los esquemas de las bases de datos relacionales sirven también como ontologías mediante la especificación de las relaciones que pueden existir en algunas bases de datos compartidas y las restricciones de integridad que las sustentan."

El desarrollo de una ontología incluye la definición de clases en la ontología, también denominados conceptos, la organización de dichas clases en una jerarquía, la definición de slots y la descripción de los valores permitidos para esos slots, y por último, llenar los valores de los slots para las instancias. Ahora bien, es fundamental tener en cuenta algunas consideraciones a la hora de construir nuestras ontologías. Por una parte, revisar otras ontologías, para ver que proponen y que se puede reutilizar de ellas. Por otro lado aplicar algún enfoque para desarrollar una jerarquía de clases (Uschold and Gruninger, 1996):

- Top-down: se empieza definiendo conceptos generales en el dominio y posteriormente se crean subclases.
- Botton-up: se definen las clases más específicas, y se van agrupando en clases de conceptos más generales.
- Un proceso combinando ambos métodos.

También en el estudio de ontologías nos encontramos con dos ramas de investigación. Aquellos grupos que se dedican a la creación de ontologías genéricas o meta-ontologías, y quienes construyen ontologías específicas de acuerdo al dominio de aplicación.

2.2.5.1 Ontología para la Comunicación entre Agentes

Los agentes necesitan intercambiar mensajes para comunicarse unos con otros, para ejecutar tareas globales en un camino distribuido y cooperativo. El intercambio de mensajes requiere que un agente pueda entender a cada uno de los otros. Cuando un agente A se comunica con un agente B, una cierta cantidad de información I se transfiere de A a B a través de mensajes ACL (Agents Communication Languages). Dentro del mensaje ACL, I se representa como una expresión de contenido que contiene el lenguaje y la forma de codificación.

La aplicación de una ontología para un sistema multi-agente, describe el conocimiento de agentes en el proceso de comunicación y esta comunicación se logra utilizando los conceptos ontológicos para mensajes de FIPA-ACL. El modelo de comunicación con FIPA se ha descrito en el apartado sobre agentes.

Las relaciones entre ontologías y el intercambio de mensajes mediante la participación de agentes se logra con FIPA-ACL. Las ontologías deben definir predicados y acciones. En este sentido FIPA provee de diferentes *performatives* para la interacción entre agentes, tales como: *Request, Agree, Inform o Propose*.

2.2.5.2 Ontologías para Dominios Context-Aware

En (Noy and McGuinness, 2001) se sugiere que para desarrollar una ontología específica se debe tener en cuenta cuál es el dominio que la ontología cubrirá, para qué será utilizada y para que tipo de preguntas la información en la ontología deberá responder.

En (Korpipää et al., 2003) se define al contexto como un contexto comprensivo que hace tiene por objetivo comprender la intención del usuario. Se propone un servicio de reconocimiento de contexto, el cual permite a los procesos añadir y/o quitar los servicios de reconocimiento, permitiendo al usuario define las entrada y salida del reconocedor a su vez que establece un ámbito para compartir los contextos reconocidos a nivel más alto, es necesaria la definición de una ontología de contextos. Las propiedades de la ontología propuesta por los autores para manipular el contexto define un atributo tipo de contexto que es un conjunto de categorías y subcategorías de los conceptos que forman una estructura de árbol jerárquica, por ejemplo: medio ambiente: tipo: luz. El atributo "valor de contexto" se refiere al "valor" semántico o absoluto del tipo de contexto o de la descripción verbal del contexto y por último, el atributo confianza que describe la incertidumbre de contexto y se representa como una probabilidad del contexto. Por su parte:

- La fuente describe la fuente del contexto
- Timestamp marca la última vez que ocurrió el evento que marca el contexto
- Los atributos son la información adicional anexa al contexto (la "otra" propiedad).

El vocabulario de contexto se compone de:

- Valores del vocabulario: luz, oscuro, normal, brillante, seco, caliente, frío
- Dos tipos: discreto y continuo
- También se pueda llamar objetivo y subjetivo: 10°C =>Frío

En la figura 2.4 se muestra la ontología de este trabajo que considera los conceptos mencionados anteriormente.

En (Gu et al., 2004b) se modela la información para hogares inteligentes. Para ello se consideran dos niveles ontológicos. En la ontología de alto nivel, se definen los conceptos: persona, actividad, ubicación y entidad computacional, a su vez estas clases derivan o heredan de entidad de contexto. En un bajo nivel ontológico, se definen las clases descendentes de esas clases básicas para el entorno del hogar inteligente. Esta ontología se basa en OWL y soporta la interacción semántica, el compartimiento del conocimiento de contexto y el razonamiento de contexto.

En el trabajo descrito en (Flury et al., 2004), se presentan distintas formas de modelar la localización para sistemas basados en contexto. A partir de esos modelos, se construyen diferentes esquemas de ontologías para representar la ubicación. Estos modelos son: modelo geométrico, modelo basado en la teoría de conjuntos, modelo basado en grafos, modelos semánticos. A los fines de la línea de investigación a seguir en este trabajo, se detallan las

Example context	Context type	Context value	Confidence	Source	Attributes
1	Environment: Sound: Type	Car	1.0	Recognition Service 1	Confidence = Probability
2	Environment: Sound: Type	Elevator	0.9	Recognition Service 1	Confidence = Probability
3	Environment: Sound: Type	Speech	0.8	Recognition Service 1	Confidence = Probability
4	Environment: Location: Building	Outdoors	1.0	Recognition Service 2	Confidence = Probability
5	Environment: Temperature: Absolute	21	1.0	Device Sensor	ValueUnit = Celsius
6	Environment: Humidity	Dry	0.7	Device Sensor	Confidence = Fuzzy membership
7	Device: Activity: Placement	AtHand	1.0	Device Sensor	Confidence = Crisp
8	User: Activity: PeriodicMovement	Walking Frequency	0.6	Device Sensor	Confidence = Fuzzy membership; Speed = 5; SpeedUnit = km/h

Figura 2.4: Estructura de una ontología para entornos contextuales. (Korpipää et al., 2003)

entidades que componen el modelo semántico. En el primer nivel de la ontología se colocan los conceptos: relación de localización, entidad y elemento de referencia. Cada uno de ellos se instancia dependiendo del dominio de trabajo. Por ejemplo, la relación de localización es un miembro, la entidad un usuario meter y por último, el elemento de referencia una habitación como ser la oficina E103. Esta ontología también se basa en OWL.

En las especificaciones de FIPA [FIPA Personal Travel Assistence Specification], se describe una ontología para el desarrollo de un sistema de asistencia al viajero. Esta ontología contempla los siguientes objetos: *trip-summary*, que contiene el detalle del viaje, origen, destino, horas de viaje, via, etc; *location*, incluye toda la información referente a una dirección, calle, ciudad, código postal, país, etc; *travel time*, para describir el tiempo necesario antes y después del viaje; *budget*, información necesaria para un presupuesto, por ejemplo moneda, y los límites que el pasajero está dispuesto a pagar; otro concepto es el de las preferencias generales, que incluye datos como que medios de transporte prefiere y cuales excluye, costos, tiempos, idioma y tipo de mapa; las preferencias específicas del viaje, como clase; *service point* donde se establece un proveedor de servicios para diferenciarlo del concepto *location*; entre otros conceptos específicos referidos al vocabulario necesario para un viaje.

Un trabajo en el que se desarrolla una ontología para entornos de inteligencia ambiental se encuentra expuesto en (Preuveneers et al., 2004). En este trabajo, consideran como aspectos importantes en la información de contexto a los usuarios, analizando su perfil, preferencias y estado de ánimo; el entorno, en el cual interactúa el usuario, siendo relevante la información espacio-temporal y la del propio entorno, como temperatura e iluminación; la plataforma, incluye la descripción de hardware y software de dispositivos específicos, y los servicios que se pueden proveer al usuario.

En (Fuentes et al., 2006a) se define una meta ontología para dominios heterogéneos para sistemas de información basados en localización en sistemas multi-agentes. La idea es cubrir la

mayor cantidad de dominios posibles. Los conceptos que incorpora para representar el entorno son:

- Framework: define el dominio de aplicación dentro de la ontología genérica.
- Participant (participante): representa una entidad que va ha realizar la negociación, dicha entidad puede ser una persona o una compañía.
- Location (localización): representa la coordenada (x,y) de cada entidad en el sistema, ya sea una persona, un objeto o un lugar de interés.
- Spatial Region (región especial): es la representación del espacio, y está compuesta a la vez de dos clases: área y segmento, para la representación del dominio.
- Place (lugar): es un punto en el que se proveen servicios.
- Temporal Region (región temporal): recoge la información de hora y fecha cuando un usuario se encuentra en cualquier ubicación dentro de la region especial, para representar el instante de tiempo.
- Services (servicios): que se pueden proveer.
- Products (productos): como objetos de negociación o recomendación.
- Devices (dispositivos): para saber de que unidad móvil dispone el usuario y poder adaptar las interfaces de presentación de la información según sus preferencias en cualquier lugar y momento.

COBRA-ONT (Chen et al., 2004a) es una ontología desarrollada para la arquitectura CoBrA es una de las más completas en cuanto a su definición en sistemas context-aware, ya que es una colección de ontologías que describe los lugares, los agentes, los eventos y sus propiedades dentro de un dominio de espacios de reuniones inteligentes.

SOUPA es una ontología estándar para aplicaciones ubicuas definida en (Chen et al., 2004b). Los conceptos que incluye son componentes asociados a agentes inteligentes BDI: tiempo, espacio, eventos, usuarios, perfiles, acciones, políticas de seguridad y políticas de privacidad. Los autores proponen esta ontología para extender la arquitectura de COBRA. De esta ontología ha de rescatarse el concepto de políticas y acciones que además de representar dichas normas, sirven para describir el mecanismo basado en lógica para razonar sobre las distintas normas. Por ejemplo, un administrador puede usar políticas para definir quién tiene el derecho de ejecutar qué servicio. Además soporta el modelo BDI a través de la instanciación de la clase o concepto Agente, pudiendo definir sus deseos, creencias e intenciones para caracterizar un estado mental. SOUPA presenta algunas extensiones:

- Meeting & Schedule: para describir información típica asociada a reuniones, eventos y quienes participan.
- Document & Digital Document: para describir por ejemplo la fecha de creación de un documento y el autor.

- Image Capture: cuando un teléfono celular con cámara toma una fotografía, es un evento que puede ser almacenado en este tipo de concepto.
- Region Connection Calculus: define el vocabulario para expresar relaciones espaciales para razonamiento espacial cualitativo.
- Location: para describir la ubicación obtenida por sensores de una persona y objeto, la cual incluye propiedades espaciales y temporales.

Los autores de esta ontología creen que la misma envuelve todo el proceso de investigación en la computación ubicua.

2.3 Representación del Usuario: Perfiles y Reputación

2.3.1 Perfiles de Usuarios

Los intereses del usuario como se describe en (Nijholt, 2007) abarcan el conocimiento del dominio, es decir, los intereses que el usuario tiene sobre una situación en particular y el conocimiento acerca del usuario. Hay varios caminos para la adquisición del conocimiento. Una posibilidad es, simplemente preguntar que características de los productos y servicios son de interés para el usuario en un momento en particular. Otra manera, es utilizando cuestionarios para obtener indirectamente las preferencias del usuario. Esto hace posible el aprendizaje sobre las características del usuario (personalidad, costumbres, preferencias, etc.) y sus relaciones con las decisiones del usuario en ciertas situaciones. Claramente, una vez introducidos en la detección de las preferencias del usuario, estamos realizando investigaciones relacionadas a la inteligencia ambiental.

Para aprender acerca del usuario, su personalidad, valores humanos o actitudes es necesario consultarle acerca de sus preferencias sobre el entorno. Para ello el usuario debe otorgar un valor a cada preferencia (calidad, costo, tiempo) respecto de una tarea específica (ver televisión, jugar a la pelota, hacer check-in en el aeropuerto), en el ambiente donde se encuentre.

Asimismo, existe la necesidad de personalizar la información que se recupera y muestra a los usuarios, de forma que la información presentada sea lo más próxima posible a sus necesidades de información (Samper Márquez, 2005). El perfil de usuario va a contener información modelada sobre el usuario, cuya exploración permitirá al sistema incrementar la calidad de sus adaptaciones.

Los perfiles pueden crearse, de acuerdo a (Samper Márquez, 2005), a partir de diferentes métodos:

- Explícito: los datos son introducidos directamente por el usuario. Una posibilidad es preguntar al usuario acerca de cuales son los productos y servicios en los que está interesando en el momento actual. Otra manera es utilizar cuestionarios para obtener de manera indirecta las preferencias del usuario.
- Colaborativo: el perfil se crea y modifica a partir de su interacción colaborativa con otros perfiles con los que se relaciona, recurriendo a conocimiento específico del dominio y heurísticas inteligentes.

- Implícito: se crea y modifica el perfil automáticamente, recurriendo a técnicas de Inteligencia Artificial.

Por su parte en (Fuentes et al., 2006b) se clasifica la adquisición del conocimiento en:

- Perfil del usuario basado en el conocimiento, permite el aprendizaje de las características del usuario (personalidad, costumbres, preferencias, etc) y su relación con las decisiones del usuario en ciertas circunstancias. Es un método estático, intrusivo y que consume tiempo.
- Perfil de usuario basado en comportamiento, utiliza el comportamiento del usuario para construir y mejorar su perfil dinámicamente.

En (Spanoudakis and Moraitis, 2006) se propone la estructuración de los datos del perfil en dos tipos: dinámicos (aprendidos del comportamiento del usuario) y estáticos (definidos por el usuario). Un problema que surge en la gestión de perfiles y cómo adquirir las preferencias del usuario es el hecho de seleccionar una buena técnica, dado que el usuario debe sentirse motivado a comenzar y concluir la interacción para configurar sus gustos. Este problema es conocido como "paradoja del usuario activo" (Carroll and Rosson., 1987). Normalmente el usuario quiere ingresar al sistema y obtener respuestas rápidamente, y piensa que el hecho de completar formularios acerca de su información personal es una pérdida de tiempo. No tienen el concepto de que eso a futuro servirá como filtro de información.

2.3.1.1 Ontologías para la Adquisición de Perfiles de Usuario

Dentro del marco del proyecto "Morfeo-MyMobileWeb", se define una ontología capaz de capturar el perfil de usuario cubriendo todo sus aspectos, a los fines de aprovechar todos los beneficios que se pueden adquirir una vez conocido el perfil de usuario. La importancia de la adquisición del perfil de usuario reside en la riqueza de dicha información a la hora de proveer servicios. En este sentido, dicha información se utiliza como se expone en (Berrueta, 2007) para adaptar el servicio individualmente, realizar marketing o mostrar al usuario publicidad dirigida. En esta ontología se hace una especial diferenciación entre los datos que corresponden al perfil de usuario y aquellos que forman parte del contexto, pues que no siempre es fácil discernir entre las propiedades de contexto o las del perfil. Las propiedades identificadas se han agrupado por claridad: datos

- personales relativos a la identidad del usuario
- rasgos físicos
- características sociales
- datos geográficos
- educación
- datos profesionales
- relaciones con otros individuos

- rasgos de la personalidad
- vinculación con proyectos, actividades, intereses, etc
- datos económicos
- relación con la administración pública y la justicia
- vinculación con objetos informativos

El enfoque de los autores es destacable puesto que aclaran que estas especificaciones deben tomarse como modelo para la aplicación a realizar, haciendo hincapié en cuidar la privacidad de la información dado que recolecta datos personales delicados tales como la salud.

Una ontología genérica aplicada al dominio de las conferencias se expone en (Plaza., 2004). Esta ontología incluye como concepto a los roles, lo que define el perfil de usuario (ponente, sesión chair, oyente o perteneciente al staff de organización), el concepto ubicaciones (áreas de exhibición, salas de conferencia, áreas públicas), planificación de eventos y conjunto de instrucciones por usuario. Sobre esta ontología se razonan los siguientes componentes del sistema:

- NLG: generador de lenguaje natural
- Sintetizador de voz: recibe un mensaje de audio y se lo reproduce al usuario
- Transformadores de mensajes CAPIA a HTML
- Servicio de sensibilidad: detección de dispositivos por proyectores infrarojos; suscripción de los usuarios para recibir mensajes de los cambios de ubicación de las personas.

2.3.2 Heurísticas con Algoritmos Genéticos para la Adquisición de Perfiles

Un sistema recomendador puede presentar al usuario una sugerencia de las tareas que puede realizar, de acuerdo a sus intereses inmediatos. Los usuarios, normalmente, se muestran preocupados en lo que respecta a ciertos aspectos de privacidad, optando por no revelar sus preferencias a otros usuarios cercanos (Masthoff et al., 2007). De este modo, la obtención de explicaciones explícitas sobre las preferencias de los usuarios en sus actividades inmediatas pasadas, puede ser muy difícil. En el capítulo 6 de esta tesis, se intenta determinar la relevancia de cada decisión pasada tomada por el usuario, a los fines de deducir sus preferencias de manera implícita.

Algunos enfoques para dar una solución a este problema incluyen técnicas de aprendizaje automático, tales como redes neuronales (Morariu and Vlad, 2007) y computación evolutiva (Berlanga et al., 2001). Estas técnicas permiten el aprendizaje automático de preferencias. Es de particular interés para la tesista abordar específicamente la técnica de algoritmos genéticos (en adelante, AG), dentro de la computación evolutiva, dado que permite construir poco a poco del sistema de control mediante la exploración de las variaciones en las interacciones entre el entorno y el propio agente. Un prescendente de la técnica de AG en la adquisición de perfiles es el trabajo de (Chen et al., 2004c).

Los algoritmos genéticos tienen indudables ventajas, en especial la posibilidad de aproximarse a un problema real mediante la búsqueda de las regiones más prometedoras del espacio de búsqueda completo. Los AGs pueden ser más potente cuando se encajan en un método heurístico tradicional (por ejemplo, una búsqueda local) (Chen et al., 2004c). Un AG no sólo optimiza el sistema con precisión y en tiempo real, sino que también encuentra los intereses de los usuarios de acuerdo a información pasada. Usando algoritmos genéticos, los intereses personales se modelan y luego se codifican en un cromosoma, como se describe en (Ujjin and Bentley, 2002). La operación de cruce hace posible descubrir nuevos intereses mediante la búsqueda de los mejores intereses padres (Chen et al., 2004c).

El trabajo propuesto en (Chen et al., 2004c) es un sistema de servicios del hogar personificados a través del uso de AG y extendiendo la interacción hombre-máquina usando RDF. La HCI es distribuida y ubicua. El escenario propuesto es el de una persona en su hogar, cuya agenda incluye: tomar el desayuno, almorzar, trabajar, jugar al tenis y a partir de las siete de la tarde realizar actividades de distracción. El sistema que construyeron, ofrece servicios a las personas de acuerdo a sus listas de intereses, al entorno y a sus redes de contactos.

Los intereses del usuario se almacenan en su calendario. Un usuario puede tener muchos intereses en un período. Se define el interés del usuario como un conjunto de descriptores de los intereses a largo plazo del usuario, que se intuyen de su comportamiento a largo plazo e intereses a corto plazo (positivos y negativos) que se ajustan con cada operación del usuario: requerimientos diarios y cosas que en las que no tiene interés. En este modelo, un interés es un conjunto de descriptores con la forma:

$$(flag^k, (w_p^k, u_p^k); (w_n^k, u_n^k); (dCount^k, w_i^k, u_i^k))$$

donde 'dCount' significa el número de feedbacks recibidos sobre cada interés; 'Flag' es el tipo de feedback recibido sobre el interés; 'w' representa el peso del interés sobre cada descriptor y 'u' es la frecuencia con que cada descriptor es usado en el tiempo.

Un interés es cada una de las actividades mencionadas anteriormente, mientras que los descriptores w, son si el usuario clasifica la actividad como positiva o negativa, si el interés es a largo o a corto plazo, etc.

Una agenda o calendario se diseña como:

$$Calendar = ((Time_1, Interest_1^p), (Time_2, Interest_2^p), ..., (Time_n, Interest_n^p))$$

Por otra parte, Ujjin y Bentley en (Ujjin and Bentley, 2002) especifican aun más datos para poder construir un genético para la generación de perfiles. En este trabajo, ellos diseñan un perfil como una colección de items i suponiendo un escenario de películas vistas por los usuarios. Utilizan un conjunto de datos que incluye como características: el rating de la película, la edad de la persona, su ocupación, y el género de la película: terror, acción, crimen, documental, fantasía, romance, etc.

Una vez construido el perfil del usuario, puede empezar el proceso de recomendación. Dado un usuario activo A, se debe encontrar un conjunto o vecindario de perfiles similares al perfil de A.

Lo que hace el algoritmo es ir evolucionando el fenotipo de preferencias $w_1, w_2, w_3, ...w_n$ de un individuo de la población. Donde cada w_f es el peso asociado a la característica f cuyo genotipo es una cadena de valores binarios. Cada individuo contiene 22 genes que evolucionan a través de un algoritmo genético elitista.

Finalmente, la función de fitness que aplican consiste en una distancia euclidea, para medir la diferencia entre lo que el usuario elegiría para ver y lo que vio otro usuario del sistema con un perfil similar al suyo.

$$euclidean(A,j) = \sqrt{\sum_{i=4}^{z}\sum_{f=1}^{22} w_f * dif_{i,f}(A,j)^2}$$

donde, A es el usuario activo, j es el usuario provisto por el proceso de selección, z es el número de películas que vieron en común A y j, w_f es el peso de la característica f del usuario activo, i es el item de la película en común donde profile(a,i) y profile (j,i) existe, y, $dif_{i,f}(A,j)$ es la diferencia de perfiles en el valor de la característica f del item i de la película entre los usuarios A y j.

2.3.3 Reputación de los Usuarios

Estos modelos tienen su origen tanto en el estudio de ciencias cognitivas como de la teoría de juego. Por parte de las ciencias cognitivas, las relaciones de confianza y la reputación vienen dadas por las creencias y son una función del grado de esas creencias (Esfandiari and Chandrasekharan, 2001). Por su parte, en la teoría del juego, son consideradas probabilidades subjetivas de que un individuo A espera que otro individuo, el individuo B, ejecute cierta acción de la cual depende su propio bienestar (Gambetta, 1990).

Es posible hacer un cálculo del grado de confianza y de la reputación según el tipo de recursos a tener en cuenta. De acuerdo a la naturaleza y manera de interactuar de los agentes, las fuentes de información tradicionales son la experiencia directa y "ganar" información (Sabater and Sierra., 2005). La reputación puede medirse, según el grado de satisfacción, de forma booleana (bueno o malo) o de manera numérica, continua. En lo pertinente a la inteligencia ambiental, las sociedades de agentes son ricas en información en cuanto al aprendizaje que tienen los agentes del comportamiento de otros agentes. Otro tipo de información es el pre-juicio que tienen las personas. En la revisión del tema propuesta por (Sabater and Sierra., 2005), se afirma que trust and reputation son propiedades dependientes del contexto.

El modelo ReGret (Sabater and Sierra., 2005) es un modelo subjetivo, orientado a entornos de e-commerce. La idea es explotar la estructura del entorno para determinar los valores de trust. Usa fuzzy logic para resolver los problemas inherentes a en quien confiar. Cuenta con: un sistema de reputación, la reputación del vecino y la reputación de un testigo. El concepto más relevante del trabajo es el de Imagen. Una imagen es el resultado de un proceso interno del mismo agente, después de su evaluación o experiencia directa sobre otro usuario. Más allá de recolectar información de tipo "social", el modelo no considera ningún aspecto relacionado a la posición física del usuario.

RepAge (Sabater et al., 2006) es un modelo de reputación de la teoría cognitiva. Presta especial atención a la representación interna de los elementos Hace una clara diferencia entre

imagen y reputación, buscando un equilibrio entre la autonomía de los agentes y su necesidad de adaptarse al entorno social. Por un lado, los agentes son autónomos si han seleccionados socios en base a sus evaluaciones sociales (imágenes). Por otro lado, ellos necesitan actualizar dichas evaluaciones teniendo en cuenta a otros agentes. Esas evaluaciones sociales deben circular como reportes (reputación), con el fin de que los agentes puedan decidir si las aceptan o no. En conclusión, tiene en cuenta la experiencia directa de los demás agentes y demuestran cómo ésta puede prevalecer frente a informaciones contradictorias. Un modelo de reputación distribuido es el modelo FIRE. Este sistema multi-agente está compuesto por agentes proveedores y agentes consumidores. Cada consumidor tiene un radio de operación, es por ello que para llegar a un proveedor, necesita hacerlo a través de la opinión de los agentes consumidores más cercanos. Sin embargo, considera el aspecto temporal de las interacciones a la hora de realizar el cálculo de la reputación, pero no la posición del usuario. Se definen vecindarios de agentes usuarios a los cuales el usuario que tiene interés puede consultarles.

De la comparación entre las propuestas de FIRE vs Sporas, surge que FIRE reduce el ruido de Sporas dando un peso mayor a la experiencia directa del agente. Por su parte, FIRE puede aprender acerca de los proveedores más rápido que Sporas tan solo en las primeras interacciones (hasta que se forman los conjuntos *NoTrustValue* y *HasTrueValue*).

El trabajo propuesto por (Bertocco and Ferrari, 2008) (y que más se adapta a SMA para dominios heterogéneos) define una arquitectura centralizada para el manejo de las relaciones de confianza en sistemas multi-agente. Los conceptos que se manejan son reputación y honestidad. La reputación está relacionada estrictamente al contexto mientras que la honestidad es un concepto relacionado a un dominio específico. Este último se mide en base al último valor de la opinión de un agente sobre otro, y el promedio de los valores positivos y negativos enviados en el tiempo por los usuarios que dan sus opiniones. Se toma la ontología de este trabajo como base para la ontología que proponemos, puesto que el valor de *SocialTrustworthiness* se considera muy interesante, como así también los predicados propuestos. Este trabajo pese a ser para entornos abiertos, limita la arquitectura a un entorno centralizado con un agente encargado de gestionar y evaluar las opiniones, cosa que queremos evitar, dando más peso a interacciones limpias entre agentes usuarios. Un agente le pide al administrador que le envíe las opiniones de los otros agentes.

El sistema de reputación TOMS (Boukerche and Ren, 2008) es un sistema destinado a entornos de computación ubicua en el cual los agentes o nodos se clasifican en honestos y maliciosos. Para evaluar a un agente proponen dos funciones, si el agente es honesto, una exponencial; si es malicioso, lo evalúan con una logarítmica. A los grupos le llaman comunidades. Un agente puede ser malicioso en una comunidad y en otra no. El valor de *trustworthy* es el tiempo que está un agente en una comunidad. Cuando la reputación de un agente recae en valores menores a 0, van teniendo menos posibilidades de interactuar, tienden a quedar excluidos. Cuando un agente malicioso tiene un valor muy bajo, pasa a una lista negra para esa comunidad. El sistema de reputación es distribuido, no tienen un agente que supervise el modelo de confianza. Por lo tanto va encontrando la mejor ruta entre vecinos, sin incluir información de localización en sus cálculos. Tienen una política de confianza entre los nodos vecinos que la denominan *Trust Assistant Policy* (TAP).

En Travos (Patel, 2007) se modela el entorno como un sistema multi-agentes A con n agentes denotado como $A = a1, a2, ..., an$. En cada instante de tiempo un par de agentes (ax, ay) interactúan. Una interacción entre $a1$ y $a2$ finalizada por $a1$ si $a2$ satisfizo sus necesidades.

La reputación se calcula usando la teoría de probabilidades teniendo en cuenta las interacciones pasadas entre los agentes, y cuando este historial no existe, el modelo estima una reputación en base a la experiencia de terceros. TRAVOS asume que el comportamiento de los agentes no se modifica en el tiempo. Es un modelo poco representativo para los escenarios de inteligencia ambiental, puesto que no le da mayor relevancia a la información contextual o de localización, sino que se concentra en las interacciones mantenidas.

Un modelo de reputación distribuido es el modelo FIRE (Hunyh, 2006). El sistema multi-agente está compuesto por agentes proveedores y agentes consumidores. Cada consumidor tiene un radio de operación, es por ello que necesita llegar a un proveedor a través de la opinión de los agentes más cercanos. Considera el tiempo transcurrido desde que sucedió la interacción del agente hasta el momento de enviar la recomendación. Recolecta información contextual a través de: a) la experiencia directa del usuario que solicita recomendaciones, b) de aquellos vecinos que han interactuado con el agente de interés, c) haciendo inferencias sobre si ese agente pertenece a una compañía importante o si esa compañía cumple con ciertas normas, d) e información respecto de referencias provistas por el mismo agente con el que se desea interactuar. El peso de la reputación está dado por la frescura de las interacciones.

Las propiedades espacio-temporales de los fenómeno, así como los agentes, son especialmente importantes para calcular los valores de trust que se pueden utilizar para evaluar la calidad de la información geoespacial en un entorno de generación de contenidos colaborativo (Bishr and Mantelas, 2008). En el trabajo de Bishr y Mantelas se suponen que los usuarios contribuyen con información de manera similar a las aplicaciones colaborativas de la web 2.0. Proponen una formulación del modelo tomando en cuenta el contexto espacial y las contribuciones de los usuarios, en un escenario de una reciente urbanización y sus residentes. La reputación final se calcula mediante la relación entre la información proporcionada por los usuarios y la posición en el momento de evaluarlos, haciendo la sumatoria de los valores de rating (evaluación directa) sobre el logaritmo de la distancia. En esta publicación se propone una medida denominada "el grado de un nodo". Se observa el uso del logaritmo en base 10 como penalización de la función puesto que llega un punto en que esa distancia se hace insignificante de grande entre los agentes que interactúan. La reputación final se calcula mediante la relación entre la información proporcionada por los usuarios y la posición en el momento de evaluarlos, haciendo la sumatoria de los valores de rating (evaluación directa) sobre el logaritmo de la distancia.

El problema es que ninguno de estos sistemas de reputación incorporan en su cálculo los tres conceptos interesantes para la AmI: información de contexto, hace cuánto tiempo las entidades han interactuado ni información de localización.

TRAVOS empieza a incluir información respecto de interacciones pasadas, pero no considera el factor temporal. FIRE, si bien incluye el factor tiempo (frescura de la interacción) en su cálculo, no considera aspectos espaciales. Y, finalmente, (Bishr and Mantelas, 2008) incorpora el concepto de distancia (localización) para computar la reputación, pero descarta el factor temporal.

2.4 Conclusiones del capítulo

Tal como se mencionó anteriormente, han surgido diversos sistemas multi-agente para suplir las necesidades estructurales de los sistemas basados en contexto y de las aplicaciones de

inteligencia ambiental. El principal aspecto que resalta del estudio de estos sistemas, es que en su mayoría son aplicaciones enfocadas a resolver problemas para un dominio en particular, y muy poco de ellos son genéricos. Dentro de estos últimos, las arquitecturas existentes poseen una orientación muy marcada a la adquisición de información contextual por medio de sensores, cámaras de video y sistemas de localización. Los menos son los que se centran en las necesidades o preocupaciones del usuario del sistema en sí, tal como sobresale en la sección "arquitecturas de sistemas de agentes context aware". La mayoría de estos sistemas contemplan la creación de agentes que ofrecen servicios al usuario, sin interés en la personalización de dichas ofertas. Si bien el consumo de servicios se realiza por lo general gracias al razonamiento espacio-temporal previo de un agente, la información espacio-temporal no se utiliza para otros fines. Pues, en base a la posición del usuario, los trabajos revisados indican que un proveedor puede hacer sus ofertas o un agente administrador puede sugerir la formación de grupos o facilitar el encuentro con los contactos del usuario. No obstante, a modo de ejemplo, no se sugiere la inclusión de la información espacio-temporal en los procesos de toma de decisión, como el cálculo de la reputación de un agente usuario.

Ahora bien, en cuanto a la metodología de desarrollo de un sistema multi-agente, se espera obtener un buen diseño que lleve directamente a la implementación de los agentes involucrados. De la revisión de las metodologías existentes es sobresaliente la carencia de una metodología que parta de un alto nivel de abstracción y finalice en la programación de la estructura interna del agente, sin omitir, además, el proceso comunicativo en la organización de agentes. Por ello es necesario, obtener de las metodologías más destacadas y usadas, aquellos modelos que permitan cubrir de forma completa el análisis y diseño del sistema multi-agente, facilitando su desarrollo.

Puesto que, de la revisión del estado del arte, queda bastante por definir en cuanto al perfil del usuario del sistema, se estudiará cómo realizar la adquisición del perfil del usuario, de forma no intrusiva; cómo utilizar ese perfil para contestar a las solicitudes de reputación de los agentes; y de qué manera sería posible gestionar los intereses comunes de los perfiles de usuario para la formación de coaliciones. Ninguna de las arquitecturas genéricas estudiadas en esta sección incluyen, de forma conjunta, los aspectos mencionados anteriormente (localización, adquisición de perfiles, reputación, formación de coaliciones).

En cuanto a la comunicación de los agentes, los diversos trabajos sobre ontologías cubren aspectos parciales de las actividades que se preveen para los agentes. Existen ontologías genéricas, propias de AmI, para el manejo de la negociación entre los agentes y ontologías dedicadas particularmente a la manipulación de información de reputación o de perfiles de usuarios. Por lo tanto, nos vamos a centrar también en la búsqueda de una estructura de información que permita el intercambio de mensajes entre los agentes para este tipo de actividades, reutilizando ontologías estudiadas en el estado del arte.

Las técnicas de adquisición de perfiles de usuario, si bien no parecen presentar inconvenientes en cuanto a la implementación, no están asociadas a una planificación de tareas del usuario en el contexto en donde se encuentre. Es por ello que se trata de presentar un nueva técnica de adquisición de perfiles, que además de proveer de los gustos de los usuarios por cada tarea que tenga para ejecutar en el sistema, brinde la posibilidad de obtener una lista de tareas ordenada y priorizada según las preferencias, sin la necesidad de contar con un algoritmo de planificación.

Si bien existen variados sistemas y arquitecturas para entornos de inteligencia ambiental, trataremos de construir una arquitectura multi-agente aplicable a aquellos dominios en los que

se cuente con un conjunto de personas, usuarios del sistema, con necesidad de información contextual, como así también con un conjunto de proveedores dispuestos a suplir dichas necesidades, en base a las preferencias del usuario. Limitaremos el sistema a los aspectos relacionados al uso de software inteligente tanto para la provisión como para la adquisición de servicios e información. El sistema no abarcará aspectos relacionados a hardware, como el uso de sensores o actuadores, y especificaciones de los sistemas de localización.

A modo de conclusión de este capítulo, mencionar entonces que las investigaciones en el dominio de la inteligencia ambiental (AmI) (o computación ubicua) y los sistemas basados en contexto son numerosas actualmente, dado el auge que vienen teniendo estas ramas de la informática, y el uso de dispositivos móviles con conexión a internet en el mercado.

En este capítulo de la tesis, se ha elaborado una revisión del estado del arte para ambos temas, describiendo dos escenarios propuestos por la Information Society Technologies Advisory Group, para representar a qué apuntan estas áreas temáticas futuristas. Se analizaron los trabajos existentes sobre agentes dado que en los primeros pasos de la investigación se optó por esta tecnología para dar soporte a la AmI, dada la naturaleza pro-activa de los agentes. Se estudiaron los agentes inteligentes, su forma de comunicarse, arquitectura interna, el modelo deliberativo BDI y su vida social dentro de sistemas multi-agentes. Además se recogieron trabajos de sistemas basados en contexto y de inteligencia ambiental con aplicación de SMA. En particular, se resumieron las metodologías existentes a los fines de desarrollar un nuevo SMA y se estudiaron casos de negociación entre agentes. Se hizo una recopilación de ejemplos de SMA en AmI detectando en qué dominios son utilizados, con el objetivo de que el SMA ha desarrollar cubra aspectos de todos ellos, convirtiéndose en un SMA para dominios heterogéneos. Dada la necesidad de comunicación entre los agentes, se estudiaron ontologías que modelan un vocabulario permitiendo la comunicación en sistemas basados en contexto. Como último aspecto de estudio, se revisaron los trabajos publicados sobre la representación del usuario, esto es, ontologías para la representación del perfil de usuario, algoritmos genéticos y otras técnicas de inteligencia computacional para el aprendizaje automático de perfiles y, por último, los sistemas de reputación en el que un usuario basa su confianza para relacionarse con otros.

3

Planteamiento del Problema

FRENTE a un mundo que avanza tecnológicamente a pasos agigantados, sumándole a ello la gran ambición del ser humano por ir más allá de todo, nos sumergimos en un contexto hambriento de innovaciones tecnológicas, que requieren fehacientemente de la inteligencia artificial. Podría decirse que finalmente la ciencia ficción ha llegado a los hogares. Junto con las nuevas tendencias marcadas por la domótica, los aportes de las ciencias de la computación permiten diseñar o pensar entornos inteligentes en cualquier ámbito de la vida. La provisión de servicios en estos entornos, es el tema que se aborda en el presente trabajo. Los entornos inteligentes requieren de diferentes herramientas, para en conjunto, satisfacer las necesidades, no solo de la persona, del individuo, sino de la sociedad entera.

Un sistema adaptativo, para los nuevos entornos inteligentes, precisa de programas ejecutándose constantemente en tiempo real, con continua conexión a internet, captando todos los movimientos de las personas involucradas en el entorno, a los fines de aprender su comportamiento. Este comportamiento sirve para poder extraer cuáles son las necesidades del individuo, facilitando la provisión de servicios. Planteado este escenario, se puede resumir los problemas a abordar en:

- la necesidad de construir un sistema que se ejecute en tiempo real, que sea multiplataforma;
- la visión de usuarios conectados segundo a segundo;
- es necesario que el sistema propuesto para la provisión de servicios identifique al usuario cada vez que lo detecta;
- el problema de la extracción de las características de la persona a los fines de personalizar los servicios ofertados;
- que el sistema que se desarrolle conjugue tanto las necesidades de los usuarios como así también la oferta de los proveedores;
- que se incorpore un modelo de reputación que permita decidir al usuario el grado de confianza sobre el proveedor;
- y, finalmente, que la estructura que de soporte a los items anteriores, considere, con especial relevancia, las particularidades del entorno/contexto en el que conviven las

entidades involucradas, como asi también, que sea capaz de ubicar geográficamente cada entidad, y utilizar esta información para razonar acerca de los servicios ofrecidos.

Es necesario además, contemplar el hecho que estos entornos inteligentes tienen como motivación facilitar y dar soporte a las tareas del usuario, por lo tanto, se tratará de minimizar la interacción humano-máquina, procurando establecer acciones automáticas, y en lo posible ir gradualmente reemplazando las tareas de los seres humanos por tareas realizables por una entidad de software, altamente configurable y adaptable.

Dando solución al problema de que el sistema requerido se ejecute en tiempo real, se opta por la implementación de sistemas multi-agentes, ampliamente conocidos y aceptados por la comunidad científica (Corchado et al., 2008).

Respecto de los agentes inteligentes y su comportamiento social, existen dos corrientes de investigación sobre sistemas multi-agentes que se diferencia entre si, a nivel arquitectura (Sridharan, 1986). Una de ellas, afirma que la mejor arquitectura es aquella que tiene muy pocos tipos de agentes, cada uno con muchos comportamientos, y la otra defiende aquel esquema en donde la arquitectura del sistema multi-agentes posee varios tipos de agentes pero cada uno con pocos comportamientos. Las ventajas y desventajas de uno u otro se pueden evaluar pensando en el tiempo que lleva la implementación, cuán eficaz es la comunicación entre los agentes, y el mantenimiento que requieren a largo plazo. La arquitectura presentada en esta tesis, intenta hacer una propuesta equilibrada de ambas corrientes, enfatizando en especial la construcción del agente usuario, también denominado agentes personal en otros trabajos, dado que reemplazará en muchas actividades al usuario propiamente dicho.

La necesidad de un sistema multi-agente para entornos inteligentes se sustenta en que la mayoría de los trabajos actuales consideran el rol del agente usuario dependiente del agente administrador, también denominado agente central o de gestión de servicios, como se revisa en las secciones 2.2.3 y 2.2.2.1. En esta tesis se observa la necesidad de independizar al agente usuario para llevar adelante cierta tareas, como la negociación. La mayoría de las propuestas que se encuentran en el ámbito científico son para un entorno en particular. En esta tesis se aborda la necesidad de presentar una arquitectura genérica, adaptable a cualquier dominio, apuntando a convertirse en un framework para el desarrollo de sistemas multi-agente que se adapte a cualquier contexto que lo requiera.

3.1 Escenarios que presentan la Problemática

La Inteligencia Ambiental (AmI) es una nueva generación de entornos de computación centrados en el usuario, destinados a obtener mayores beneficios de las tecnologías de la información, utilizando los dispositivos móviles con los que cuentan a diario en todas sus actividades. Estos entornos inteligentes cuentan además con la presencia de sensores y actuadores en el ambiente, de los cuales el usuario desconoce su presencia, que se comunican con sus dispositivos móviles de manera transparente, a los fines de beneficiar y asistir al usuario. Es así como por ejemplo, en un escenario de la vida cotidiana, asumimos la existencia un anciano que vive solo en su hogar. En este hogar inteligente, un dispositivo como el refrigerador puede aprender sobre los alimentos que consume de acuerdo a una dieta saludable, y si existen faltantes, el mismo refrigerador se contacta con el supermercado para hacer el pedido. Las aplicaciones de la AmI

han surgido, y crecen continuamente, para diversos escenarios: entornos sociales, educativos, culturales, turísticos, cuidado de la salud.

La presencia de un turista en la ciudad permitirá darle asistencia durante su estadía, según sus gustos personales, indicándole dónde se puede alojar, comer o divertir; optimizando sus tiempos, costos o cualquier otro factor que le sea relevante personalmente. Además podrá pedir recomendaciones a otros visitantes de la ciudad sobre si asistir a museos, excursiones, actividades culturales, etc.. En un escenario de la vida cotidiana en la ciudad, las personas pueden saber sobre la presencia de sus conocidos en un bar, en la plaza; pueden coordinar encuentros o quedar para jugar un partido de tenis o ir a ver una película al cine. Asimismo, podrían solicitar recomendaciones a otras personas con preferencias similares sobre la película que eligen para ver. También podrían hacer recomendaciones sobre sitios que estén rebajando sus precios y advertir cuán buenos son los proveedores, etc. En un hospital, el uso de la AmI brindaría asistencia a médicos y enfermeros indicando la situación de salud de los pacientes, e informando respecto de las habitaciones en donde se encuentran, enviando alarmas a sus móviles cuando un paciente necesita atención inmediata, etc.

En estos escenarios, las computadoras monitorean nuestras actividades para predecir que queremos hacer en un contexto particular. En este sentido nos enfrentamos a la problemática de captar el contexto y la ubicación de las personas, sus preferencias, sus contactos, para comprender cuáles son sus deseos o intenciones en el entorno, y poder ofrecerle servicios de forma transparente, que satisfagan sus necesidades.

Dichos entornos están compuestos por usuarios-clientes de servicios o productos, y proveedores de los mismos. Además existen sistemas que colaboran en el descubrimiento del contexto: sistemas de localización que a través de la posición física del usuario permiten conocer en qué lugar se encuentra e inducir la situación contextual del usuario, usando un sistema razonador de contexto. Por ejemplo, una coordenada (x,y) representa la posición de un lector en la biblioteca, y además puede representar que el lector se encuentra en el 2º piso a mano derecha frente al stand de economía-política, indicándonos entonces un interés del usuario.

3.2 Una Arquitectura Multi-Agente en un Sistema Contextual Robusto

Este trabajo nace a partir de la arquitectura propuesta por (Fuentes et al., 2006a), colega del grupo de investigación del tesista. En dicho trabajo se considera una arquitectura genérica con tres tipos de agentes: central, cliente y proveedor. El agente central es el encargado de gestionar la información de contexto y de localización, como así también es el encargado de ofrecer servicios a los agentes a partir de sus perfiles como consecuencia de las ofertas que registran los agentes proveedores. En un principio, se trabajó sobre esta arquitectura, pero a raíz de la posibilidad de llevar esta arquitectura a un entorno en donde la cantidad de clientes y proveedores sea elevada, podría llevar a un cuello de botella importante en situaciones pico para el agente central. Es por ello que se estudió la necesidad de descomponer y distribuir sus tareas en otros agentes especializados, como se propone en el capítulo 4 de esta tesis, donde se plantea una extensión de la arquitectura multi-agente para la gestión de la información de contexto y localización.

Asimismo, luego del estudio de diferentes metodologías para el desarrollo de agentes se realizó una aproximación metodológica entre dos de las principales metodologías en la rama de agentes: GAIA y AUML. Incorporando conceptos de cada una, se logró realizar un completo diseño del sistema multi-agente, apuntando a su veloz desarrollo e implementación. Revisando el capítulo 2 el lector encontrará un breve resumen de las metodologías estudiadas antes de optar por las mencionadas.

Luego de un profundo análisis, se optó por el uso de GAIA, dado que es una metodología completamente orientada al desarrollo de agentes. Pero puesto a que esta metodología no lleva directamente a la implementación del sistema, además de que pierde conexión entre los distintos diagramas, se realizó una hibridación con AUML. En el trabajo expuesto por Huhns en (Huhns, 2004) se explica los pasos de AUML. En el diseño del presente capítulo se utiliza los modelos de agente y de interacción propuestos por AUML, y los modelos de roles y de servicios provistos por la metodología GAIA. El modelo de agentes de AUML resume y relaciona todos los modelos usados en esta metodología, mientras que el modelo de interacción (con el soporte de GAIA) permite llegar a la implementación del sistema de una manera mucho más fácil y directa.

AUML permite modelar agentes tipo BDI (Rao and Georgeff, 1991) y la arquitectura que se propone tiene características de BDI y de FIPA compliant (Mas, 2004). Al desarrollarla en JADE, que no implementa estrictamente la arquitectura BDI sino una aproximación, el diseño general es BDI aunque la implementación esté limitada a las posibilidades que otorga JADE (Bellifemine et al., 2007). Los agentes que conforman la arquitectura no pueden llevar adelante una acción sin un razonamiento previo sobre la localización del usuario y el contexto en el que los agentes interactúan. Esta característica es la que define que la arquitectura propuesta no pueda calificar como arquitectura híbrida.

3.3 Una propuesta de Gestión de Perfiles de Usuario basada en Algoritmos Genéticos

Un problema adicional al que nos enfrentamos al diseñar el agente usuario es de qué forma construir el perfil de usuario de manera no intrusiva y transparente. Normalmente, las personas son reacias a dar información personal, pues, mayormente, es visto como una invasión a la privacidad. Tal como se estudió en el capitulo 2 en la sección 2.3.1, existen diferentes técnicas para la generación de perfiles. Si se considera la técnica de encuestas o cuestionarios, con el simple hecho de consultarle a las personas a nuestro alrededor, se puede concluir en que el hecho de pedirles información solicitando la completitud de formularios, lo ven tedioso o como una pérdida de tiempo. Es por eso que se piensa en una técnica para extraer las preferencias de los usuarios de manera implícita. Dado que la arquitectura propuesta, permite tener un histórico del comportamiento del usuario en su agente, este agente puede extraer las preferencias del usuario sobre las características en cada tarea que realiza. A su vez, otro agente puede incorporar algún algoritmo que lea esas preferencias y construya el perfil del usuario, a los fines de proveerle servicios personalizados. En esta línea de trabajo (Chen et al., 2004c) propone la adquisición de perfiles con técnicas de algoritmos genéticos. Teniendo como base esta propuesta, se elabora en el capítulo 6 de esta tesis una propuesta innovadora que consiste en un algoritmo genético que además de extraer las preferencias sobre las características de cada tarea, genere una lista

de tareas a ejecutar optimizando los intereses del usuario. Otro trabajo sobre adquisición de perfiles de usuario utilizando técnicas de algoritmos genéticos es el expuesto por Marghny en (Marghny, 2006). El autor presenta un sistema adaptativo para aprender el perfil del usuario y su cambio de intereses en el tiempo en buscadores web. El perfil de usuario se aprende de forma incremental y continua de acuerdo al perfil inicial del usuario, sus acciones y de la interpretación semántica de la consulta para la búsqueda. Mediante el empleo de un algoritmo genético se adaptan los intereses del usuario. El sistema acepta la retroalimentación del usuario para la evaluación del fitness. La función de fitness se realiza sobre el perfil inicial de usuario.

3.4 Un Modelo de Confianza adaptado a un Sistema Contextual

Por último, dada la impronta de las relaciones humanas que pueden surgir a un nivel de agentes, sin la interacción del usuario, es necesario que el sistema multi-agente incorpore un modelo de confianza y reputación que asista al agente usuario en la toma de decisiones.

Las solicitudes de reputación tienen lugar cuando un agente solicitante no tiene confianza en un agente en particular (llamado de aquí en adelante "agente objetivo"). En este caso, el agente solicitante pide recomendaciones a sus pares (denominados de aquí en adelante agentes recomendadores) sobre el agente objetivo.

Se espera que las recomendaciones se produzcan a partir de las interacciones directas pasadas con el agente objetivo. Estas recomendaciones tratan de evitar decepciones futuras con cualquier agente objetivo. Por lo tanto, en general el sistema de reputación define:

- cómo el agente recomendador calcula las recomendaciones, a través de una ecuación que incorpora el resultado de las interacciones directas pasadas con el agente objetivo considerando propiedades temporales (la frescura relativa de cada interacción). Un sistema de reputación de ejemplo que cuida la frescura de la información es el modelo FIRE (Hunyh, 2006).
- cómo se calcula la reputación del agente objetivo consultando a otros agentes, a través de una ecuación que incluye las recomendaciones recibidas considerando la reputación del los agentes recomendadores.
- cómo se computa la reputación del recomendador de acuerdo al resultado final de la interacción actual del agente solicitante con el agente objetivo.

En los servicios basados en contexto tiene lugar un caso particular. En este entorno, las posiciones más cercanas producen interacciones más exitosas debido a que se incrementa la habilidad de discernir la calidad de los servicios. Ejemplos de dominios basados en contexto pueden ser agentes recomendadores de usuarios que presenciaron espectáculos en vivo como ser funciones teatrales y operas, visitas a monumentos y que han ejecutado tareas de vigilancia y de reconocimiento. En este tipo de dominios, la información espacial debe jugar un rol similar a la información temporal. Generalmente, las investigaciones de otros autores asumen que las interacciones que tuvieron lugar hace un tiempo atrás, tienen menos influencia en el cálculo de la reputación y por lo tanto, es necesario establecer una relación entre los valores de reputación y el tiempo que ha pasado desde el momento que tuvo lugar la interacción (esto es *la frescura*).

En una línea análoga, las interacciones producidas más lejos pueden tener menos influencia en el cálculo de la reputación y este factor lleva a pensar en un nuevo tipo de sistema de reputación que integre el valor que representa la distancia entre el solicitante y el agente objetivo dentro de la evaluación de la interacción. Tal como se describió en el capítulo 2, el trabajo de (Bishr and Mantelas, 2008), que propone un sistema de reputación basado en información geográfica colaborativa, donde se reconoce la relevancia de la información espacial y la incluyen como un logaritmo entre las distancias de los agentes, es un precedente al trabajo desarrollado en el capítulo 7.

La Fig. 3.1 muestra los distintos componentes que se describen en la segunda parte de esta tesis, y que conforman la solución propuesta, descripta en grandes rasgos anteriormente.

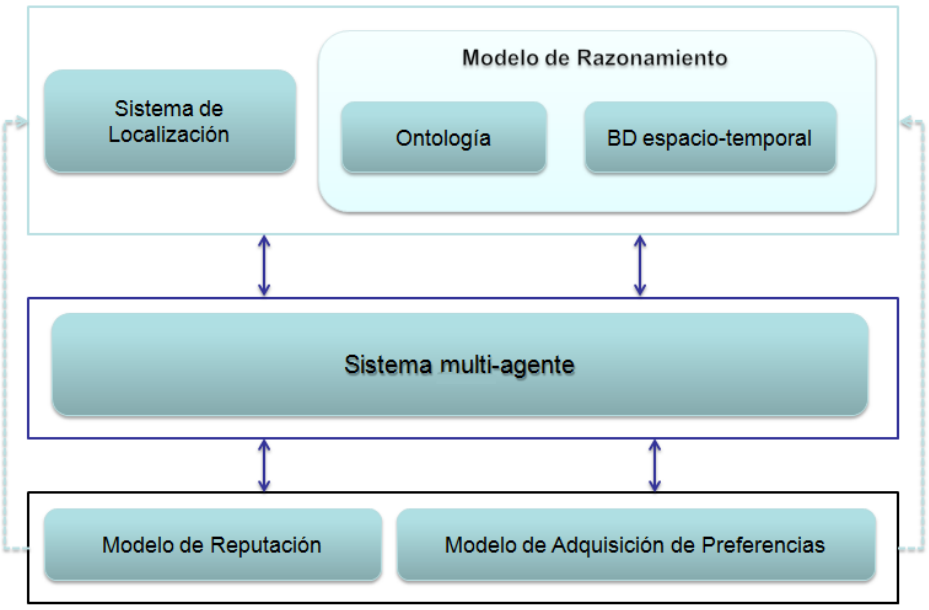

Figura 3.1: Estructura de la solución propuesta. Se muestran los módulos que componen la arquitectura diseñada y la interacción entre los mismos.

II

Propuestas y Experimentos

4

Diseño Metodológico y Evaluación Comparativa de un SMA en AmI

Los beneficios de la AmI se pueden observar en distintos contexto: sociales (Kjeldskov and Paay., 2005), educativos (Bravo et al., 2006), en el turismo (Paganelli et al., 2006), en la medicina (Corchado et al., 2008), etc.

La AmI trabaja en conjunto con los sistemas basados en contexto. Los sistemas basados en contexto deben proveer una serie de servicios de acuerdo a las preferencias del usuario, teniendo en cuenta cual es la posición actual del mismo y que actividad está realizando. Existen distintas plataformas de este tipo de sistemas tal como se expone en las plataformas contextuales Hydrogen (Hofer et al., 2003), Cortex (Sorense et al., 2004) y Citron (Yamabe et al., 2005), y como se resumió en el capítulo 2.

Para dar soporte a los sistemas basados en contexto, se han creado diferentes arquitecturas multi-agentes. Dichas arquitecturas se clasifican, según el dominio que cubren, en: arquitecturas para dominios específicos y arquitecturas para dominios genéricos. Algunos ejemplos para el primer caso son: ALZ-MAS, un SMA para el cuidado de la salud (Corchado et al., 2008); ASK-IT (Spanoudakis and Moraitis, 2006), un sistema que proporciona un agente de viajes personal para asistir al viajante durante sus actividades de recreación. Un ejemplo de arquitectura para dominios genéricos es la arquitectura CONSORTS (Nakashima, 2007). Esta es una arquitectura ad hoc para AmI. Otro ejemplo del mismo tipo es el trabajo presentado en (Fuentes et al., 2007), siendo una arquitectura que centraliza todas las tareas en un agente intermediario.

El problema abordado, es que muy pocas de las arquitecturas estudiadas han sido evaluadas dado que en el momento en que se presentaron, no existían métodos específicos para hacerlo. Con lo cual, la tendencia actual es la presentación de métodos para evaluar los SMA. El tesista propone una nueva arquitectura evaluada con distintos métodos descriptos en los trabajos (Davidsson et al., 2006), (Joumaa et al., 2008).

El método descrito en (Joumaa et al., 2008), analiza el sistema de acuerdo a la comunicación entre agentes, por ejemplo: se cuenta la cantidad de mensajes enviados en un proceso en particular. Luego se asigna un valor de acuerdo a la importancia que tiene el tipo de mensaje para el agente. Por su parte, el trabajo propuesto en (Davidsson et al., 2006) revisa las arquitecturas multi-agente desde el punto de vista del tipo de sistema y sus características, por ejemplo: si es centralizado o distribuido. Este último trabajo presenta una tabla de valores sobre

cada característica (robustez, escalabilidad, reactividad, etc.) respecto del sistema (jerárquico, centralizado o distribuido).

Ahora bien, para poder construir una arquitectura multi-agente, es necesario optar por una metodología de desarrollo que permita el diseño en detalle de todos los agentes que la componen, de los roles que puede asumir cada agente y los servicios que brinda, como así también que permita diseñar aspectos comunicacionales entre ellos. Es por ello que en este capítulo se realiza una aproximación metodológia que resulta en una nueva arquitectura multi-agente para AmI basada en contexto y locaclización, la cual es formalmente evaluada con los métodos descritos anteriormente.

En esta sección se presenta una arquitectura multi-agente para la provisión de servicios en entornos inteligentes. Primero, se realiza el análisis del sistema proponiendo un entorno de conferencias como escenario de estudio. Luego, se realiza un diseño metodológico del sistema mediante una metodología híbrida, compuesta por técnicas de GAIA y AUML. Como resultado de esto, se presenta en la sección siguiente la arquitectura propuesta con la descripción en detalle de los agentes que la componen y las interacciones entre ellos. Finalmente, se evalúa la arquitectura propuesta.

4.1 Definición del Problema

Se considera un escenario genérico basado en contexto, publicado en (Venturini et al., 2010), que involucra la satisfacción de requerimientos de los usuario como así también, a los proveedores de servicio que pueden satisfacer esas necesidades. Los proveedores intentan ofrecer sus servicios de acuerdo a las preferencias de los usuarios. En este escenario genérico, los servicios basados en localización son especialmente importantes dado que los servicios son provistos en relación al contexto del usuario. Por ejemplo, si nos situamos en el dominio de una feria de tecnología, un proveedor puede ser un stand de Nokia y los servicios ofrecidos pueden ser: dar información de teléfonos móviles, arreglar teléfonos móviles o vender accesorios.

Una actividad común que se observa en este escenario, es que continuamente los usuarios interactúan con sus pares. En el modelo de agentes que se presenta en esta tesis, la influencia de terceros sobre la decisión del usuario tienen gran relevancia, por ejemplo, a través de las recomendaciones. De hecho, las recomendaciones juegan un rol principal a los fines de obtener servicios personalizados de acuerdo a la situación actual del usuario. Por ejemplo, los usuarios pueden desear obtener recomendaciones personales de los lugares por donde se encuentran andando (sitios históricos, centros comerciales, etc.). Esta personalización es posible gracias a técnicas de adquisición de perfiles. En este sentido, en la sección 6 se trata de obtener perfiles de usuarios, de forma no intrusiva, mediante la observación externa de las acciones del usuario en el sistema. A esto es a lo que llamamos perfil de usuario público en contraposición al perfil privado del usuario utilizado por el usuario de forma interna para tomar decisiones. Además, se considera la posibilidad de abordar el tema de la formación de coaliciones de los usuarios cuando tienen intereses compartidos. Varios usuarios podrían ejecutar acciones en conjunto o compartir información dentro de la coalición.

Resumiendo, para un escenario genérico dado, se tratarán de modelar los siguientes conceptos: un servicio de localización, recomendaciones usuario-a-usuario, adquisición de perfiles públicos de usuario de manera no intrusiva y la formación de coaliciones. En esta sección se presenta

el diseño y la construcción de una arquitectura que incluye las características mencionadas anteriormente, utilizadas en la tecnología de agentes.

Para ilustrar esta situación, se optó por un dominio turístico, considerado uno de los escenarios más ricos en cuanto a información contextual. Suponemos entonces unas vacaciones en la ciudad de Salta (ver Fig. 4.1), Argentina, ubicada a 1200 mts. sobre el nivel del mar. Salta se ubica en la provincia del mismo nombre, al norte del país. Está rodeada por la gran Cordillera de los Andes al oeste y por una zona selvática al este. Desde la ciudad se pueden tomar excursiones para recorrer toda la provincia y también la vecina provincia de Jujuy.

Figura 4.1: Mapa de la ciudad turística de Salta Capital, Argentina

Un usuario llamado Juan llega desde Madrid a la ciudad por vía aérea, previa escala en Buenos Aires, capital de Argentina. Luego de recoger su maleta, recibe una propuesta para rentar un coche directamente en el aeropuerto. En su móvil se muestra además un mapa con el recorrido que debería seguir para llegar hasta el hotel que ha reservado. El mismo día que gestionó su billete para las vacaciones, automáticamente una aplicación de su móvil conectada con la billetera electrónica, se encargó de hacer la reserva del hospedaje según las preferencias de Juan. Decide aceptar la propuesta. Toma el coche y se dirige hacia el hotel.

Después de acomodarse, decide salir a pasear, y camina hacia la plaza principal de la ciudad. Recibe un mensaje con una oferta para probar unas riquísimas humitas (plato típico del norte a base de maíz) en una terraza frente a la plaza. Mientras almuerza recibe otro mensaje para unirse a un grupo que visitará el museo de Alta Montaña, ubicado a unos metros de la cafetería en la que se encuentra. El grupo se formó ad-hoc, y se negociará una visita guiada a precio reducido. Juan confirma su asistencia.

Por la tarde, revisa su móvil en búsqueda de otras actividades para hacer durante su estadía. Ya no tiene la necesidad de ir hasta los puntos de información turística desplegados en la ciudad, que han sido reemplazados por pantallas RFID. En estas pantallas se muestra información, de

acuerdo a los gustos de los visitantes, almacenados en sus móviles.

Al día siguiente, decide realizar el circuito histórico por el centro de la ciudad. Una recorrida que permite descubrir museos, iglesias y casonas que fueron construidos en época de la colonia. Durante su paseo, recibe mensajes de otros visitantes que vienen de su pueblo, y que también eligieron Salta como destino turístico para sus vacaciones de invierno. A medida que pasa por cada edificio, en la pantalla de su móvil se muestra información sobre el año de construcción, una breve descripción histórica, y en los casos correspondientes, el costo de la entrada. Juan puede elegir entonces, cuál visitar por dentro.

Durante su estadía, recibe recomendaciones de otros turistas, con intereses similares a los suyos. Algunas sobre sitios para comer platos típicos, otras sobre excursiones que no debería perderse como el viaje hasta el Parque Nacional Los Cardones (con gran variedad de fauna típica de la puna: llamas, guanacos, etc) o un paseo en el Tren de las Nubes (que llega hasta los 4200 mts. de altura). Se decide por esta última, que le parece la más exótica, y automáticamente su billetera digital negocia con la agencia de turismo un precio razonable, concretando finalmente la operación.

El ayuntamiento de la ciudad cuenta con un servidor que maneja toda la información contextual: hoteles, restaurantes, museos, festivales de folklore, excursiones, circuitos turísticos, deportes extremos, movilidad. Hay un despliegue de redes WiFi y WiMax en todo el micro y macro centro de la ciudad, y en otros sectores que se convirtieron en el centro de atención de los turistas. Estas antenas permiten que los visitantes estén conectados a la red constantemente, y puedan recibir ofertas en sus móviles. Además, la Secretaría de Turismo, exige que todos los museos, iglesias y demás edificios que forman parte del circuito histórico y de arte de la ciudad, cuente con sensores de presencia, y tarjetas RFID con información detallada de lo que se expone. Esto hace posible que si una persona no toma una visita guiada, en su dispositivo móvil se muestre información sobre el edificio, escultura, obra de arte, o pieza histórica que tiene frente suyo. También, cada proveedor de servicios cuenta con un servidor con los productos y servicios negociables, que ofrece a los clientes potenciales.

La lista de actividades-tareas que puede ejecutar el usuario en un escenario como el explicado anteriormente, incluye:

- visitar el museo de Alta Montaña,
- visitar el museo de Bellas Artes,
- concurrir a la Catedral Basílica,
- hacer un city tour por la ciudad,
- ir al teatro,
- concurrir a festivales de danza y música folklórica (zambas y chacareras),
- ver a la orquesta sinfónica,
- conocer los parques nacionales: Baritú, Los Cardones, El Rey,
- compartir experiencias con los gauchos de la zona,

- comprar comidas típicas,
- comprar artesanías y souvenirs

Además, desde otra perspectiva, el usuario en sí tiene una serie de actividades innerentes al entorno en el que se mueve, y relacionadas a la convivencia con otros usuarios en el contexto:

- pedir recomendaciones sobre sitios para comer o visitar,
- responder solicitudes de recomendaciones,
- aceptar ofertas de las agencias de turismo,
- consultar el mapa de la ciudad y el de la provincia,
- contactar a otros visitantes

La lista de servicios de los proveedores son:

- ofrecer productos regionales,
- proveer de mapas del circuito histórico, de pesca en río,
- ofertar excursiones: a los Valles Calchaquíes, la ruta del vino, deportes extremos, las yungas
- informar sobre shows y festivales públicos,
- brindar información de localización,
- comunicar ofertas,
- hacer descuentos para grupos de personas,
- enviar información sobre paquetes turísticos,
- brindar información sobre horarios y estaciones de buses y trenes

4.2 Interacciones de los Agentes en el Sistema

De acuerdo a la descripción del problema de la sección 4.1, se observa que una arquitectura multi-agente podría adaptarse fácilmente al dominio del turismo. La arquitectura que se propone cuenta con cinco tipos de agentes: agente usuario (*User Agent*), agente proveedor (*Provider Agent*), agente de localización (*Locator Agent*), agente intermediario (*Broker Agente*) y agente administrador de servicios (*User Manager Agent*). Inicialmente, el agente de localización (LA) establece la posición actual del usuario en el área del aeropuerto donde se recogen las maletas. Cuando el usuario se mueve dentro del edificio, el *Locator Agent* detecta su posición en el sector cercano a las agencias que rentan coches. El *Broker Agent* advierte a los proveedores sobre un cliente potencial. El agente localizador muestra al usuario el recorrido que debe seguir

hasta su hotel sobre un mapa. El *Provider Agent* de la agencia envía una propuesta al *User Agent*. Luego, mientras que el turista camina hacia la plaza principal, el agente intermediario *Broker Agent* cruza el perfil de usuario del visitante con los proveedores que hay en el camino. Constantemente le llega información según sus gustos. Cuando se sienta en la terraza a comer, el *User Manager Agent* identifica a varios turistas, que al igual que este turista, tienen la idea de conocer el museo de Alta Montaña, y gestiona con el agente proveedor del museo la presencia de un guía disponible para la visita. El *User Manager Agent* envía un mensaje a los turistas que se encuentran cercanos al edificio para unirse al grupo que hará la visita, resolviendo una coalición. Durante la estadía del turista, el agente usuario de su dispositivo móvil se comunica con otros agentes usuarios para pedir o contestar solicitudes de recomendación. Cuando decide tomar la excursión en el Tren de las Nubes, el agente usuario se encarga de negociar con la agencia de turismo a través de su *Provider Agent*.

De la misma manera, la tarea de cada agente puede adaptarse a cualquier contexto. El uso de una ontología genérica (como se verá en el capítulo 5) facilita esta adaptación.

El sistema posee las siguientes funcionalidades: localización, provisión personalizada de servicios, manejo de coaliciones, procesos de recomendación. Por lo tanto, el agente de localización se encarga de obtener la posición geográfica de los agentes usuarios, y con ésta inferir su localización a un alto nivel para comprender en qué contexto se encuentra el usuario. Por otra parte, la posición geográfica como tal será un factor importante en el cálculo de las recomendaciones de los agentes usuarios, suponiendo que la calidad del consumo del servicio, depende de la distancia, entre otros factores. Luego el agente intermediario es quien puede advertir a los agentes proveedores sobre clientes potenciales, puesto que puede inferir las preferencias del usuario, a través de sus movimientos e interacciones, y cruzarlas con las ofertas que hacen los proveedores a través de sus agentes proveedores. Entonces los agentes proveedores podrán luego ofrecer sus servicios y negociar o hacer subastas de acuerdo a los gustos de los usuarios. En cuanto a los agentes usuarios, estos pueden por defecto negociar o aceptar acuerdos con los agentes proveedores, pueden hacer recomendaciones a otros agentes usuarios, calcular la reputación de un proveedor sobre el que están interesados según esas recomendaciones, formar grupos o negociar entre ellos. No obstante, sus funciones principales consisten en el mantenimiento de su perfil interno, el cual contiene información de las interacciones que realizan con los agentes proveedores, las preferencias de los usuarios y las recomendaciones recibidas. Finalmente, el rol del agente administrador de usuarios consistirá en dos funciones principales, mantener actualizado el perfil que el agente usuario comparte con otros agentes, y estar atento a la formación de coaliciones, negociando las mismas.

Una primera aproximación de la arquitectura construida se muestra en la siguiente Fig. 4.2

4.3 Diseño de un SMA en un Escenario de AmI

4.3.1 Paradigma de Agentes y uso de Metodologías

El sistema multi-agente basado en contexto que se trata de diseñar en esta tesis, sigue el paradigma BDI (Belief-Desire-Intention, en español Creencias-Deseos-Intenciones) de (Rao and Georgeff, 1991). Los agentes BDI se construyen de acuerdo a esos tres niveles de conocimiento

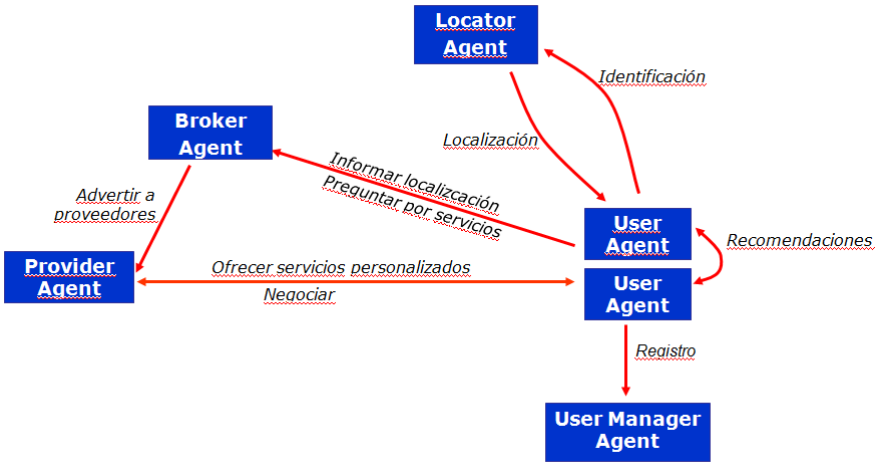

Figura 4.2: Estructura del sistema multi-agente basado en contexto y localización

inspirados en el razonamiento humano, soportando comportamientos reactivos y proactivos. Las creencias vienen dadas por el conocimiento que tiene el agente del entorno y de otros agentes, los deseos representan los objetivos que tiene cada agente, y finalmente, un conjunto de intenciones que pueden ser creadas/actualizadas/finalizadas si el entorno se modifica. El paradigma BDI provee soluciones de agentes a un entorno dinámico con un grado de incertidumbre.

AUML, como extensión de UML, permite la incorporación de BDI en el modelo de agentes. AUML facilita la implementación del sistema a partir del desarrollo de agentes. AUML es una extensión del conocido UML con las siguientes características: una clase de agentes organizada especial, el nuevo concepto de roles y el nuevo diagrama de protocolo de interacción entre agentes. En el diseño, solo se utilizarán los modelos de agente y de interacción.

Por otro lado, GAIA es una metodología específica para el diseño de sistemas multi-agentes, que trabaja en dos fases: el análisis y el diseño. La fase de análisis intenta comprender el sistema y su estructura. Los pasos que incluye son: (i) la estructura organizacional, (ii) el modelo de entorno, (iii) el modelo de roles preliminar, (iv) el modelo de interacción preliminar y (v) las reglas organizadas. En cuanto a la fase de diseño, se realiza una mejora en las especificaciones de los modelos de la fase de análisis a fines de implementar los agentes (Fuentes et al., 2007). De esta metodología se adoptarán los siguientes modelos: el modelo de roles y el modelo de servicios.

Un problema que surge al emplear la GAIA es que no considera en ningún momento una fase de requisitos, y la solución a ello es el uso de los diagramas de caso de uso de UML como se describe en el anexo A.3.

En este apartado se describe el diseño del SMA propuesto, paso a paso, considerando los posibles agentes que derivan del escenario analizado anteriormente. Como consecuencia de dicho escenario, se detecta la presencia de:

- uno o más usuarios que requieren de servicios para satisfacer sus necesidades de información,

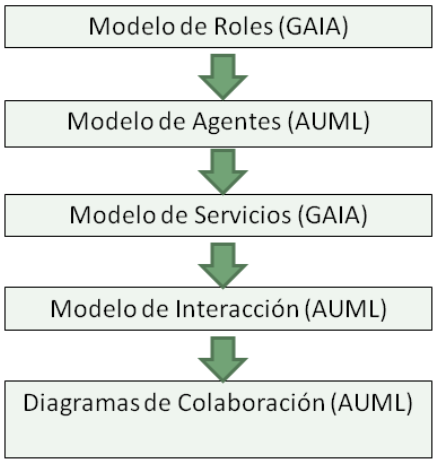

Figura 4.3: Hibridación de GAIA y AUML

- la presencia de varios proveedores de servicios, que ofertan sus servicios personalizando los mismos de acuerdo al perfil del cliente,
- uno o más agentes administradores de información contextual y de información de localización.

En la Fig. 4.3 se exponen los modelos usados de GAIA y AUML, que en las secciones siguientes se usan para diseñar la arquitectura multi-agentes propuesta.

4.3.2 Modelo de Agentes

En la metodología AUML, los agentes se definen utilizando diagramas de agentes (ver Tabla 4.1). De acuerdo al modelo de agentes propuesto en este trabajo, se consideran agentes cognitivos/deliberativos para la arquitectura. Los agentes deliberativos se diseñan teniendo en cuenta los siguientes conceptos: roles, servicios que ofrecen, descripción del protocolo de comunicación, planes, objetivos, acciones y conocimiento (creencias). Teniendo en cuenta el rol descrito en el modelo de roles, se describen las características de los agentes para luego mostrar el modelo de agentes.

Providers Agents (PA - Agentes Proveedores) ofrecen servicios a los usuarios y realizan negociaciones con ellos. Sus creencias son su propia información de contexto y la información que intercambian con el agente intermediario y con otros agentes proveedores que comparten su conocimiento. Uno de los objetivos de los PAs es ofrecer servicios de acuerdo a las preferencias del usuario, para lo cual el agente intermediario confronta el perfil del usuario con los servicios ofrecidos por los PAs. Otro objetivo es lograr acuerdos con los usuarios en cualquier proceso de negociación. Los planes de estos agentes consisten en dialogar con los usuarios para llegar a esos objetivos.

Tabla 4.1: Diagrama de Agente de AUML.

«Agente»
Roles
Diagrama de estado 1, Diagrama de estado 2
Atributo 1, Atributo 2
[precond]operación 1 [postcond]
[precond]operación 2 [postcond]
Capacidades, servicios
Percepción 1, Percepción 2
Protocolo 1: rol, Protocolo 2: rol
Organización 1: rol, Organización 2: rol

Tabla 4.2: Diagrama de agente: Agente Proveedor

Agente Proveedor
Roles - Proveedor de Servicios
Servicios - Ofrecer servicios - Solicitar negociación - Aceptar negociación - Intercambiar información
Creencias - Su propia información contextual - La información contextual del Agente Intermediario (BA) y de otros - Agentes proveedores que comparten su conocimiento
Deseos - Comunicar servicios - Lograr un compromiso con los clientes
Intenciones - Ofrecer servicios a los usuarios - Negociar con los usuarios para lograr acuerdos

Users Agents (UA - Agentes Usuarios): pueden negociar con los proveedores, realizar acuerdos, hacer recomendaciones a otros usuarios o negociar con ellos. A tales efectos, los UA, pueden requerir ciertos servicios o información. Después de esto, los UA pueden recibir múltiples ofertas de distintos proveedores y de esa manera decidir con cuál de ellos iniciar el proceso de negociación. Los UAs son responsables de la administración de los perfiles internos de cada agente usuario. Pueden mejorar los perfiles según la recepción de recomendaciones. Creen solo en su propia información de perfil (información pública y privada). Sus planes son: negociar y realizar acuerdos con los proveedores, dialogar con otros usuarios (por reputaciones, recomendaciones o para formar grupos) y gestionar el plan del perfil.

Tabla 4.3: Diagrama de agente: Agente Usuario

Agente Usuario
Roles
- Rol de Recomendador
- Rol de Negociador
- Administrador de perfil interno
Servicios
- Hacer recomendaciones a otros usuarios: Rol de Recomendación
- Solicitar recomendaciones: Rol de Recomendación
- Recibir recomendaciones: Rol de Recomendación
- Decidir si confiar: Rol de Recomendación
- Consultar Información: Rol de Negociación
- Recibir Servicios: Rol de Negociación
- Preguntar por acuerdos: Rol de Negociación
- Recibir solicitudes de negociación: Rol de Negociación
- Intercambio de información: Administración del perfil interno
Creencias
- Información del perfil de usuario (público y privado)
Deseos
- Negociación entre usuarios y proveedores
- Recomendaciones entre usuarios
- Confiar en otros agentes
- Administrar su perfil interno
Intenciones
- Plan de negociación
- Dialogo entre usuarios
- Plan para la administración de perfiles

Locator Agent (LA - Agente Localizador) juega el rol de identificación de usuarios y de localización de usuarios en el entorno. Para ello, el LA se conecta a la plataforma Appear, la cual provee servicios de localización, tal y como se hizo en (Sánchez Pi et al., 2007). Además, debe razonar sobre datos espacio-temporales. Es decir, localización implica algo más que la posición geográfica, incluye información de alto nivel, el agente de localización puede inferir a través de la posición del agente usuario cuál es su contexto. Este agente puede ser un agente de fusión de datos en caso de que el sistema lo requiera, y de esta manera conectarse a múltiples LAs que se encuentren trabajando bajo diferentes plataformas. Sus creencias son: la posición del usuario y la información espacio-temporal (ambas definidas en la ontología propuesta para la arquitectura en la sección 5). Los deseos del LA son: la localización e identificación de los agentes usuarios UAs. Para lograr estos objetivos debe contactarse a una plataforma proveedora

Tabla 4.4: Diagrama de agente: Agente Localizador

Agente de Localización
Roles - Administrador de Localizaciones
Servicios - Chequear localización de usuarios - Identificar usuarios
Creencias - Localización de los Agentes Usuarios - Datos espacio-temporales
Deseos - Localizar e identificar agentes usuarios
Intenciones - Posición del usuario a través de servicios de localización

de servicios de localización.

User Manager Agent (UMA - Agente Administrador de Usuarios) debe registrar/quitar del registro cada uno de los agentes usuarios UA que deseen incorporarse al sistema. Luego de recibir una solicitud de registro, debe analizar la situación del UA antes de registrarlo en el sistema. Por ejemplo, si el dominio de aplicación es el de conferencias de ingeniería y una enfermera quiere registrarse, esta solicitud debería ser rechazada. Otra actividad del UMA es actualizar los perfiles compartidos de los usuarios. Este agente cree en el perfil público de cada agente usuario (el cual se introduce al sistema cuando el usuario se registra). Además, es el encargado de manejar coaliciones sobre intereses comunes.

Broker Agent (BA - Agente Intermediario), es el responsable de administrar servicios y de descubrir y filtrar a los proveedores de servicios. Además, debe hacer el matching (cruzar) el perfil del usuario con los servicios ofrecidos por los PAs. El Broker Agent recibe la posición del UA y en base a las coordenadas y a la zona donde se encuentra el UA, filtra a aquellos PAs que se encuentran más cerca. Luego, el BA advierte a los agentes proveedores PA acerca de la presencia de un cliente potencial. Las creencias del BA son la información de contexto y la posición del agente proveedor de servicios. Las intensiones de este agente están relacionadas al objetivo de filtrar proveedores y aplicar la correspondencia entre el perfil del UA y los servicios de los PA.

4.3.3 Modelo de Roles

El modelo de roles se utiliza para identificar los roles principales de cada agente. Cada rol puede verse como una descripción abstracta de la función esperada de una entidad en particular. Además, cada rol tiene asociados atributos: responsabilidades (un rol se crea para 'hacer' algo) y permisos (relacionados al tipo y cantidad de recursos que puede explotar el agente cuando tiene ese rol). De acuerdo a cada modelo de agentes, se detalla el modelo de roles usando GAIA. Los roles identificados en el sistema son: Administrador de Usuarios (UM - *User Manager*), Administrador de Localización (AL - *Location Manager*), Administrador de Servicios (AS - *Services Manager*), Descubridor de Proveedores (DP - *Provider Discover*),

Tabla 4.5: Diagrama de agente: Agente Administrador de Usuarios

Agente Administrador de Usuarios
Roles - Administrador de Usuarios - Administrador de Coaliciones - Procesador de búsquedas - Administrador de Confianza
Servicios - Solicitar registro de perfiles: Administrador de Usuarios - Aceptar registro: Administrador de Usuarios - Registrar usuarios: Administrador de Usuarios - Dar de baja usuarios: Administrador de Usuarios - Recibir secuencia de usuarios: Administrador de Usuarios - Mejorar perfiles: Administrador de Usuarios - Actualizar perfil interno: Administrador de Usuarios - Enviar registro de perfiles compartidos: Administrador de Usuarios - Formar grupos de agentes usuarios: Administrador de Coaliciones - Buscar actividades/lugares para recomendar: Administrador de Confianza - Administrar mecanismos de reputación: Administrador de Confianza
Creencias - Información pública del perfil de usuario
Deseos - Registrar/Dar de Baja usuarios - Mejorar perfiles de usuarios compartidos - Formar grupos de usuarios
Intenciones - Plan de registro de usuarios - Administrar perfil de usuario - Administrar plan de formación de coaliciones - Plan de recomendaciones o acciones de reputación

Tabla 4.6: Diagrama de agente: Agente Intermediario

Agente Intermediario
Roles - Administrador de Servicios - Descubridor de Proveedores
Servicios - Correspondencia-Servicios-Perfiles: Administrador de Servicios - Filtrado de proveedores: Descubridor de Proveedores - Advertencia a proveedores: Descubridor de Proveedores - Búsqueda de información: Administrador de Servicios
Creencias - Identificación de Agentes Usuarios - Localización de Agentes Proveedores
Deseos - Detectar usuarios - Proveedores filtrados
Intenciones - Plan de advertencia a proveedores

Proveedor de Servicios (PS - *Service Provider*), Administrador de Perfiles (AP - *Profile Manager*), Rol de Negociador (RN - *Negotiate Role*), Rol de Recomendador (RR - *Recommend Role*), Administrador de Coaliciones (AC - *Coalition Manager*) y Administrador de Confianza (AC - *Trust Manager*). A continuación se describe cada uno de ellos.

Administrador de Usuarios: este rol es responsable del registro de usuarios como así también tiene la facultad de mejorar el perfil público de los usuarios.

Administrador de Localización: este rol es responsable de la identificación y localización de los usuarios.

Administrador de Servicios: este rol realiza la correspondencia (matching) entre los servicios ofrecidos y el perfil de usuario.

Descubridor de Proveedores: este rol obtiene los proveedores cercanos y les comunica acerca de la cercanía de clientes potenciales (usuarios).

Proveedor de Servicios: este rol es responsable de informar sobre los servicios y de realizar los tratos en la negociación con los usuarios.

Administrador de Perfiles: la responsabilidad de quien asume este rol es actualizar el perfil de usuario interno y ofrecer la posibilidad de enviar parte del perfil compartido al administrador de usuarios.

Rol de Negociación: este rol permite que los agentes realicen negociaciones y, de acuerdo a esto, recibir nuevos servicios o mejorarlos.

Rol de Recomendación: este rol hace posible la recomendación de información entre agentes, y a decidir en quien confiar para pedir recomendaciones.

Administrador de Coaliciones: este rol permite resolver el problema de la formación de coaliciones entre agentes usuarios.

Gestor de Confianza: este rol es responsable de manejar todo tipo de negociación entre agentes como ser: recomendar información, asistir en el proceso de reputación y manejar las relaciones de confianza.

La Tabla 4.8 contiene la representación de cada uno de los roles definidos para el SMA, en el modelo de roles de GAIA. El campo responsabilidades (*responsibilities*) posee dos propiedades *seguridad (safety)* y *bondad (liveness)* claramente explicadas en (Cernuzzi et al., 2004). La propiedad *seguridad* es una propiedad que representan al agente actuando en el rol que debe preservar. Estos se expresan como predicados sobre las variables o los recursos que puede tener. La propiedad *bondad* describe el ciclo de vida o patrón de comportamiento del rol. Se representa con una expresión regular sobre un conjunto de actividades y protocolos que ejecuta el rol. La Tabla 4.7 sintetiza los operadores que puede utilizar la expresión regular.

Tabla 4.8: Modelo de Roles de GAIA para el sistema multi-agente basado en contexto.

Diagrama de Rol: Administrador de Usuarios (User Manager - UM)
Descripción: Este rol es responsable del registro de usuarios, como así también de mejorar sus perfiles públicos
Continued on next page

Tabla 4.8 – *Continued from previous page*

Protocolos y Actividades:
recibir-registro-perfiles, aceptar-registración,registrar-usuarios, baja-de-usuario, recibir-secuencia-usuarios, mejorar-perfil-público-usuario

Permisos:
Lee el perfil_usuario, secuencia cambios registro_de_usuario, perfil_público_de_usuario

Responsabilidades
Bondad:
UM = (Recibir_registro_de_perfil_público. Aceptar_registro. (Registrar_usuario | Baja_de_usuario)) ^{n} | (Recibir_secuencia_de_usuario. Mejorar_perfil_de_usuario)^n
Seguridad: es necesario para garantizar que no existan intrusos en el sistema

Diagrama de Rol: Administrador de Localización (AL)

Descripción
Este rol es responsable de la identificación y localización de usuarios.

Protocolos y Actividades
Checkear-localización-del-usuario, identificación-de-usuario

Permisos
Lee las posiciones_de_usuarios

Responsabilidades
Bondad:
AL= (Checkear-localización-del-usuario. Identificar-usuarios)^n
Seguridad: es necesario para asegurar la conexión con el sistema de localización.

Diagrama de Rol: Administrador de Servicios (AS)

Descripción
Este rol hace una correspondencia entre los servicios ofrecidos por los proveedores y el perfil de usuario.

Protocolos y Actividades
Correspondencia-servicios-perfiles

Permisos
Lee servicios_del_proveedor, perfil_de_usuario
Cambia el resultado_de_la_correspondencia

Responsabilidades
Bondad:
AS=(Correspondencia-servicios-perfiles)^n
Seguridad: es necesario que el servicio_del_proveedor y el perfil_de_usuario estén disponibles para aplicar la correspondencia.

Diagrama de Rol: Descubridor de Proveedores (DP)

Descripción
Este rol se encarga de obtener a los proveedores cercanos y comunicarse con ellos para advertirles sobre la presencia de un usuario.

Protocolos y Actividades
Filtrar-proveedores, advertir-a-proveedores

Permisos
Lee los resultados_de_la_correspondencia
Cambia información_de_comunicación

Continued on next page

Tabla 4.8 – *Continued from previous page*

Responsabilidades
Bondad:
DP= (Filtrar-proveedores. Advertir-proveedores)^n.
Seguridad: si es positivo el resultado de la correspondencia, es posible comunicarlo con el proveedor.

Diagrama de Rol: Proveedor de Servicios (PS)

Descripción
Este rol es responsable de informar acerca de servicios y realizar acuerdos con otros usuarios después de negociar.

Protocolos y Actividades
Ofrecer-servicios, solicitar-negociaciones, aceptar-negociaciones, intercambiar-información

Permisos
Lee aceptaciones-de-negociacion, information_exchange
Modifica servicios_ofrecidos

Responsabilidades
Bondad:
PS= (Ofrecer-servicio)^n | (Solicitar-negociación. Aceptar-negociación. Intercambiar-información)^n.(Ofrecer-servicio)
Seguridad: es necesario negociar primero para lograr un acuerdo e intercambiar información.

Diagrama de Rol: Administrador de Perfil Interno(Profile Manager - PM)

Descripción
Este rol es responsable de la actualización de perfiles de usuarios internos y ofrece la posibilidad de enviar una parte al agente intermediario.

Protocolos y Actividades
Actualizar-perfiles-internos, enviar-registro-perfil-compartido

Permisos
Lee perfil_usuario, información_recomendada
Cambia perfil_usuario

Responsabilidades
Bondad:
PM= (Actualizar-perfiles-internos)^n | (enviar-registro-perfil-compartido)^n.
Seguridad:
Es necesario recibir información externa, como recomendaciones, para mejorar o actualizar el perfil interno.

Diagrama de Rol: Rol de Negociación (RN)

Descrìpción
Este rol le permite a los agentes negociar y, de acuerdo a esto, recibir servicios nuevos o mejorados.

Protocolos y Actividades
Consultar-información, recibir-información, preguntar-por-acuerdos, recibir-solicitar-negociación, cambiar-información

Permisos
Lee información-negociación
Cambia servicios

Responsabilidades
Bondad:
RN= (Consultar-información)^n | (Preguntar-por-acuerdos. Recibir-pedir-negociación.
(Intercambiar-información)^n . (Recibir-servicios)^n.
Seguridad:

Continued on next page

Tabla 4.8 – *Continued from previous page*

Existe una fase de negociación para alcanzar acuerdos y recibir nuevos servicios.

Diagrama de Rol: Rol de Recomendador (RR)

Description
Este rol ofrece la posibilidad de recomendar información entre usuarios, y decidir a quién confiar para compartir opiniones.

Protocols and Activities
Recomendar, preguntar-por-recomendaciones, decidir-si-confiar, recibir-recomendación

Permisos
Lee recomendaciones
Cambia opinión_a_otros, perfil_público

Responsabilidades
Bondad:
RR= (decidir-si-confiar, Recomendar)^n | (Perguntar-por-recomendaciones. Recibir-recomendaciones)^n.
Seguridad:
Es necesario asistir en el proceso de búsqueda de información del usuario.

Diagrama de Rol: Administrador de Coaliciones (AC)

Descripción
Este rol permite resolver el problema de formación de coaliciones entre usuarios.

Protocols and Activities
Recibir-solicitudes-de-usuario, enviar-información-compartida, formación-de-grupos

Permisos
Lee solicitudes de usuarios

Responsabilidades
Bondad:
AC=(Recibir-solicitudes-de-usuarios | enviar-información-compartida | formación-de-grupos)^n.
Safety:
Es necesario administrar las coaliciones entre usuarios.

Diagrama de Rol: Administrador de Confianza (Trust Manager - TM)

Descripción
Este rol es responsable de administrar todo tipo de negociación entre agentes del usuario como ser:
información de recomendación, valoración de la reputación, relaciones de confianza.

Protocolos y Actividades
Buscar-actividades/lugares-para-recomendar. Administrar-mecanismos-de-reputación

Permisos
Lee solicitudes de usuarios

Responsabilidades
Bondad:
TM=(Buscar-actividades/lugares-para-recomendar | Administrar-mecanismos-de-reputación)^n.
Seguridad:
Es necesario establecer relaciones de confianza entre usuarios en el sistema.

Tabla 4.7: Operadores del Modelo de Roles de GAIA

Operador	Interpretación
x.y	x seguido de y
x|y	ocurre x o y
x^w	x ocurre indefinidas veces
$[x]$	x es opcional
x||y	x e y

El perfil interno será el perfil completo del usuario gestionado por el agente usuario. El agente usuario decidirá que parte de su perfil compartir con otros agentes, y este perfil es el que se denomina perfil público, gestionado básicamente por el rol Administrador de Usuarios. El perfil que no comparte y que utiliza el agente usuario para cálculos y decisiones internas, es el que se denominará perfil privado.

4.3.4 Modelo de Servicios

El modelo de servicios en la metodología GAIA trata de identificar las funciones (servicios) asociadas con el rol de cada agente, y para especificar las propiedades principales de esos servicios (Fuentes et al., 2007). Debemos identificar entonces por cada servicio las entradas, salidas y pre y post condiciones. A continuación se detalla este modelo rol por rol.

El primer rol es el Administrador de Localización (*Locator Manager*) encargado de chequear la posición del usuario y de identificar servicios de usuarios. (ver Tabla 4.9)

Tabla 4.9: Servicios: Administrador de Localización (AL)

Diagrama de Servicios: Administrador de Localización				
Servicio	**Entradas**	**Salidas**	**Pre-Condiciones**	**Post-Condiciones**
Verificar ubicación de usuario	Posición de usuario	Posición verificada	Usuarios conectados a la red inalámbrica	Usuarios localizados
Identificar usuario	Posición de usuario verificada	Identificación	Usuario localizado	El usuario está identificado

El rol Administrador de Usuarios (*User Manager*) está relacionado a los siguientes servicios listados en la tabla correspondiente: solicitud de registro de perfiles, aceptar el registro, registrar y quitar de registro a los usuarios y mejorar los perfiles de usuario, tal como se muestra en la Tabla 4.10.

El servicio relacionado con el rol Administrador de Servicios (*Services Manager*) es hacer la correspondencia entre el perfil de usuario y los servicios ofrecidos por los distintos proveedores. (ver Tabla 4.11)

El rol Descubridor de Proveedores (*Provider Discover*) es responsable de filtrar todos los

Tabla 4.10: Servicios del Rol Administrador de Usuario usando GAIA

Diagrama de Servicio: Administrador de Usuario (UM)				
Servicio	**Entradas**	**Salidas**	**Pre-Condiciones**	**Post-Condiciones**
Solicitar de Registro de usuario	Usuario envía un mensaje de solicitud	Solicitud de registro y perfil de usuario enviados	El usuario está en la red inalámbrica	El Agente Central recibe una solicitud del usuario
Aceptar Registro	Solicitud de registro y perfil de usuario	Aceptación de mensaje y registro hechos	El usuario envía un mensaje de solicitud	Registro o no de usuario
Registrar usuario	Registro propuesto	Registro solicitado	Localizar e identificar usuarios	Usuario registrado
Baja de usuario	Baja propuesta o usuario fuera de la red inalámbrica	Baja solicitada o usuario desconectado	Usuario registrado o conectado a la red inalámbrica	Usuario dado de baja
Recibir secuencia de usuario	Información externa	Perfil mejorado	Recibir información de sensores, etc.	El perfil ha mejorado
Mejorar Perfil	Información sobre el comportamiento del usuario	Perfil mejorado	Información recibida sobre el comportamiento del usuario	El perfil ha mejorado

Tabla 4.11: Servicios del Rol Administrador de Servicios con GAIA

Diagrama de Servicio: Administrador de Servicios (AS)				
Servicio	**Entradas**	**Salidas**	**Pre-Condiciones**	**Post-Condiciones**
Matching-servicios-perfiles	Perfil de usuario conocido por el central y y el proveedor Información y localización	Matching entre el perfil de usuario y el proveedor Información y localización	Usuario registrado	El resultado del matching

proveedores de acuerdo a las preferencias de los usuarios y advertir a los proveedores sobre las preferencias de esos usuarios.(ver Tabla 4.12)

Tabla 4.12: Servicios del Rol Descubridor de Proveedores utilizando GAIA

Diagrama de Servicio: Descubridor de Proveedores (DP)				
Servicio	**Entradas**	**Salidas**	**Pre-Condiciones**	**Post-Condiciones**
Filtrar proveedores	Resultado del matching	Proveedor cercano	El resultado del matching validado	Proveedor filtrado por localización e información
Advertir al proveedor	Proveedor filtrado	El proveedor recibe un mensaje de alerta	Obtener proveedores cercanos	El Proveedor informa a los usuarios cercanos

El rol Proveedor de Servicios (*Service Provider*) incorpora las siguientes funciones o servicios: ofrecer servicios a los usuarios, solicitar y aceptar negociaciones entre ellos e intercambiar información, tal como se describe en la Tabla 4.13.

Tabla 4.13: Servicios del Rol Proveedor de Servicios con GAIA

Diagrama de Servicio: Proveedor de Servicios (PS)				
Servicio	**Entradas**	**Salidas**	**Pre-Condiciones**	**Post-Condiciones**
Ofrecer servicios	Alertar a proveedor para informar	Servicios ofrecidos de acuerdo a la posición y al perfil de usuario	Proveedores cercanos advertidos	El usuario recibe servicios según sus preferencias
Solicitar negociación	Los proveedores solicitan a los usuarios negociar	El proceso de negociación es solicitado	–	El usuario acepta o rechaza el mensaje de solicitud
Aceptar la negociación	Enviar mensaje para negociar	Aceptar o no el mensaje para negociar	Solicitar mensaje	Proceso de negociación iniciado
Intercambio de información	Aceptar mensaje para negociar	Información para intercambios entre el proveedor y el usuario	Aceptar mensaje	Proceso de negociación finalizado

A continuación, se representan los servicios que se relacionan con el rol Recomendador (*Recommend*). Estos son: hacer preguntas sobre recomendaciones y recibir recomendaciones, recomendar usuarios, decidir en quien confiar.

El rol Administrador de Perfiles (*Profile Manager*) es el responsable de la actualización del

Tabla 4.14: Servicios del Rol Recomendador usando GAIA

Diagrama de Servicio: Rol de Recomendador (RR)				
Servicio	**Entradas**	**Salidas**	**Pre-Condiciones**	**Post-Condiciones**
Recomendar usuarios	Propuesta de recomendación	Aceptación o rechazo de la recomendación	–	Ocurrió o falló la recomendación
Preguntar por recomendaciones	Necesidad de recomendación	Solicitud de recomendaciones	–	El agente usuario recibe o no recomendaciones
Recibir recomendaciones	Solicitud de recomendaciones	Recomendaciones recibidas	Preguntar por recomendaciones	El usuario recibe las recomendaciones
Decidir si confiar	–	Decision de confianza	–	Compartir opinión con otros agentes

Tabla 4.15: Servicios del Rol Administrador de Perfiles en GAIA

Diagrama de Servicio: Administrador de Perfiles (PM)				
Servicio	**Entradas**	**Salidas**	**Pre-Condiciones**	**Post-Condiciones**
Actualizar-perfil-interno	Recomendaciones de otros usuarios	Actualización del perfil	Recibir otras recomendaciones de usuarios	Perfil actualizado
Enviar registro de perfil compartido	Información necesaria para el registro	Enviar mensaje de solicitud y perfil al agente intermediario	Los usuarios están conectados a la red inalámbrica	Usuarios registrados

perfil interno y del envío del registro de perfiles compartidos.

Al rol Negociador (*Negotiate*) le corresponden tareas tales como realizar consultas sobre ofertas recibidas, recepción de servicios personalizados, consultas por acuerdos, recibir solicitudes de negociación e intercambiar información hasta concluir con dicho proceso de negociación.

Tabla 4.16: Servicios del Rol de Negociador usando GAIA

Diagrama de Servicio: Rol de Negociador (RN)				
Servicio	**Entradas**	**Salidas**	**Pre-Condiciones**	**Post-Condiciones**
Consultar Información	Propuesta de negociación	Acuerdo o desacuerdo	–	Acuerdo logrado o fallido
Recibir servicios	Solicitud de servicios o resultados del matching validados	Servicios recibidos	Matching realizado por el agente intermediario y proveedor advertido para ofrecer servicios	Los usuarios obtienen servicios e información perfonalizada
Solicitar acuerdos	Necesidad de acuerdos con otros agentes	Solicitud para lograr un acuerdo	–	Acuerdo aceptado o rechazado
Recibir solicitudes de negociación	Agente usuario consulta al proveedor sobre acuerdos	Proceso de negociación solicitado	El agente usuario tiene intención de lograr un acuerdo	El proceso de negociación está abierto (o no) para el intercambio de la información requerida

Recibir solicitudes del usuario, enviar información compartida y crear grupos de usuarios, son los servicios que se manejan en el rol Administrador de Coaliciones (*Coalition Manager*).

Para finalizar con el modelo de servicios, los servicios relacionados con el rol Administrador de Confianza (*Trust Manager*) son: buscar sitios o actividades para recomendar y gestionar los mecanismos de reputación.

4.3.5 Modelo de Interacción

La construcción de diagramas de Protocolo de Interacción AUML llevan el diseño del sistema cerca de su implementación en relación al modelo de interacción propuesto por GAIA. En este trabajo se optó por trabajar con los Diagramas de Interacción AUML para analizar la interacción entre los agentes. Estos diagramas hacen posible el seguimiento de los mensajes entre agentes. Las acciones de los agentes presentes en el modelo de interacción son: registrar usuarios, advertir a proveedores, procesos de negociación, consultar sobre recomendaciones, recomendar actualizaciones de perfiles de usuario, intercambio de información, decidir en quien confiar y el problema de formación de coaliciones. A continuación se explica en detalle el

Tabla 4.17: Servicios del Rol Administrador de Coaliciones en GAIA

Diagrama de Servicio: Administrador de Coaliciones (AC)				
Servicio	**Entradas**	**Salidas**	**Pre-Condiciones**	**Post-Condiciones**
Recibir solicitud de usuario	Solicitud del agente usuario	Buscar usuarios con necesidades similares	–	Otros usuarios están advertidos para hacer acuerdos
Enviar información compartida	Perfiles de usuarios actualizados	–	Algunos perfiles modificados	El administrador de coaliciones actualiza su conocimiento sobre el perfil público del usuario
Formación de grupos	–	Un nuevo grupo de usuarios	Usuarios con intereses similares	El grupo de usuarios inicia un proceso de negociación con el agente proveedor

Tabla 4.18: Servicios del Rol Administrador de Confianza con GAIA

Diagrama de Servicio: Administrador de Confianza (TM)				
Servicio	**Entradas**	**Salidas**	**Pre-Condiciones**	**Post-Condiciones**
Buscar actividades o lugares para recomendar	El agente usuario consulta sobre recomendaciones	Recomendaciones	El administrador de confianza tiene los valores de reputación	El agente usuario recibe o no recomendaciones
Administrar mecanismos de reputación	Recibir perfiles de usuarios actualizados	–	Chequear que las actividades o lugares con esa valoración existan	Valores de reputación actualizados

protocolo de comunicación construido para el sistema multi-agente propuesto utilizando las especificaciones FIPA para el envío de mensajes, que luego se explica con detalle en la sección de Implementación del SMA.

Registrando Usuarios

En primera instancia, cuando un usuario quiere usar el sistema, debe crear una cuenta de usuario en el mismo. A esto le sigue el registro del agente usuario lo cual permite filtrar que usuario utilizará el sistema. Es una comunicación simple donde el agente usuario (usando su rol de administrador del perfil interno) solicita registrarse al agente administrador de usuarios (en su rol de administrador de usuarios).

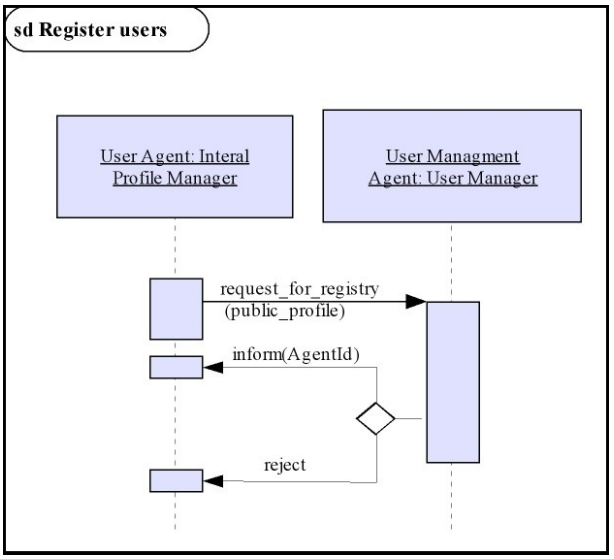

Figura 4.4: Diagrama de Interacción de AUML: Registro de Usuarios

Advertir a Proveedores

Este protocolo se basa en el servicio Advertir a Proveedores del Agente Intermediario (BA). El BA tiene conocimiento sobre todos los posibles servicios del entorno y la posición de cada proveedor. Después de que el LA retorna la posición del UA, le devuelve esa posición al BA. El BA filtra todos los proveedores cercanos al UA. Luego realiza una correspondencia entre el UA y los proveedores filtrados.

Proceso de Negociación

Luego de que se advierte al proveedor respecto de la presencia de un usuario, se puede iniciar entre ambos un proceso de negociación. El proveedor envía una oferta a un usuario. Cuando un agente usuario recibe la oferta, puede rechazarla, aceptarla o realizar alguna consulta sobre la misma. Para ello se necesita configurar algunos parámetros. Por ejemplo, el agente usuario puede consultar si es posible abonar con tarjeta de crédito o preguntar si le hacen precio por compras en cantidad. Luego el agente proveedor es quien rechaza el acuerdo, responde la consulta del usuario o simplemente acepta el acuerdo.

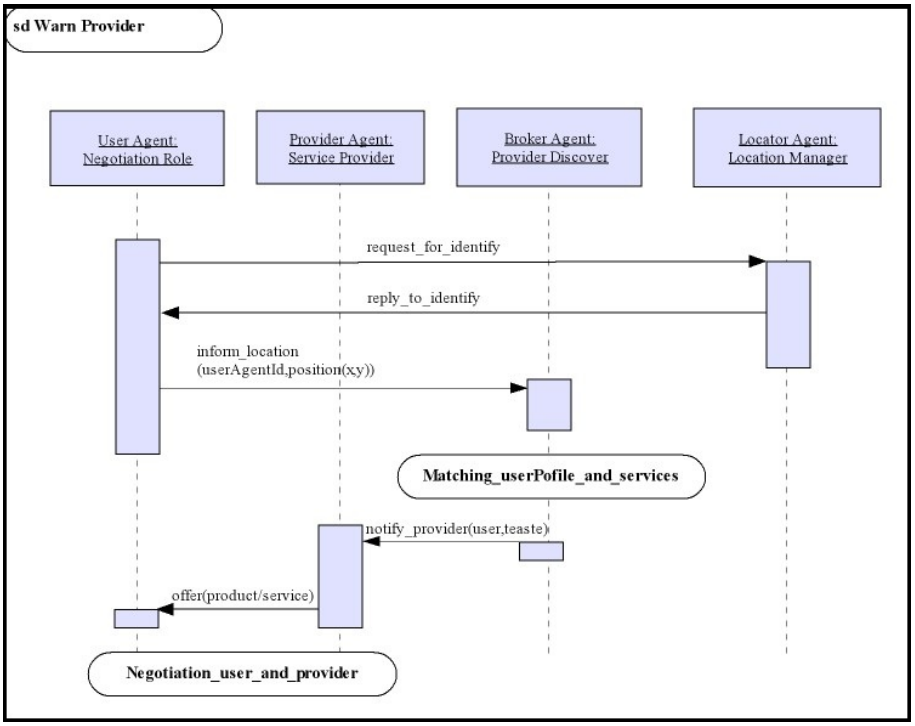

Figura 4.5: Diagrama de Interacción de AUML: Advertir a Proveedores.

Preguntar por Recomendaciones

Algunas veces el agente usuario necesita realizar preguntas sobre recomendaciones. Por ejemplo, puede ser que necesite recomendaciones sobre qué actividades se puede realizar en una ciudad o en una feria, o que sitios visitar. El agente usuario envía una solicitud al agente administrador de usuarios, quien busca la información y responde con los datos encontrados. El agente usuario puede quedar satisfecho y finalizar el proceso, o en otro caso, puede requerir una valoración (*assessment*). Finalmente, el agente usuario puede finalizar el proceso o puede iniciar un nuevo proceso de recomendaciones con otros usuarios.

Recomendar

Luego de hacer cualquier actividad, el agente usuario puede recomendar dicha actividad a otro agente usuario. En caso de que eso sea posible el segundo agente usuario puede requerir información adicional. También, un agente usuario puede pedir a sus pares recomendaciones acerca de un proveedor sobre el que tiene interés, antes de contactarlo.

Actualizando el Perfil de Usuario

El agente usuario puede tener actualizado su perfil público. Esto es, por ejemplo, cuando realiza una nueva actividad o realiza cambios sobre sus preferencias dinámicas. El agente administrador de usuarios debe enterarse de este cambio. Con este propósito, el agente usuario le informa respecto de la actualización. Luego, el agente administrador de usuarios puede aceptar o rechazar el cambio de acuerdo a sus creencias (*beliefs*).

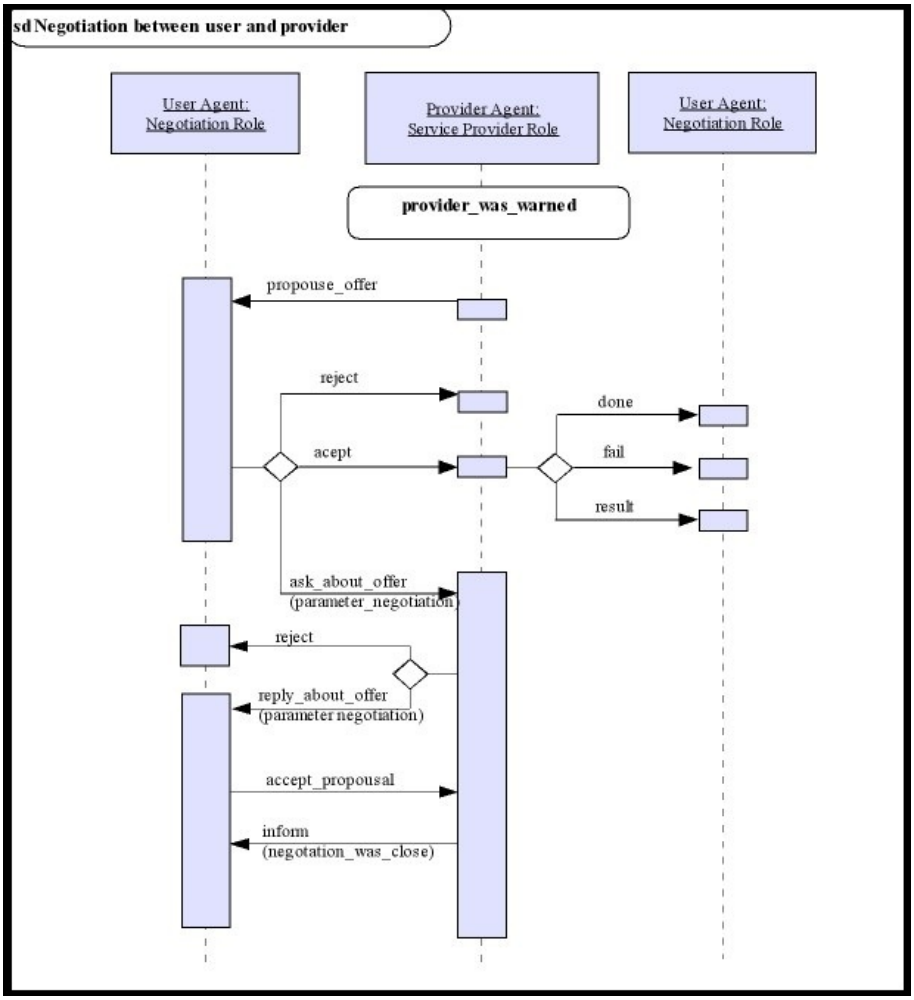

Figura 4.6: Diagrama de Interacción de AUML: Proceso de Negociación y Oferta.

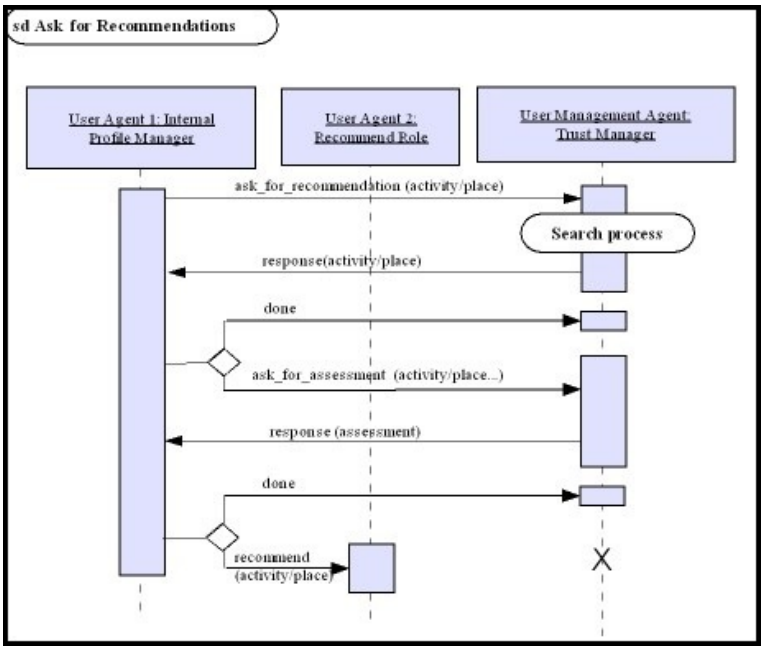

Figura 4.7: Diagrama de Interacción de AUML: Solicitar Recomendaciones

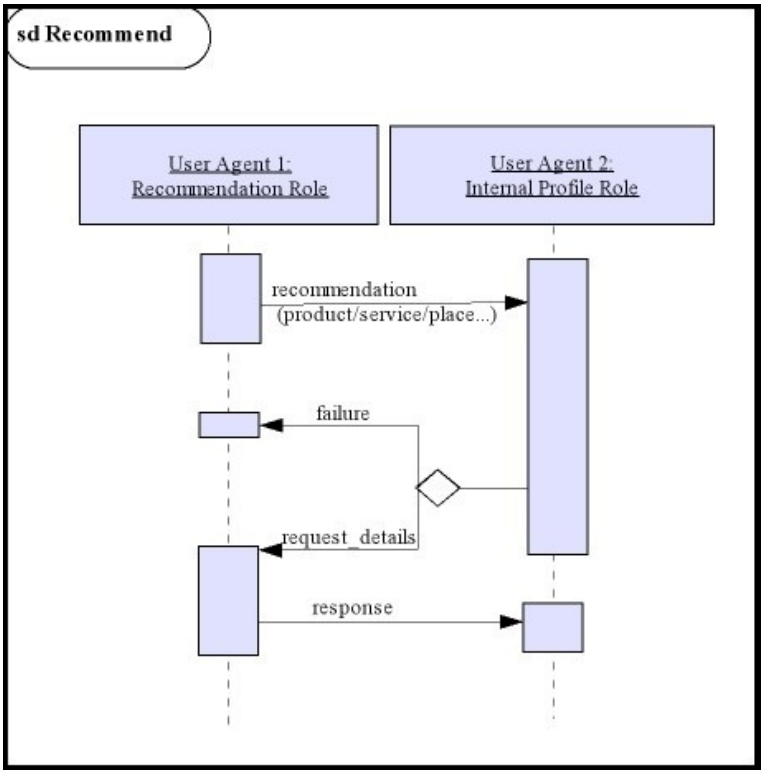

Figura 4.8: Diagrama de Interacción de AUML: Recomendar Servicios

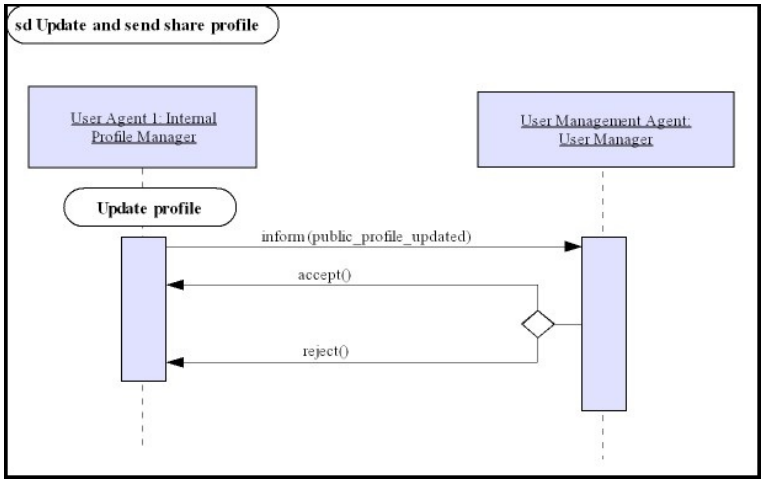

Figura 4.9: Diagrama de Interacción de AUML: Actualización y envío del Perfil Compartido.

Intercambiando Información El agente proveedor envía información a los agentes usuarios de productos o servicios, a los efectos de mejorar sus ofertas o realizar acuerdos.

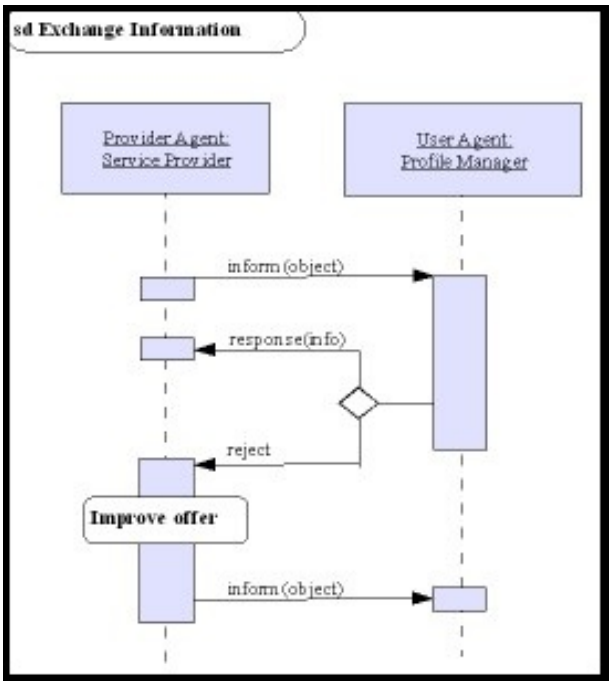

Figura 4.10: Diagrama de Interacción de AUML: Intercambio de Información.

Problema de Formación de coaliciones

La interacción entre agentes es sumamente necesaria para cumplir los objetivos tanto individuales como compartidos dentro del sistema multi-agente. En algunas ocasiones ciertos agentes se pueden ver beneficiados por otros agentes de la población (Merida Campos and Willmott, 2007). Los agentes deben explorar el espacio de soluciones del grupo de agentes para

mejorar el flujo de datos, ubicar de manera eficiente los recursos, resolver problemas de forma coordinada o mejorar sus productos creando alianzas entre ellos para llegar a los objetivos.

Suponemos entonces al Agente Administrador de Usuarios, UMA, como creador de una coalición en donde los candidatos que forman la coalición son todos aquellos agentes usuarios, UA, con intereses en común y que pertenecen a un mismo vecindario, tal como lo hacen en (Andriatrimoson et al., 2012). Al vecindario lo conforman aquellos agentes usuarios que se encuentran a una distancia considerablemente cercana unos de otros. El UMA intercambia mensajes con otros agentes usuarios que son potenciales miembros de la coalición. Si no hay candidatos, el UMA reinicia la coalición con nuevos parámetros, a los fines de encontrar agentes que se unan a la misma. Siguiendo el protocolo explicado por el autor de (Sotela, 2010), el UMA envía un mensaje FIPA para dar inicio a la coalición del tipo *CallForPropose*, y los agentes usuarios UA_i tienen la posibilidad de *Aceptar* o *Rechazar* la propuesta de coalición, como se muestra en la Fig. 4.11. Si todos los agentes rechazan la propuesta, el proceso de coalición falla. El creador de la coalición, es decir, el agente administrador de usuarios, vuelve a iniciar el proceso un número limitado de veces. En caso de que la coalición se haya formado, el UMA negocia el objeto de la coalición con el agente proveedor que pueda cumplirla. Posteriormente, el UMA envía un mensaje tipo *Inform_Coalition* a los agentes usuarios involucrados, con el fin de informarles sobre la resolución de la coalición. En algunos casos puede ser necesario, previamente, un mensaje tipo *Query_if* en caso de tener que ejecutar más ciclos de negociación y llegar a un acuerdo entre todos los candidatos de la coalición. Al producirse una negociación directa entre el UMA y el agente proveedor, se resguarda la privacidad del perfil de cada agente usuario. En cualquier instante del proceso, un agente usuario que se une a la coalición puede romper con la misma o unirse a otra. En caso de que un solo agente usuario se uniera a la coalición, el agente administrador de usuarios la cancelará.

En las aplicaciones de la AmI esto es un problema común. Un ejemplo ilustrativo puede darse en el escenario turístico planteado. Varios turistas tiene la idea de visitar el Museo de Alta Montaña ubicado en frente de la plaza principal de la ciudad. Esto se conoce puesto que figura en el perfil de cada agente usuario que se ejecuta en el teléfono móvil de los turistas. El agente administrador de usuarios detecta la coalición y se contacta con el agente proveedor del Museo de Alta Montaña. Entre ambos negocian una visita guiada a un precio promocional para un momento del día en que los visitantes no han planeado ninguna actividad. El agente administrador de coaliciones informa la propuesta final a los visitantes.

Decidir si Confiar

Dada la autonomía de los agentes, es imposible tener control sobre el comportamiento interno de cada agente en el sistema. Quedaríamos expuestos a enfrentar algunos riesgos en el caso de decidir confiar en cada uno de ellos. Este problema se puede resolver incluyendo un modelo de confianza a los fines de filtrar a aquellos agentes con comportamientos maliciosos. En este sentido, cada agente debería incluir dentro de sus creencias un modelo de los otros agentes, lo cual le ayudaría a decidir en quien confiar. Habitualmente, el proceso de decisión de confianza utiliza el concepto de reputación como factor clave para decidir en quien confiar tal como lo exponen (Conte and Paolucci, 2002). En el capítulo 7 se presenta un modelo de reputación y confianza para un SMA en entornos inteligentes. En el siguiente diagrama de interacción, un agente usuario le consulta al administrador de agentes usuarios respecto del valor de reputación de un tercer agente. Este le retorna el valor solicitado y de acuerdo al conocimiento que tenga el

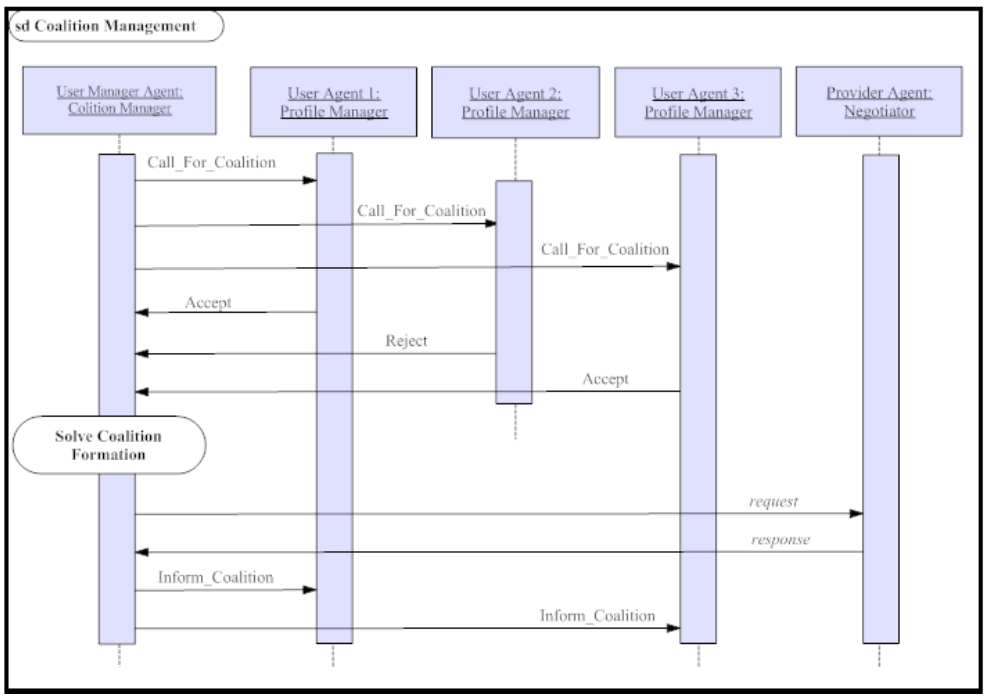

Figura 4.11: Diagrama de Interacción de AUML: Administración de Coaliciones

agente administrador el agente usuario interesado decidirá si confiar o no en esta tercera parte. Este tipo de interacción se implementa a través de un modelo de confianza centralizado que maneja el administrador de agentes usuarios, lo que supone que se aplica un criterio objetivo de aceptación. Lo que se propone luego en el modelo de reputación descrito más adelante, es complementar esta observación centralizada del comportamiento de los agentes, con un modelo de confianza distribuido donde los agentes comuniquen directamente sus evaluaciones subjetivas (basadas en sus experiencias directas, es decir, en sus propias observaciones).

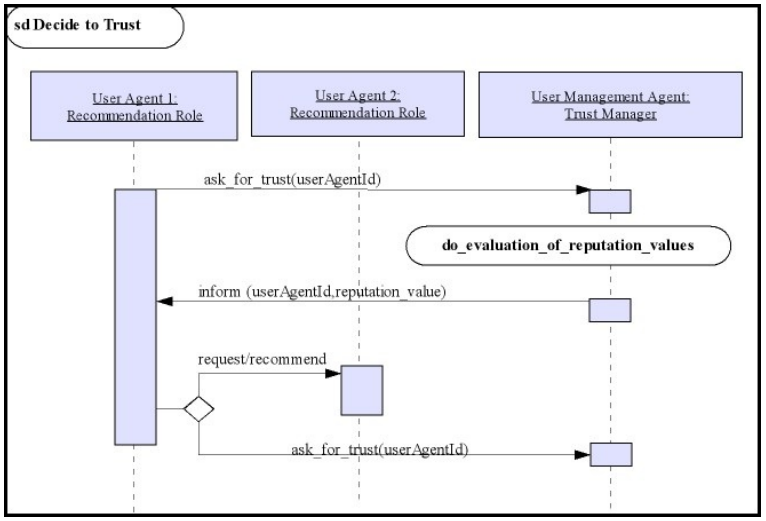

Figura 4.12: Diagrama de interacción de AUML: Decidir si confiar

4.3.6 Diagramas de Colaboración

Los diagramas de colaboración representan instancias de agentes en sus roles de agentes. Siguiendo la propuesta de (Odell et al., 2000), se diseña el diagrama de colaboración donde se muestran las interacciones de los agentes con los roles. Estos diagramas permiten obtener un esquema final de la arquitectura del sistema multi-agente. El diagrama se muestra en la Fig. 4.13: cada rectángulo de la figura es un agente. La secuencia de las interacciones se ennumera sobre el diagrama de colaboración. Las líneas de puntos representan el rol que juega cada agente al momento de la interacción.

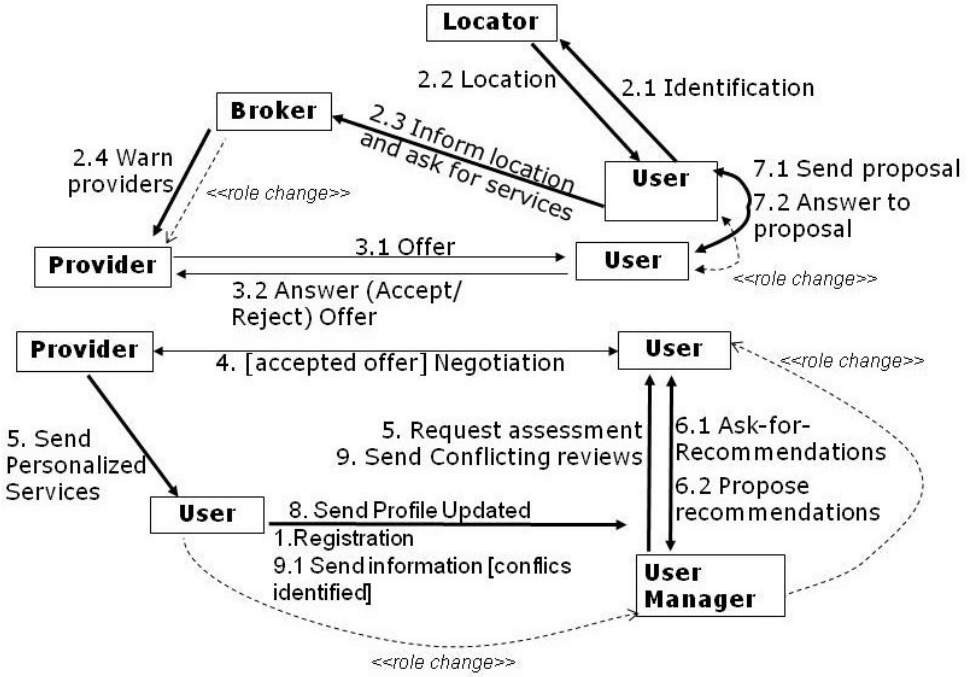

Figura 4.13: Diagrama de Colaboración AUML del SMA para AmI.

4.4 Arquitectura SMA para AMI

La arquitectura propuesta está compuesta por los siguientes agentes: Proveedores (Providers - PA), Usuarios (Users - UA), Intermediario (Broker - BA), Localizador (Locator - LA) y Administrador de Usuarios (User Manager - UMA), de acuerdo con el diseño detallado anteriormente y al modelo BDI. En la Fig. 4.14 se expone la arquitectura propuesta.

Primero, cada usuario se registra con el agente administrador de usuarios, UMA. Luego, el usuario agente, UA, debe identificarse con el agente localizador, LA, quien le retorna su posición actual. El LA, dependiendo del sistema, puede ser un agente de fusión de datos cuando el sistema trabaja con distintas tecnologías de localización como wireless, wimax o ultrawideband. El LA es el responsable del razonamiento sobre los datos espacio-temporales. Por su parte, el UA requerirá diferentes servicios de acuerdo a la posición del usuario. Para ello, debe comunicarse

con el BA. Este agente es el que realizará la correspondencia entre las preferencias del usuario y los servicios ofrecidos por los proveedores que se encuentren en la misma zona. El BA es también responsable de la gestión de servicios y del descubrimiento de proveedores.

El proceso de negociación inicia una vez que el PA haya sido advertido sobre las preferencias de un usuario con un perfil específico. Este proceso es bastante simple: un usuario solicita un producto, para lograr este objetivo, el usuario puede consultar el precio de un producto, por ejemplo. Pero también esta negociación puede estar basada en servicios. Además, el proveedor puede enviar ofertas personalizadas (luego de un proceso de correspondencia realizado por el BA).

Por su parte, un UA puede comunicarse directamente con otros UAs a los fines de realizar negociaciones o pedir recomendaciones. El UMA juega un papel muy importante en esta operación. Por ejemplo, si el diseño del sistema incluye información sobre reputación, el UMA asume el rol de ejecutar la correspondencia entre los perfiles de usuarios. En otro caso, puede ser un agente que envía recomendaciones de manera proactiva. Al mismo tiempo, el UMA puede manejar las coaliciones entre agentes sobre intereses comunes.

El framework ha sido diseñado para un dominio genérico que soporte todo tipo de actividades posibles en el entorno: modelos de confianza, negociación, localización y gestión de perfiles.

Los agentes administrador de usuarios (UMA), intermediario (BA) y localizador (LA) se ejecutan en uno o varios servidores. Mientras que por su parte cada agente proveedor, PA, corren sobre un servidor particular. Y, finalmente, UAs se ejecutan en los dispositivos móviles de los usuarios.

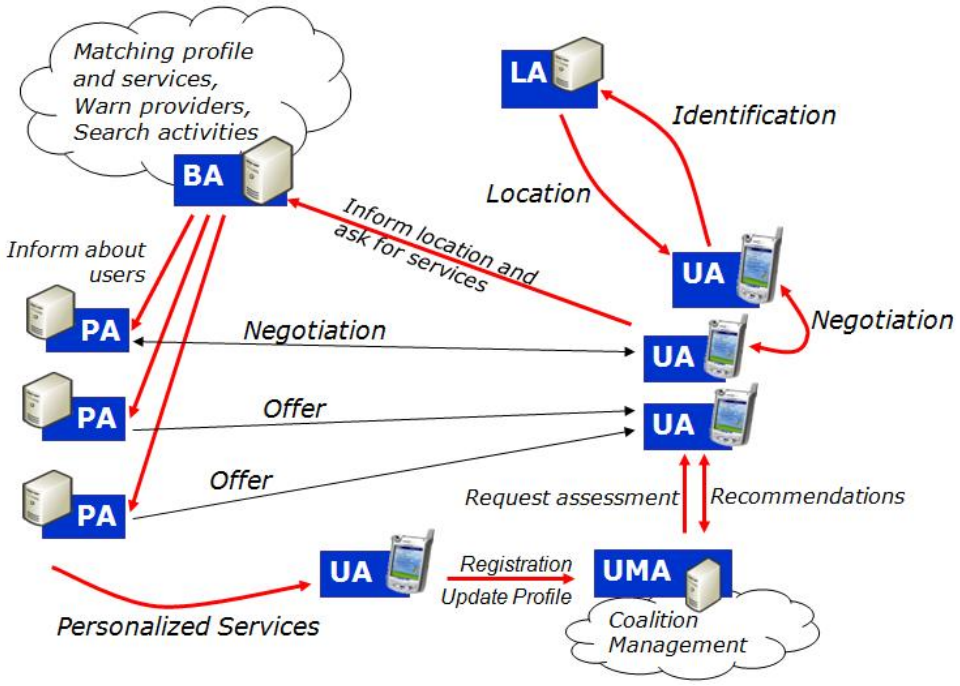

Figura 4.14: Arquitectura del sistema multi-agente basado en contexto y localización para dominios heterogéneos.

En la Tabla 4.19 tomamos diferentes dominios de aplicación para mostrar la adaptación de la arquitectura propuesta sobre entornos heterogéneos. El entorno de conferencia, ya descrito en la definición del problema, es aquel en el cual dentro de un mismo edificio o lugar físico, una persona puede entrar a distintas salas para atender diferentes conferencias. La aplicación eTourism fue realizada en el dominio de una persona que va de turismo a otra ciudad. Y por último, la feria SIMO que es una feria tecnológica en Madrid. Ahora bien, es importante aclarar que esta arquitectura no prevee la transición de un contexto a otro, por lo tanto, cada vez que el usuario cambie de entorno, su agente usuario le requerirá autenticarse para que los demás agente detecten su presencia y los servicios posibles en el dominio queden a su alcance.

El siguiente gráfico (Fig. 4.15) muestra la arquitectura del SMA y cómo se integran los demás módulos propuestos en los siguientes capítulos de esta tesis para dar solución a tres problemas básicos: cómo se estructura la información que intercambian los agentes, cómo se realiza la extracción del perfil del usuario observando su comportamiento y cómo se gestionan las relaciones de confianza entre los agentes usuarios.

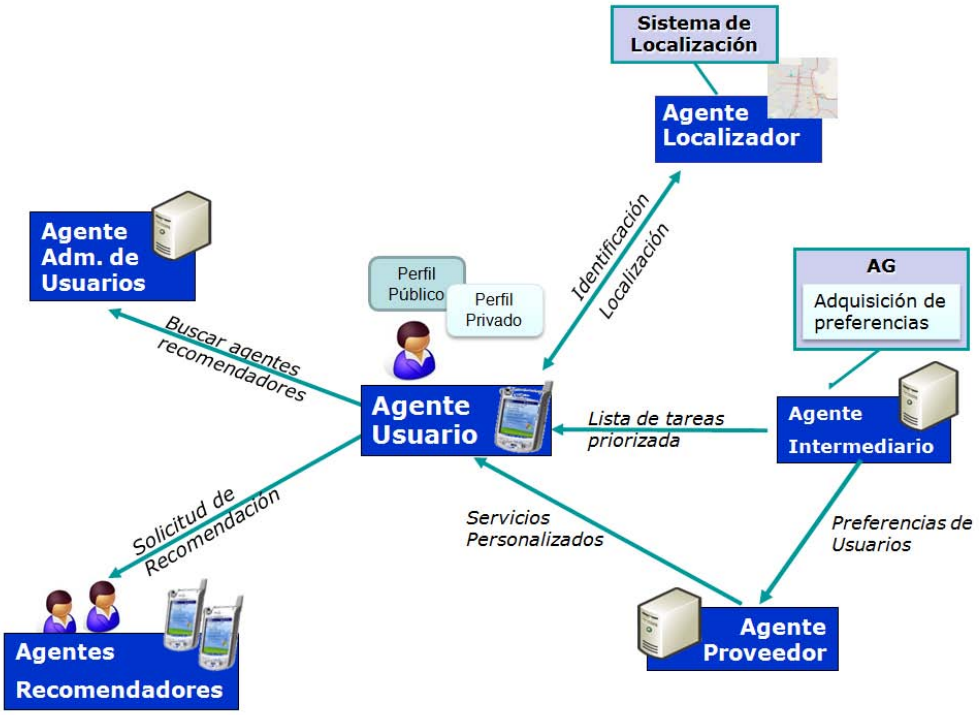

Figura 4.15: Estructura de la solución propuesta incluyendo los agentes de la arquitectura.

4.5 Implementación del SMA

La implementación del sistema ha sido desarrollada en JAVA. Específicamente se han utilizado las librerías JADE para los agentes que se ejecutan en el servidor, mientras que para los agentes de los dispositivos móviles se utilizó JADE-LEAP (Sánchez-Pi et al., 2008). La ontología que

Tabla 4.19: Implementación del SMA en diferentes escenarios de AmI

Agente	Entorno de conferencias	eTourism	Feria SIMO
Localizador	Ubica la posición del usuario respecto de las salas de conferencias o de sesión de pósters de acuerdo a su interés.	De acuerdo a la posición del turista, se analiza la proximidad a centros históricos, hoteles, restaurantes, parques de diversión o centros comerciales.	Toma la posición del visitante para tener en cuenta cual es la distancia que tiene a los stands de su interés.
Administrador de Usuarios	Registro de Usuario	Registro de Usuario	Registro de Usuario
Intermediario	Realizar el matching entre el perfil del oyente y lo que dictará el disertante. Advertir al disertante.	Realizar el matching entre el perfil del turista y los proveedores de entretenimiento, comidas, etc. Advertir a los proveedores.	Realizar el matching el perfil del asistente, posible comprador, y los expositores o vendedores de productos y servicios. Adevertir a los expositores.
Usuarios	Consultar información sobre la conferencia. Realizar recomendaciones a otros oyentes sobre un póster que le pareció interesante. Intercambio de información con el ponentes u otros asistentes.	Consultar información sobre puntos históricos, centros comerciales, medios de transporte. Realizar recomendaciones a otros turistas sobre buenos restaurantes, obras de teatro, museos, entretenimientos familiares. Intercambio de información con otros turistas.	Consultar información sobre los productos que ofrecer, cómo móviles, tablets, Pcs. Realizar recomendaciones a otros visitantes de la feria o amigos. Intercambio de información con otros visitantes y los expositores.
Proveedores	Ofrecer servicios o productos: posters y seminarios. Negociar: realizar acuerdos para trabajar juntos.	Ofrecer servicios o productos: menúes, tickets, bonos de transporte turísticos. Negociar: alquilar un vehículo, comprar entradas para una obra de teatro o para el museo.	Ofrecer servicios o productos: ponencia. Negociar: comprar un móvil, proponer ser distribuidor de un producto, negociar con otros proveedores de la feria.

permite la comunicación de los agentes, ha sido desarrollada con la herramienta Protegeè [1] y se describe en el siguiente capítulo 5.

Los comportamientos programados para cada agente se listan a continuación y se resumen en en anexo C:

- *Agente Usuario (User Agent): AskFor Agreements; AskForReputation; MakeRecommendations; ProposeDutchAuction; ReceiveMsg; ResponseReputation; ResponseRecommendation.*
- *Agente Proveedor (Provider Agent): Register, ContractNet; DutchAuction; OfferRequestService; OfferServices; RequestNegotiation.*
- *Agente Localizador (Locator Agent): DetectClient.*
- *Agente Intermerdiario (Broker Agent): ReplyToRegister; WarnProvider.*
- *Agente Administrador de Usuarios (User Manager Agent): ClientFiltering.*

Cada tipo de agente extiende de la clase *Agent* del paquete JADE de JAVA. A su vez, los comportamientos de los agentes pueden extender de diferentes clases heredadas de una clase superior denominada *behaviours (comportamientos)*. El tipo de comportamiento indicará cómo reaccionará el agente cada vez que reciba un mensaje, pudiendo quedarse constantemente a la espera de mensajes antes y luego de una acción, o pudiendo recibir un mensaje, ejecutar su tarea y finalizar.

Las tareas o servicios que un agente realiza, se especifican en comportamientos. Un comportamiento (*behaviour*) hace referencia a una funcionalidad que incorpora el agente. A continuación se explican los comportamientos implementados en el sistema explicados por Botía en (Botía, 2005) e implementados en JADE:

- Comportamientos one-shot: tipos de comportamiento los cuales se ejecutan de manera casi instantánea, y solamente una vez.
- Comportamientos cíclicos (*CyclicBehaviour*): son aquellos que nunca son sacados del conjunto de comportamientos del agente y cuyo método *action()* siempre ejecuta el mismo código. Por lo tanto, nunca finalizan.
- Comportamientos genéricos (*Behaviour*): algo mas complejos, el código que se ejecuta en ellos depende del estado del agente, y eventualmente finalizan su ejecución.

En el sistema se implementaron tres protocolos: Una oferta de productos, en donde un agente proveedor debe ofrecer un libro a un usuario, y este decide si comprar o no; el protocolo Contract-Net que consiste en los pasos que deben seguir los agentes intervinientes en un proceso de negociación (ver para más detalle (Mas, 2004)) ; y finalmente, una Subasta Holandesa, que es similar a cualquier subasta, solo que en vez de que el precio crezca en cada ronda, aquí decae.

[1] http://protege.stanford.edu

Para la implementación del protocolo de Oferta de Productos en primera instancia el agente localizador detecta la presencia de un agente usuario por medio del comportamiento *DetectClient*, y le evía la posición del mismo al agente intermediario. El agente intermediario en ese momento mediante el comportamiento *WarnProvider* advierte al proveedor *Book-Shop*, agente proveedor implementado, acerca de la presencia de un posible comprador, habiendo cruzado previamente los intereses del usuario con los del proveedor. La implementación del comportamiento *WarnProvider* se muestra en el siguiente código:

```
Clase WarnProvider viene-de OneShotBehaviour
{
   Atributos InterfazBroker interfaz;
   Codec codec;
   Ontology ontology;
   String nameClientAgent;
   //Creates a new instance of WarnProvider
   Constructor WarnProvider(Agent agente, InterfazBroker interfaz)
   {
      super();
      myAgent=agente;
      this.interfaz= interfaz;
   }
   Método action()

      String nameProviderAgent;
      nameProviderAgent <– "BookShop";
      Escribir (Broker said: Provider + nameProviderAgent+
      , you have a closer client....);
      Objeto codec ejemplar-de LEAPCodec();
      ontology <– LEAPOntology.getInstance();
      myAgent.getContentManager().registerLanguage(codec);
      myAgent.getContentManager().registerOntology(ontology);
      Objeto id ejemplar-de AID (nameProviderAgent,AID.ISLOCALNAME);
      Objeto msg ejemplar-de ACLMessage(ACLMessage.INFORM);
      msg.addReceiver(id);
      msg.setOntology(ontology.getName());
      msg.setLanguage(codec.getName());
      String sClientName <– "Albert";
      msg.setContent(sClientName);
      myAgent.send(msg);
      String messenger = msg.getContent();
      Escribir (Provider you have to contact with client:+messenger);
   Fin Método
}
```

Ahora bien, el agente proveedor *Book-Shop*, recibe en la clase *Provider* (clase que genera al agente) el mensaje que el agente intermediario (*Broker*) envió utilizando el comportamiento *WarnProvider* descrito anteriormente. El código a continuación pertenece a la clase *Provider*:

```
//Espera un mensaje del agente intermediario
Esperar(3000);
Objeto msg ejemplar-de ACLMessage;
msg <– this.receive();
Si (msg != null)
   //El Agente Proveedor ha recibido el msn del Agente Intermediario;
   String idCliente <– msg.getContent();
   Si (idCliente != null)
      addBehaviour(nuevo OfferRequestService( interfaz, msg));
```

```
    Fin Si
Fin Si
```

Luego, el agente Book-Shop lanza el comportamiento *OfferServices*. Este comportamiento envía un mensaje de tipo *Propose* al agente Alberto, detectado por el agente localizador, cuyo contenido es un producto en oferta con el precio del mismo.

El mensaje que recibe el proveedor, que además contiene el nombre del cliente al que debe contactar, es procesado en el constructor de la clase *OfferRequestService*, como se observa a continuación:

```
Método (OfferRequestService(Objeto interfaz ejemplar-de InterfazProvider, Objeto msgReceiveFromBroker ejemplar-de
ACLMessage, Objeto negotiationParameters ejemplar-de NegotiationParameters )
    this.interfaz <– interfaz;
    msgReceiveProvider <– msgReceiveFromBroker;
    closerClient <– msgReceiveFromBroker.getContent();
    paramNegotiation <– negotiationParameters;
Fin Método
```

En el método action de *OfferRequestService*, se implementan las *performatives Propose* para ofrecer el producto e *Inform* para responder ante alguna consulta.

Durante el proceso de negociación se utiliza el siguiente identificado de conversación *setConversationId("book-trade")*.

Para que sea posible que el agente usuario reciba e interprete el mensaje, se ha construido el comportamiento *ReceiveMsg*. Este último es un comportamiento del tipo *CyclicBehaviour*, que se activa ante la llegada de un mensaje de tipo *Propose* el cual contiene los conceptos de la ontología *Offer, Products* que se explicará en el capítulo siguiente.

En cuanto al comportamiento del cliente *ReceiveMsg*, por cada mensaje que se recibe, se chequea si es de tipo *Propose* o *Inform*. Por el momento, en el caso de recibir una propuesta, el agente cliente formula una pregunta en base a la misma, y cuando recibe la contestación directamente acepta la propuesta (Accept_Proposal).

```
Método action()
    //El Agente Cliente hace una consulta al Proveedor sobre la propuesta recibida
    Si (oParameters != null )
        iPayForm <– NegotiationBeliefs((String)oParameters[0]);
        paramCardNumber <– (String)oParameters[1];
    Fin Si
    ACLMessage msg <– myAgent.receive();
    Objeto codec ejemplar-de SLCodec();
    ontology <– (LEAPOntology) LEAPOntology.getInstance();
    Si (msg != null)
        ACLMessage reply <– msg.createReply();
        Si (msg.getPerformative()==ACLMessage.PROPOSE)
            Offer myOffer;
            myOffer <– (Offer) msg.getContentObject();
            Product producto <– (Product) myOffer.getHasProduct();
            sProductId <– producto.getProductID();
            iProductPrice <– producto.getPrice();
            //start negotiation with this offer
            NegotiationParameters myNegotiation <– new NegotiationParameters();
```

```
            myNegotiation.setPayForm(iPayForm);
            reply.setPerformative(ACLMessage.REQUEST);
            reply.setSender(myAgent.getAID());
            myAgent.getContentManager().registerLanguage(codec);
            myAgent.getContentManager().registerOntology(ontology);
            reply.setContentObject(myNegotiation);
            myAgent.send(reply);
        Fin Si
        Si (msg.getPerformative()==ACLMessage.INFORM)
            reply.setPerformative(ACLMessage.ACCEPT_PROPOSAL);
            NegotiationParameters myNegotiation <- new NegotiationParameters();
            myNegotiation.setCardNumber(paramCardNumber);
            reply.setContentObject(myNegotiation);
            reply.setSender(myAgent.getAID());
            reply.setConversationId("book-trade");
            reply.setReplyWith("order" + System.currentTimeMillis());
            cantMsn.setMsnAccept();
            myAgent.send(reply);
            Escribir("Client said: Accept treatment");
            myAgent.blockingReceive();
        Fin Si
    Fin Si
Si-no
        bloquear-agente();
Fin si-no
Fin Método
```

El agente usuario además posee un comportamiento *AskForAgreements*, para enviar un mensaje a los proveedores, a modo de solicitud. El comportamiento *AskForAgreements* se implementa a partir de la clase *OneShotBehaviour*, y consiste en el envío de un mensaje de tipo Request al agente proveedor en cuestión. El proveedor, al recibir este mensaje, utilizará el comportamiento *RequestNegotiation*, creado para aceptar o rechazar alguna propuesta del cliente, contestando la solicitud del agente usuario.

Para la implementación del protocolo de la Subasta Holandesa, el cuál consiste en que el proveedor realiza una oferta de un producto a un precio alto, el cual va bajando a medida que los clientes (agentes usuarios) envían sus propuestas de precios, hasta que haya un comprador. Más detalles pueden leerse en (Mas, 2004). Para la simulación de este protocolo se programaron los comportamientos *ProposeDutchAuction*, en el usuario, y *DutchAuction*, en el agente proveedor. El comportamiento *DutchAuction* se implementa de la clase *Behaviour*, y primeramente crea un mensaje FIPA de tipo *CFP* (call for propose) que será enviado a un grupo de usuarios con un perfil particular. El escenario de ejemplo para la ejecución de la simulación consiste en un agente proveedor denominado *ArtProvider* que tiene a la venta una réplica del cuadro de la Mona Lisa, que ofrecerá a $205. Se crean dos agentes usuarios que van a competir por el cuadro enviando sus ofertas como parámetros: Alberto y Juan.

La porción de código siguiente corresponde al método *action* del comportamiento *DutchAution* de *ArtProvider*. Primero el proveedor prepara un mensaje *cfp*, agregando como destinatarios a todos los agentes de la lista de agentes usuarios que recibe como argumento. Luego asigna la onotlogía con la que se van a comunicar, de la cual se utilizan dos conceptos para el intercambio de información *Product* y *Offer*. Se asigna al producto *sArt* el identificador 'Mona Lisa' y el precio. Luego se crea el objeto *myOffer* de tipo *Offer* para enviar el objeto *sArt*. Con la línea *cfp.setContentObject(myOffer);* se configura el contenido del mensaje que recibirán y

decodificarán luego los compradores potenciales. Finalmente se envía el mensaje haciendo *myAgent.send(cfp)*.

```
ACLMessage cfp = new ACLMessage(ACLMessage.CFP);
Para i desde i=0 hasta i<args.length-1 incremento 1 hacer
   Objeto buyer ejemplar-de AID( (String) args[i], AID.ISLOCALNAME);
   Escribir("Buyer " + i + ": " + args[i]);
   cfp.addReceiver(buyer);
Fin Para
cfp.setOntology(ontology.getName());
cfp.setLanguage(codec.getName());
//Ofrecer un cuadro en una subasta
Objeto sArt ejemplar-de Product();
sArt.setProductID("Mona Lisa");
String sPrecioSubastaProducto <– 205;
sArt.setPrice(sPrecioSubastaProducto);
Escribir("Provider auction: " + sArt.getProductID() + " - " + sArt.getPrice());
Objeto myOffer ejemplar-de Offer();
myOffer.setHasProduct(sArt);
cfp.setContentObject(myOffer);
cfp.setConversationId("art-auction");
cfp.setReplyWith("cfp" + System.currentTimeMillis());
myAgent.send(cfp);
```

Cada agente usuario de la lista recibirá el mensaje *cfp* a través de su comportamiento *ProposeDutchAuction* que extiende de la clase *CyclicBehaviour*, dado que constantemente el agente se queda a la espera de una propuesta del proveedor lo cual se conoce por medio de la sentencia *ACLMessage msg = myAgent.receive();*, en el método *Action*, cómo se muestra a continuación:

```
Método action()
{
   Objeto msg ejemplar-de myAgent.receive();
   Si (msg != null)
   { ...}
   Fin Si
Fin Método
```

Una vez que se recibe el mensaje pueden suceder dos cosas, el mensaje no es comprensible por el usuario lo cual se corrobora de la siguiente manera:

```
Si (msg.getPerformative()==ACLMessage.NOT_UNDERSTOOD)
   { ...}
Fin Si
```

o el mensaje corresponde a los tipos *Propose, Cfp o Inform*, en cuyo caso el agente ejecutará la porción de código correspondiente. Si el mensaje recibido es un *call for propose* o un *propose*, el agente usuario procederá a decodificar el mensaje para ver que producto y a qué precio se está haciendo la oferta. Luego generará de manera aleatoria para la simulación (*priceOffered = RandomAuctionPriceGenerator(iProductPrice);*), un precio (verificando en la función que lo genera que sea menor al precio propuesto por el proveedor), con el cuál intentará conquistar al vendedor. Finalmente enviará su el precio ofertado a través de un mensaje del tipo *Propose* usando el concepto de la ontología *NegotiationParameters*.

```
Si ( (msg.getPerformative()==ACLMessage.PROPOSE) ||
(msg.getPerformative()==ACLMessage.CFP) )
   OfferReceived <– (Offer) msg.getContentObject();
   ProductOffered <– OfferReceived.getHasProduct ();
   sProductId <– ProductOffered.getProductID();
   iProductPrice <– ProductOffered.getPrice();
   Escribir("Oferta recibida por el Cliente" + myAgent.getName() + " : " +
   sProductId + " - " + iProductPrice);
   priceOffered <– RandomAuctionPriceGenerator(iProductPrice);
   //start negotiation with this offer
   Objeto myNegotiation ejemplar-de NegotiationParameters();
   myNegotiation.setMyOffer(priceOffered);
   reply.setPerformative(ACLMessage.PROPOSE);
   reply.setSender(myAgent.getAID());
   myAgent.getContentManager().registerLanguage(codec);
   myAgent.getContentManager().registerOntology(ontology);
   reply.setContentObject(myNegotiation);
   Escribir( "Agente: " + myAgent.getName() + "My offer: " + myNegotiation.getMyOffer());
   cantMsn.setMsnPropose();
   myAgent.send(reply);
Fin Si
```

En caso que el usuario reciba un mensaje de tipo *Inform*, contesta con un mensaje *Accept_Proposal* aceptando la compra y envía su número de tarjeta de crédito para el pago.

```
Si (msg.getPerformative()==ACLMessage.INFORM)
   reply.setPerformative(ACLMessage.ACCEPT_PROPOSAL);
   Objeto myNegotiation ejemplar-de NegotiationParameters();
   myNegotiation.setCardNumber(paramCardNumber);
   reply.setContentObject(myNegotiation);
   reply.setSender(myAgent.getAID());
   reply.setConversationId("art-auction");
   reply.setReplyWith("order" + System.currentTimeMillis());
   cantMsn.setMsnAccept();
   myAgent.send(reply);
   System.out.println("Client said: Accept treatment");
   myAgent.blockingReceive();
Fin Si
```

Y finalmente, al ser un comportamiento cíclico, el agente se quedará bloqueado a la espera de un nuevo mensaje: *this.block();*. Un ejemplo de los agentes ejecutándose sin interfáz gráfica se muestra en la Fig. 4.16.

Por último, a los fines de probar un mecanismo de reputación, se desarrollaron en primera instancia los comportamientos *MakeRecommendation* y *ResponseRecommendation* para poder validar la arquitectura, sin la intención de crear un escenario de reputación propiamente dicho. Posteriormente, al elaborar un modelo de confianza y reputación, se han refinado estas clases en otros comportamientos, que se expone luego en el capítulo 7. Continuando con el ejemplo de solicitud de recomendaciones, el agente usuario tiene dos comportamientos paralelos asignados en la clase *Client* (clase que representa a los usuarios en el código fuente):

```
 ParallelBehaviour paralelo1 = new ParallelBehaviour(this, 1);
paralelo1.addSubBehaviour(new MakeRecommendations(this,myconcept,parameters));
paralelo1.addSubBehaviour(new ResponseRecommendation(this,myconcept,parameters));
```

;

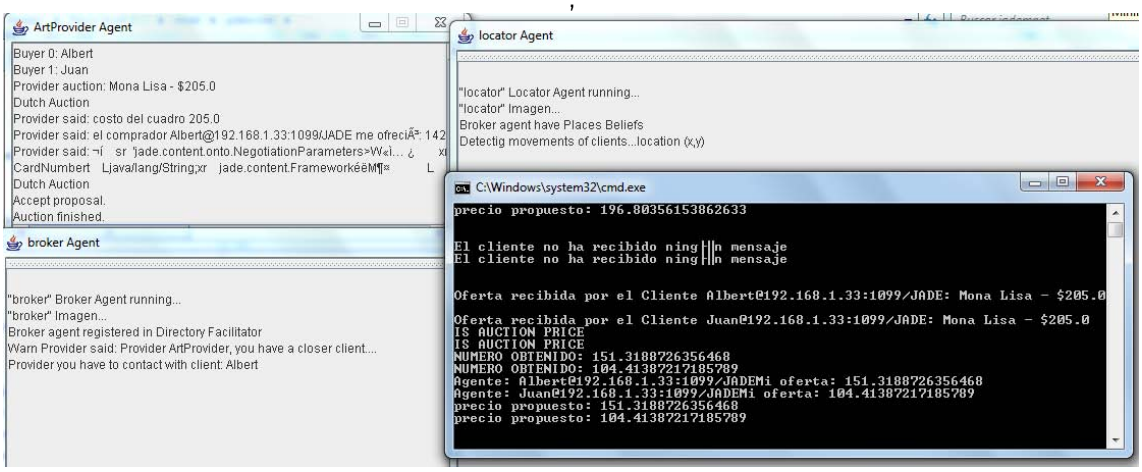

Figura 4.16: Los agentes del sistema ejecutando la Subasta Holandesa

```
addBehaviour(paralelo1);
```

MakeRecommendations es un comportamiento cíclico que envía a algún usuario (previamente elegido por el Agente Administrador de Usuarios (*User Manager Agent*), usando su comportamiento *ClientFiltering*, en base a intereses similares a los suyos) un mensaje de tipo *Request* solicitando un valor de reputación sobre un proveedor y se queda a la espera de una respuesta. El agente usuario que recibe el mensaje envía su respuesta usando el comportamiento *ResponseRecommendation*, de tipo *OneShotBehaviour*, a través de un mensaje *Inform* cuyo contenido es la valoración del proveedor.

En cada ejemplo ejecutado el Agente Localizador (*Locator Agent*) a ubicado la posición del usuario a través del comportamiento cíclico *DetectClient* (detectar cliente):

```
Método action()
    interfaz.setText("Detectig movements of clients...location (x,y)");
    //reading txt file code about the location of the client
Fin Método
```

Mientras que el Agente Intermediario (*Broker Agent*) ha identificado a cada proveedor con su comportamiento *ReplyToRegister*, y por cada usuario identificado por el localizador, ha ejecutado su comportamiento *WarnProvider* para informar al proveedor del caso sobre la presencia de un potencial cliente. En este último comportamiento se usa una función para cruzar los datos del proveedor y el cliente según los conceptos de la ontología.

4.6 Evaluación

La investigación sobre arquitecturas SMA se encuentra en continuo crecimiento, pero los métodos para evaluarlas aún no son suficientes. Algunos de los trabajos enfocados a la evaluación de SMA se describen en (Davidsson et al., 2006) (Joumaa et al., 2008). En esta sección se examinan

varias soluciones con el fin de evaluar la arquitectura descrita anteriormente. Es importante mencionar, que ninguna de las soluciones analizadas cubre todos los aspectos de la evaluación de una arquitectura.

La idea de esta sección es obtener ciertas características de los diferentes métodos de evaluación a los fines de realizar una evaluación completa de la arquitectura propuesta en este trabajo.

En primer lugar se evalúa el sistema en relación a una arquitectura centralizada propuesta por una colega del grupo de investigación y luego se realiza una evaluación subjetiva en contraste con otras arquitecturas propuestas por investigadores del área de Inteligencia Ambiental.

4.6.1 Evaluación centrada en el Intercambio de Mensajes

La evaluación de esta sección considera dos arquitecturas, SMA1 de (Fuentes et al., 2007), y la arquitectura propuesta en este capítulo: MAS2. Ambas incluyen agentes clientes que reciben servicios ofrecidos por proveedores, mientras que los agentes intermediarios facilitan la correspondencia entre ellos. Se implementan los mismos protocolos y hacen uso de la misma ontología para AmI (publicada en (Fuentes et al., 2006a)) y actualizada en (Venturini et al., 2010). Las principales diferencias entre ambos sistemas son:

- En SMA1 (Fuentes et al., 2007), la arquitectura está compuesta por tres tipos de agentes: el Agente Central, los Agentes Proveedores (uno por cada punto de interés) y agentes usuarios (uno por usuario). La comunicación entre los agentes se inicia cuando el Agente Central advierte al Agente Proveedor sobre la presencia de un usuario potencialmente interesado. Previamente, el Agente Central, localiza al usuario y realiza la correspondencia del perfil del agente usuario con el de cada proveedor. Luego, el proceso de negociación se inicia entre los clientes y proveedores. Ver Fig. 4.17

- En nuestra propuesta, SMA2, mientras el LA localiza e identifica al UA, el BA realiza la correspondencia entre el perfil de usuario y los servicios y productos que ofrecen los proveedores cercanos. Los comportamientos de los agentes evaluados en ambos sistemas incluyen: Oferta de Productos, el protocolo Contract Net y la Subasta Holandesa (*Dutch Auction*).

Los protocolos utilizados para evaluar ambas arquitecturas son, primero, un conjunto de protocolos inspirados en las especificaciones de FIPA: *Product's Offer, Contract Net y Dutch Auction*, y luego, un escenario ad-hoc de reputación, definido por nosotros y simulado para probar la versatilidad de la arquitectura propuesta para servicios de recomendación.

En las secciones siguientes, se presentan los resultados de los experimentos. Ambos sistemas se ejecutaron bajo las mismas condiciones: iguales parámetros, el mismo hardware y el mismo procedimiento para cada comportamiento.

El trabajo que se describe en (Joumaa et al., 2008) presenta un método cuantitativo para evaluar el SMA desde el punto de vista de la comunicación de los agentes. Este método sugiere una evaluación basada en el peso de la información que lleva cada mensaje.

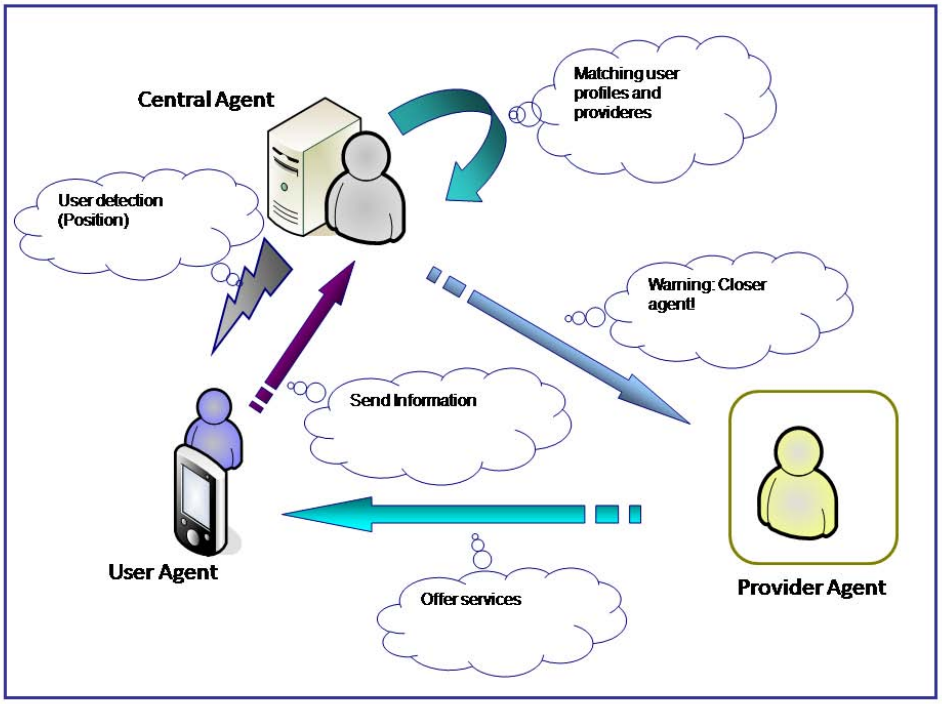

Figura 4.17: Arquitectura centralizada del SMA1
(Fuentes et al., 2007)

En la Fig. 4.18 se muestra la cantidad de mensajes enviados entre los agentes de cada SMA durante distintas situaciones cómo oferta de productos, implementación del protocolo Contract Net y un proceso de negociación implementando la Subasta Holandesa.

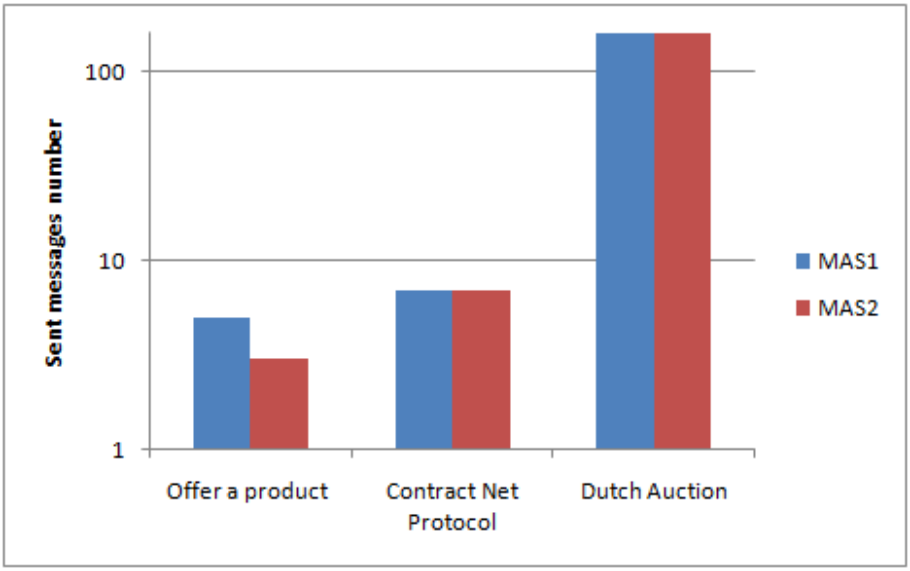

Figura 4.18: Cantidad de mensajes enviados en SMA1 y SMA2 por protocolo.

En el proceso de la Subasta Holandesa se utilizan datos estocásticos. Por ello, fue necesario realizar un promedio de los resultados de varias ejecuciones del mismo, dichos resultados se muestran en la Fig. 4.19. Mientras que para los otros dos procesos se muestra la cantidad real de los mensajes enviados durante la simulación.

En cuanto a los tipos de mensajes intercambiados durante esta simulación se utilizaron los siguientes: *Inform, Propose, Accept-Proposal, Call-for-Proposal (CFP), Request, Query-If y Reject-Proposal* (todos ellos implementados por FIPA).

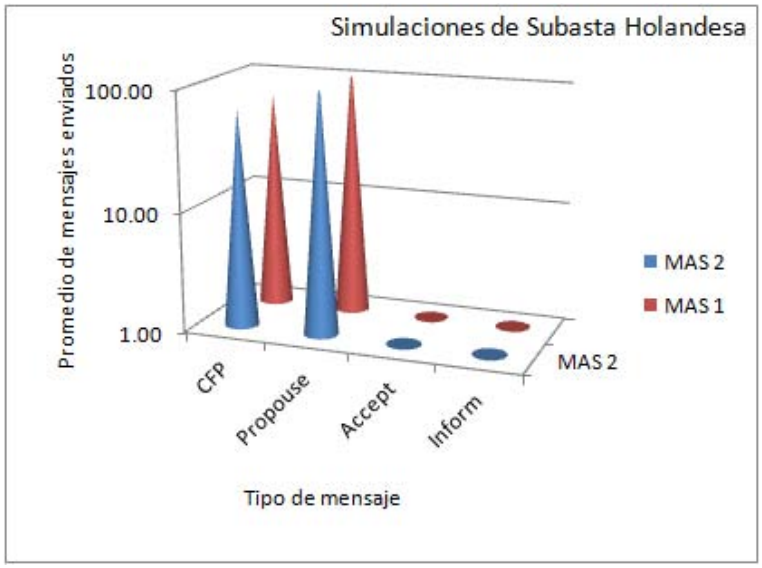

Figura 4.19: Proceso de la Subasta Holandesa para SMA1 y SMA2

Según la importancia del mensaje para el agente, se colocó un valor para cada tipo de mensaje según se propone en (Joumaa et al., 2008): *high* (1), *medium* $(0{,}75)$, *low* $(0{,}50)$ y *not pertinent* (0). Los resultados de esta evaluación se muestran en la Fig. 4.20. El criterio seguido para asignar estos pesos, a cada tipo de mensaje, fue el siguiente: en caso de que el comportamiento ejecutado trate de convencer al usuario respecto de la adquisición de algún producto o servicio, el máximo valor asignado será para el mensaje CFP (*call for propose*) o para el *Propose*, según el caso. De otra manera, si el interés sobre la negociación recae sobre el Agente Usuario, el máximo valor es asignado al resultado final de la operación, por ejemplo los mensajes: *Accept o Reject*. En las Tablas 4.20 y 4.21 se detallan los valores usados en el gráfico de la Fig. 4.20 donde se muestran los resultados del intercambio de mensajes en la ejecución de diferentes protocolos, con los pesos asignados. En dichas tablas se exponen por cada tipo de mensaje enviado (*Inform, Propose, Accept, CFP, Request, QueryIf y Reject Proposal*): la cantidad de mensajes enviados durante la simulación (cant. de msn.), el peso asignado y el resultado de la evaluación (Rtdo.). Si se analiza el ejemplo del mensaje tipo *Propose* para el protocolo de la Subasta Holandesa, como resultado de la evaluación para el SMA1 el resultado arrojado es de $132{,}8$, mientras que para el SMA2 el resultado obtenido de evaluación es de $130{,}16$, ambos con un peso de valor 1, diferenciándose en la cantidad de mensajes enviados. Tal como se observa en los gráficos, ambas arquitecturas poseen comportamientos muy similares (los resultados son prácticamente iguales), es decir, a pesar de incrementar la cantidad de

tipos de agentes en el sistema, la complejidad en la comunicación entre ellos no se ve afectada incrementalmente.

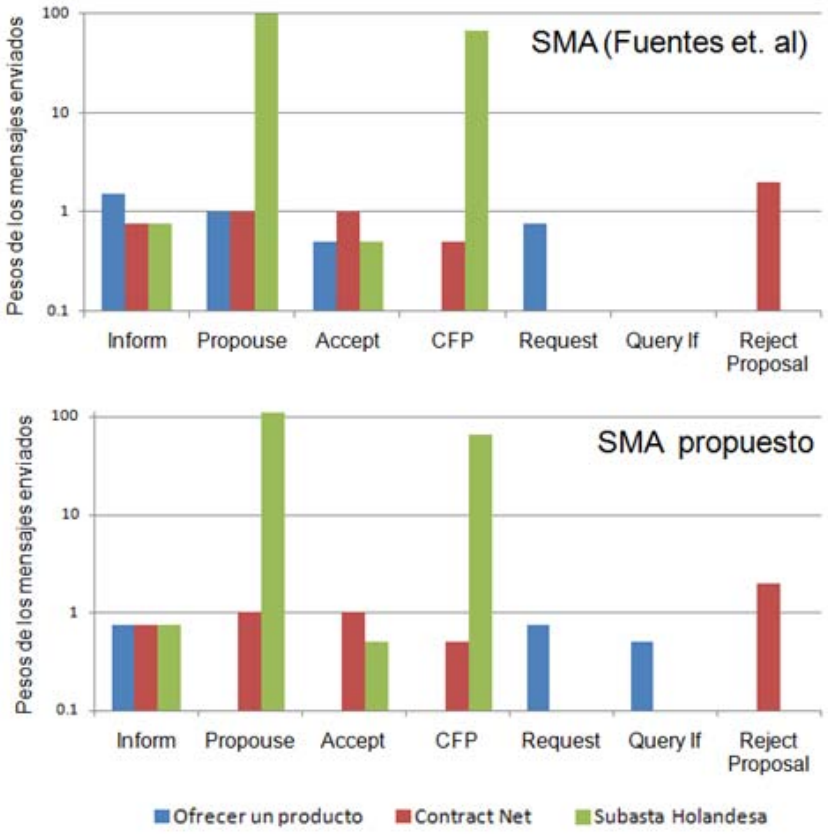

Figura 4.20: Comparación de los mensajes enviados por protocolo.

Tabla 4.20: Pesos asignados a cada tipo de mensaje en los protocolos de recomendación, dependiendo de su importancia: SMA1.

SMA 1	**Ofrecer un producto**			**Protocolor Contract Net**			**Subasta Holandesa**		
Tipo de msn	**Cant. msn**	**Pesos**	**Rtdo.**	**Cant. msn**	**Pesos**	**Rtdo.**	**Cant. msn**	**Pesos**	**Rtdo.**
Inform	2	0,75	0,75	1	0,75	0,75	1	0,75	0,75
Propose	1	1	0	2	0,5	1	132,80	1	132,8
Accept	1	0,5	0	1	1	1	1	0,5	0,5
CFP	0	0	0	1	0,5	0,5	66,40	1	66,4
Request	1	0,75	0,75	0	0	0	0	0	0
Query If	0	0,5	0,5	0	0	0	0	0	0
Reject Proposal	0	0,5	0	2	1	2	0	0,5	0

Tabla 4.21: Pesos asignados a cada tipo de mensaje en los protocolos recomendación, dependiendo de su importancia: SMA2.

SMA 2	Ofrecer un producto			Protocolo Contract Net			Subasta Holandesa		
Tipo de msn	**Cant. msn**	**Pesos**	**Rtdo.**	**Cant. msn**	**Pesos**	**Rtdo.**	**Cant. msn**	**Pesos**	**Rtdo.**
Inform	1	0,75	0,75	1	0,75	0,75	1	0,75	0,75
Propose	0	1	0	2	0,5	1	130,16	1	130,16
Accept	0	0,5	0	1	1	1	1	0,5	0,5
CFP	0	0	0	1	0,5	0,5	65,08	1	65,08
Request	1	0,75	0,75	0	0	0	0	0	0
Query If	1	0,5	0,5	0	0	0	0	0	0
Reject									
Reject Proposal	0	0,5	0	2	1	2	0	0,5	0

4.6.2 Evaluación centrada en Atributos

La evaluación propuesta por (Davidsson et al., 2006) consiste en tres dimensiones: el conjunto de posibles aplicaciones, el conjunto de posibles arquitecturas del SMA y el conjunto de atributos usados para evaluar la arquitectura. Es una buena práctica dividir la arquitectura en función de sus características a efectos de comprarla con otras arquitecturas existentes. A tales fines se considera un conjunto de atributos relacionados al rendimiento del sistema:

- Reactividad: ¿cuán rápido se reasignan los recursos cuando hay cambios en la demanda de los mismos?
- Balance de Carga: ¿cómo está distribuida la carga entre los recursos provistos?
- Equidad: ¿los clientes son tratados de la misma manera?
- Uso de los Recursos: ¿se utilizan los recursos tanto como sea posible?
- Capacidad de Respuesta: ¿cuánto tiempo le lleva a los clientes obtener una respuesta individual ante una solicitud?
- Sobrecarga de la Comunicación: ¿cuánta comunicación es necesaria para la asignación de recursos?
- Robustez: ¿qué tan vulnerable es el sistema ante fallos? Se considera que cuanto más centralizada es la arquitectura, más vulnerable es el sistema.
- Modificabilidad: ¿qué tan fácil es introducir cambios en el sistema?
- Escalabilidad: ¿qué tan bueno es el sistema en el manejo de un gran número de usuarios? Mejor escabilidad poseen aquellas arquitecturas distribuidas dado que la carga computacional se divide entre el número de computadoras, lo cuál a su vez minimiza el riesgo de que se produzcan cuellos de botellas.

A los fines de hacer una evaluación subjetiva se considera la arquitectura propuesta en comparación otras arquitecturas genéricas para AmI: CONSORT (Sashima et al., 2004b),

AmIRA (Berger et al., 2007) y OOA (Martin et al., 1995). La arquitectura propuesta provee de ofertas y servicios a los usuarios en base al perfil del usuario y el contexto en el que se encuentre, mientras que CONSORT solo considera aspectos relacionados a la situación actual del usuario (la distancia del usuario a un display y la posición de su rostro) ignorando sus preferencias. Por ende la arquitectura propuesta es más fuerte en cuanto al envío de información específica dirigida, sin gastar recursos demás, pues es probable que en CONSORT el agente usuario no tenga interés sobre la oferta enviada. Mientras que en la arquitectura propuesta la probabilidad de que el mensaje que recibe el agente usuario sea de su interés es mucho más alta gracias a la personalización de servicios. AMIRA es una arquitectura multi-agente diseñada en capas. Para razonar incorpora información sobre la situación actual del usuario pero también tiene en cuenta las situaciones pasadas a los fines de comprender que está sucediendo actualmente. Si bien considera información contextual esta arquitectura también ignora las preferencias de los usuarios, siendo una fortaleza este aspecto en la arquitectura propuesta. OOA considera las preferencias del usuario a través del uso de un agente facilitador, pero de manera enunciativa puesto que no existe una propuesta concreta de cómo hacerlo. Cómo puede apreciarse, en ninguno de los casos las arquitecturas hacen hincapié en la manera de gestionar el perfil de usuario, ni tampoco se incorpora aspectos relacionados a coaliciones sobre intereses comunes o recomendaciones entre usuarios. Por lo tanto si se evalúa el atributo "uso de los recursos", la arquitectura presentada explota al máximo los recursos disponibles en el entorno. Todas las arquitecturas al igual que la propuesta consideran la posición del usuario en el entorno para que los proveedores ofrezcan sus servicios.

Los autores de AMIRA afirman que su arquitectura maximiza aspectos de mantenimiento y flexibilidad puesto que cada capa depende de la capa inferior pero no de la superior. En el caso de nuestra arquitectura el mantenimiento no es un gran problema puesto que se conocen las interacciones entre los agentes, y los comportamientos que se gestionan por protocolo por lo tanto introducir un cambio en el SMA solo requerirá de modificaciones en el comportamiento de los agentes involucrados. En cuanto al balance de carga, el hecho de que en la arquitectura propuesta sea posible que los usuarios negocien y hagan acuerdos entre ellos, sin que intervenga otro tipo de agente, permite una mejor distribución de los procesos (a pesar de que en los casos expuestos hablamos de arquitecturas distribuidas).

La escalabilidad de la arquitectura se verifica en el trabajo de (Chmiel et al., 2005), en el cual se analiza la eficiencia de JADE en experimentos realizados con miles de agentes que intercambian una gran cantidad de mensajes.

En cuanto al dominio de aplicación de la arquitectura, se considera que todas las arquitecturas genéricas, incluída la propuesta, para entornos de AmI sirven para diversos dominios. CONSORTS es aplicable según (Nakashima, 2007) a entornos de demanda de autobuses, sistemas de navegación de automóviles, hogares inteligentes y museos. AMIRA fue validada en los siguientes escenarios: guía personal en la ciudad, hogares inteligentes, cuidado de la salud y megaciudades. En el anexo A se muestra el uso de la arquitectura propuesta en entornos inteligentes: feria tecnológica, aeropuerto, sitio de conferencias, la vida en la ciudad; mientras que a lo largo de la tesis se trabaja con un circuito turístico de ejemplo para demostrar la potencialidad de la arquitectura. De la misma manera podría usarse el framework para hogares inteligentes, una visita al museo, una universidad, un hospital, o cualquier otro dominio en el cual exista un conjunto de tareas que el usuario puede realizar, y una serie de servicios que los proveedores puedan ofrecer, ambos considerando que el sistema asiste al usuario, es su guía personal.

Davidsson (Davidsson et al., 2006) propone además la caracterización de la arquitectura contemplando: la topografía del sistema, el grado de mobilidad y dinámica de la comunicación, el grado de distribución del control y el grado de sincronización de la interacción. Nos enfocamos en los dos últimos aspectos. Se considera que agentes sincronizados son altamente sofisticados e interactúan solo en ocasiones especiales, mientras que los agentes asíncronos interactúan constantemente y de manera independiente a cuando interactúan otros agentes. Considerando entonces el método calificativo propuesto en (Davidsson et al., 2006) y de acuerdo a los resultados expuestos en las tablas presentadas en la sección anterior, se puede inferir que la arquitectura propuesta en SMA2 corresponde a un sistema distribuido asíncrono, en contraste con el SMA1 que se muestra como una arquitectura centralizada asíncrona. Por su parte, las características que resalta en el SMA2 son la reactividad, comunicación, robustez y escalabilidad, entre otras. Mientras que, el SMA1 es más robusto en la evaluación de la equidad y el equilibrio de carga. En este sentido, la diferencia entre los dos sistemas, al realizar la comparación del número de mensajes enviados es mínima. Por lo tanto, la arquitectura del SMA2, pese a tener mayor cantidad de agentes, es totalmente válida. Evaluando el grado de distribución de control, el SMA2 propuesto es mucho mejor dado que todas las tareas que el SMA1 carga sobre el agente central, aquí se encuentran distribuidas en los agentes localizados, intermediario y administrador de usuarios.

4.6.3 Evaluando un Escenario de Reputación adhoc

Para introducirnos en este experimento adicional, se evaluaron las interacciones sobre el modelo de confianza. La asignación de pesos a un escenario de reputación abre la posibilidad de comparar otros sistemas basados en contexto que incluyen modelos de confianza con nuestra arquitectura de agentes.

Por ejemplo, un cliente potencial, Juan Gallo, recibe una notificación sobre el agente proveedor Arbys. El agente de Juan se contacta con otros agentes registrados en el sistema. Este pide recomendaciones sobre la reputación de Arbys. Cada agente usuario envía a Juan un mensaje con su propia opinión sobre el proveedor en cuestión. Luego Juan, con toda esta información, arma su propia imagen respecto de la calidad de proveedor de Arbys, a los fines de decidir si tomar o no sus servicios.

A los efectos de asignar los pesos a cada tipo de mensaje, para el sistema de recomendación el máximo valor 1 se asigna a los mensajes de tipo inform y el valor 0.75 se asigna a los mensajes request, tal como se muestra en la Fig. 4.22. Los demás tipos de mensajes no se consideran relevantes para este caso.

Tabla 4.22: Pesos asignados a cada tipo de mensaje FIPA

Tipo de Mensaje	Pertinente	Peso
Inform	Si	1
Propose	No	0
Accept	No	0
CFP	Si	0
Request	Si	0.75
Query If	No	0
Reject Proposal	No	0

Se evalúan dos casos para el SMA2. Uno de ellos considera un solo agente usuario y cuatro recomendadores. El otro caso abarca a dos agentes y tres recomendadores. Ambos realizan solicitudes sobre un solo agente proveedor. La Tabla 4.23 muestra la cantidad de mensajes en cada ejecución de la simulación.

Las Fig. 4.21 y 4.22 muestran la captura de pantalla de ambos casos de recomendación en la ejecución de las simulaciones.

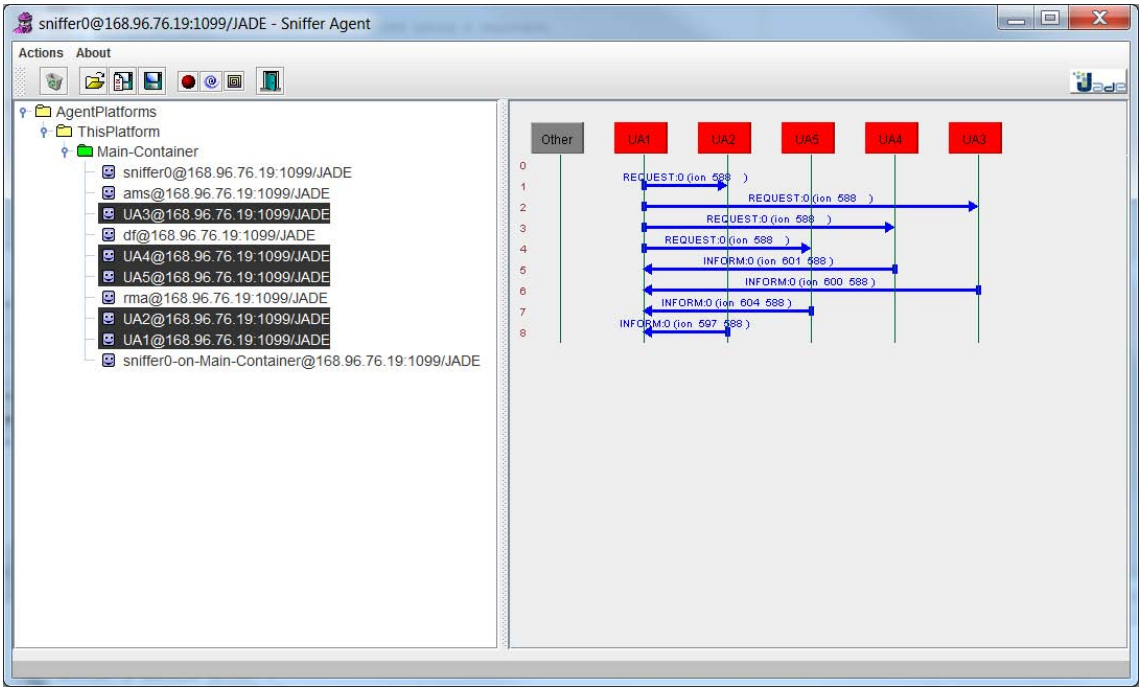

Figura 4.21: El UA1 solicita recomendaciones sobre un proveedor.

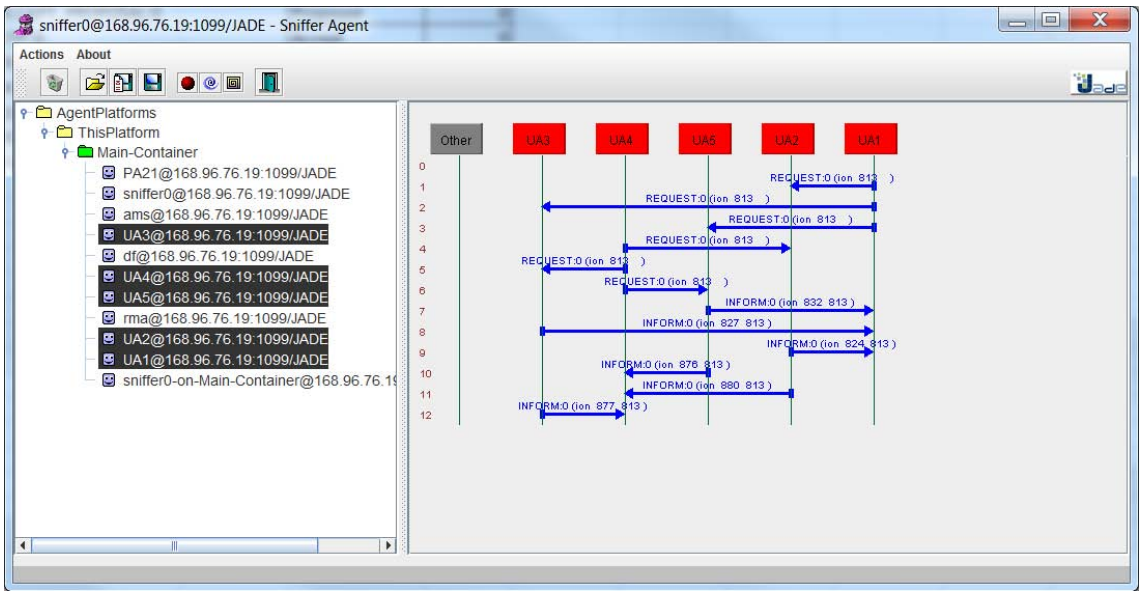

Figura 4.22: Los UA 1 y 4 consultan sobre la reputación de un proveedor.

Tabla 4.23: Pesos asignados a cada tipo de mensaje en los protocolos de recomendación, dependiendo de su importancia.

SMA 2	**1 solicitud a 4 usuarios**			**2 solicitudes a 3 usuarios**		
Tipo de mensaje	**Cantidad de mensajes**	**Pesos**	**Resultados**	**Cantidad de mensajes**	**Pesos**	**Resultados**
Inform	4	1	4	6	1	6
Propose	0	0	0	0	0	0
Accept	0	0	0	0	0	0
CFP	0	0.5	0	0	0.5	0
Request	4	0.75	3	6	0.75	4.5
Query If	0	0	0	0	0	0
Reject Proposal	0	0	0	0	0	0

4.7 Conclusiones del Diseño Metodológico y la Arquitectura Multi-Agentes

En este capítulo se presentó una arquitectura de agentes para entornos de Inteligencia Ambiental, mostrando en detalle el diseño del sistema. Este modelo se construyó siguiendo las metodologías GAIA y AUML. La arquitectura propuesta es flexible para adaptarse a cualquier dominio en entornos de la AmI, y queda abierto para la construcción de diferentes funcionalidades.

Los agentes del sistema se han implementado utilizando JADE, un paquete de JAVA específico para el desarrollo de agentes inteligentes. JADE implementa las especificaciones FIPA, permitiendo la comunicación entre ellos. Se han programado los comportamientos para los cinco tipos de agentes propuestos en el sistema: usuario, proveedor, intermediario, localizador y administrador de usuarios.

Además, se ha logrado una adecuada evaluación de dicha arquitectura siguiendo métodos diferentes de evaluación, pero sobre todo el descrito en (Joumaa et al., 2008). Dicho método consiste en contar la cantidad de mensajes intercambiados entre los agentes y asignarles un peso de acuerdo a la importancia que tiene para el agente. Este enfoque se implementó en dos arquitecturas diferentes que implementan los mismos comportamientos de agentes. Los resultados muestran la robustez y escalabilidad de la arquitectura propuesta, y su adaptación a modelos de confianza y reputación.

5

Estructuración de la Información

DENTRO de las metodologías para la construcción de un SMA, tal como se describirá en el siguiente capítulo, uno de los pasos a seguir es la definición de una ontología. La ontología representa la conexión entre la información física del mundo real y la información conceptual del mundo digital (Chen et al., 2003). Esto permite el intercambio del conocimiento entre los agentes en un determinado dominio y el razonamiento de los agentes sobre el contexto. El universo de las ontologías asigna el razonamiento de contexto a los agentes, un claro ejemplo de una ontología es el descrito en OCASS (Kim et al., 2006). En el caso de una aplicación de AmI, los agentes deben razonar acerca de los conocimientos en diferentes dominios de posible negociación (congresos, centros comerciales, ferias, etc.). Teniendo en cuenta las ontologías existentes, en este capítulo se propone una ontología genérica para representar el conocimiento de dominios heterogéneos. Nos enfocamos en un entorno inteligente dentro de cualquier dominio (hogar, oficina o espacios públicos), donde las personas deseen obtener ayuda de los agentes a la hora de ejecutar sus tareas de acuerdo a sus preferencias. Este es un punto destacado en el trabajo, dado que asumiendo un entorno cerrado y la posibilidad de obtener las preferencias del usuario, se le pueden realizar sugerencias sobre sus actividades, de acuerdo a sus intereses personales.

La comprensión del mensaje entre los agentes, el reconocimiento de las situaciones en las que se encuentra un usuario, el razonamiento sobre el contexto son tareas complejas. Es necesario elegir una tecnología apropiada o una combinación de las mismas, a los fines de incorporar información del entorno en el SMA. Ejemplos de tecnologías que permiten el razonamiento sobre el contexto son las basadas en lógica, como los motores de reglas (Chen et al., 2004b). Otros ejemplos incluyen técnicas de probabilidades como cadenas de Markov (Cilla et al., 2009) y Redes Bayesianas (Zhang and Ji, 2006). Finalmente, el enfoque basado en ontologías, en donde las situaciones dentro del contexto se definen como conceptos complejos y que con el uso de un razonador se utilizan para deducir situaciones según el estado actual del mundo. En esta tesis se opta por combinar el uso de ontologías con bases de datos espacio temporales (Venturini et al., 2008b). Si bien no se realiza el cálculo del costo computacional que conlleva el uso de ontologías, en el trabajo (Turhan et al., 2006) queda comprobado el alto rendimiento de los razonadores sobre las mismas. Por otra parte, el uso de JADE para la construcción del sistema multi-agente permite la incorporación de ontologías, y existe plugins como Beanynizer[1]

[1] http://jadex.informatik.uni-hamburg.de/docs/jadex-0.94x/toolguide/tools.beanynizer.html

que permiten al desarrollador la traducción directa de la ontología a JADE. A su vez, JADE es muy eficiente y escalable en experimentos realizados con miles de agentes que intercambian miles y miles de mensajes, tal como se expone en (Chmiel et al., 2005)

En la segunda parte de este capítulo, se aplican bases de datos espacio temporales a la manipulación de los datos de contexto, dando especial importancia a los datos de localización física del usuario, y mostrando cómo se trabaja en relación a la ontología. Ambas hacen posible la comunicación entre los agentes de la arquitectura propuesta en el capítulo (4). En particular, la base de datos espacio-temporal hace posible que tanto el agente localizador como el agente intermediario tengan un registro de los movimientos de los agentes usuarios en un lugar e instante de tiempo dado.

5.1 Ontología Propuesta

El uso de una ontología viene dado por la necesidad de representar el conocimiento de manera que pueda ser compartido por agentes de distintas plataformas. Una ontología de contexto define un vocabulario común para compartir la información de contexto en un dominio de inteligencia ambiental, e incluye la definición de conceptos básicos en el dominio y la relación entre ellos (Gu et al., 2004a). En nuestro sistema, usamos una ontología para facilitar la comunicación entre los agentes de la arquitectura propuesta en el capítulo anterior. La definición de la ontología ontología propuesta sigue una metodología genérica la cual combina las estrategias *Top-Down* y *Bottom-Up* para la extracción de conceptos tal como lo hacen los autores en (Fuentes et al., 2006a), y, además, se puso en práctica la reutilización de ontologías existentes. Una ontología que soporte la arquitectura multi-agente que se propuso anteriormente, debe cubrir la necesidad de información de contexto en diferentes dominios, y con particular interés aspectos espacio-temporales y características de los perfiles de usuarios.

Como antecedentes a este diseño se toman las ontologías presentadas en (Gu et al., 2004b), trabajo en el que se realiza la definición de una ontología de alto nivel, que captura el conocimiento de contexto acerca del mundo físico, y la de la publicación de (Fuentes et al., 2006a) para representar el conocimiento de dominios heterogéneos. En varias investigaciones sobre ontologías, surgen dos conceptos importantes para representar el perfil del usuario. En (FIPA, 2000) es representado mediante dos conceptos: *explicit preferences* (datos personales del usuario) e *implicit preferences* (aprendizaje de patrones a través del comportamiento del usuario). A su vez en (Spanoudakis and Moraitis, 2006) los datos del perfil se modelan usando las definiciones de datos estáticos (definidos por el usuario) y datos dinámicos (comportamiento aprendido). En la ontología que propone el tesista, se utilizan los conceptos *static preference y dynamic preference*, tomados de (Kwon et al., 2006), para la representación de las preferencias del usuario, y se propone la creación del concepto *negotiating parameters* a los fines de representar la información necesaria al momento de negociar, como por ejemplo: forma de pago, número de tarjeta de crédito y cantidad de cuotas, de acuerdo al dominio en el que nos encontremos.

Finalmente, luego de analizar la reutilización de las ontologías plasmadas en diferentes trabajos de investigación, la ontología que se define, para múltiples escenarios, se compone de los siguientes conceptos:

- *Framework*: define el dominio de aplicación dentro de la ontología genérica.

- *Participant* (participante): representa una entidad que va ha realizar la negociación, dicha entidad puede ser una persona o una compañía.
- *Location* (localización): representa la coordenada (x,y) de cada entidad en el sistema, ya sea una persona, un objeto o un lugar de interés.
- *Spatial Region* (región especial): es la representación del espacio, y está compuesta a la vez de más clases para la representación del dominio: país, ciudad, zona, edificio, piso, sector.
- *Place* (lugar): es un punto en el que se proveen servicios.
- *Temporal Region* (región temporal): recoge la información de hora y fecha cuando un usuario se encuentra en cualquier ubicación dentro de la region espacial, para representar el instante de tiempo.
- *Services* (servicios): servicios que se pueden proveer.
- *Products* (productos): productos como objetos de negociación o recomendación.
- *Devices* (dispositivos): este concepto sirve para saber de que unidad móvil dispone el usuario y poder adaptar las interfaces de presentación de la información según sus preferencias en cualquier lugar y momento.
- *Static Preferences* (preferencias estáticas): son las preferencias definidas por el usuario o introducidas por él en el sistema, por ejemplo su película favorita.
- *Dynamical Preferences* (preferencias dinámicas): son las preferencias que se aprenden automáticamente, por ejemplo, cuáles son los restaurantes a los que un usuario asiste frecuentemente o qué actividades realiza durante un día.
- *Negotiating Parameters* (parámetros de negociación): recoge la información necesaria al momento de llevar a cabo la negociación. Por ejemplo: forma de pago, número de la tarjeta de crédito, cantidad de cuotas, etc, de acuerdo al dominio.

La adaptación de la ontología para soportar el tratamiento de coaliciones fue tomada de (Oravec et al., 2007). El la Fig. 5.2 se muestran los conceptos que representan el problema de las coaliciones. *Coalition* es una clase abstracta de la cual se desprende el concepto *Problem*. Este concepto incluye el atributo *Objetive*, y a su vez, derivan de él dos conceptos más: *Context* y *Action*. El concepto *Context* se refiere a la coalición que resuelve el problema usando un plan de acción definido en el concepto *Action*.

Por otro lado, se define una serie de acciones de los agentes, para hacer posible la negociación y para facilitar la implementación de las performativas de FIPA. La ontología también contiene las acciones del agente, las cuales cuentan cómo un agente maneja un mensaje justo en el momento en que lo recibe. Por ejemplo: (Recommend(Person(John))) lo que equivale a "Se recomienda a la persona John". Para ser más precisos, una acción de un agente forma parte del intercambio de mensajes entre los agentes (Ahlbrecht and Bothe, 2005). Estas acciones para el sistema propuesto, en el capítulo anterior, son:

- *Recommend* (recomendar): un agente puede hacer una recomendación sobre un objeto o lugar a otro agente. Es necesario representar en esta acción a los dos agentes intervinientes y a la entidad que se recomienda.
- Register (registrar): cada agente debe ser registrado en el sistema para poder ser lanzado. Esta acción solo necesita de la identificación del agente.
- *Offer* (ofertar): esta acción es utilizada por los agentes proveedores de servicios para ofrecer productos o servicios a los agentes usuarios. Para instanciar este concepto son necesarios ciertos parámetros: la lista de agentes usuarios, el producto o servicio que se ofrece, con un conjunto de características como precio, cantidad de la oferta, etc, y finalmente, quien es el proveedor del servicio.
- *Negotiate* (negociar): los agentes usuarios o proveedores pueden usar esta acción una vez aceptada una oferta para realizar consultas sobre la acción de negociar en sí, como ser cuál es el método de pago o si le pueden hacer una rebaja al comprar por grandes cantidades.
- *Communicate* (comunicar): acción utilizada por un agente para enviar una comunicación a otro.

El uso de una acción se muestra en el siguiente código, el cual implementa una acción dentro de la Subasta Holandesa:

```
NegotiationParameters myReceivedNegotiation
Product sArt = new Product();
//Offer a Picture in an auction
sArt.setProductID("Mona Lisa");
sArt.setPrice(iAuctionProductPrice);
system.out.println("Provider auction: " + sArt.getProductID() +"  " + sArt.getPrice());
Offer myOffer = new Offer();
myOffer.setHasProduct(sArt);
```

Otro item que incluye la ontología son los predicados (*predicate*), los cuales son términos que representan expresiones que pueden ser verdaderas o falsas acerca del estado del entorno. Por ejemplo: `(Located(person(Mary),location(2,4)))`.

Finalmente, podemos observar en la Fig. 5.1 la ontología resultante, utilizando los conceptos definidos anteriormente para nuestro sistema multi-agente. Y en la Fig. 5.2, se muestra una extensión para representar coaliciones. Esta ontología fue construída utilizando Protégé [2].

En la tabla a continuación (Tabla 5.1) se muestra como distintos escenarios se adaptan a los conceptos que la componen, exponiendo los más representativos, puesto que los otros son conceptos genéricos, y utilizando los mismos escenarios que se mostraron en la implementación del sistema multi-agente (entorno de conferencias, eTurism, feria tecnológica SIMO) y un escenario en la universidad.

[2] "http://protege.stanford.edu/"

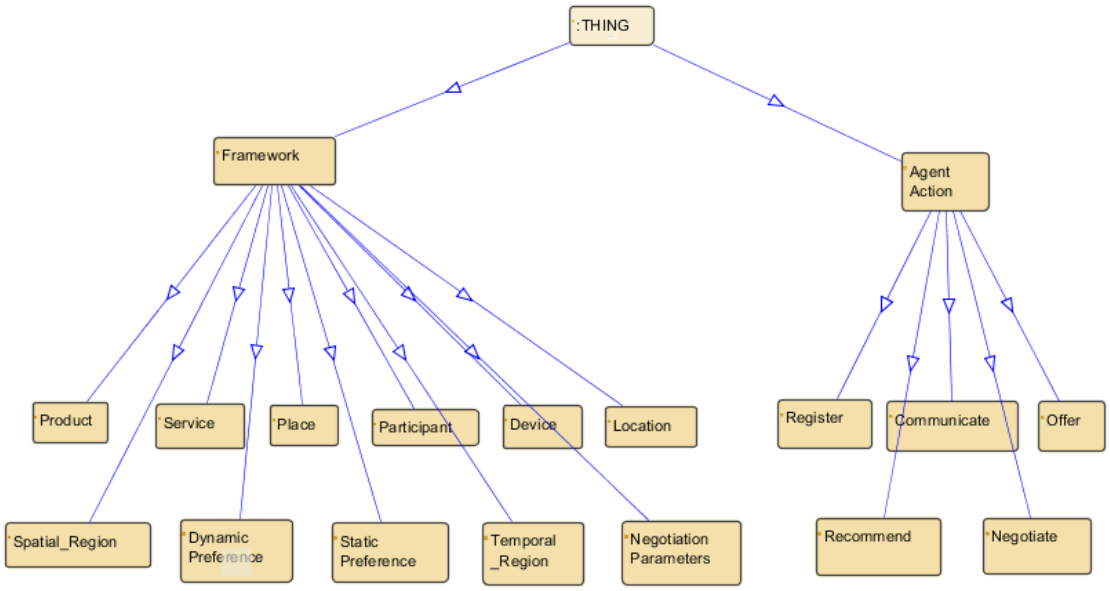

Figura 5.1: Ontología propuesta para el SMA

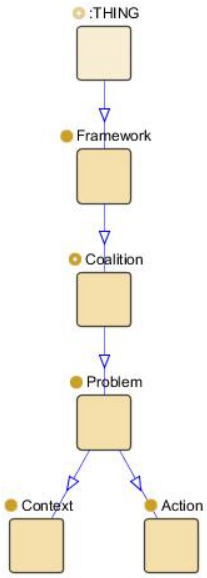

Figura 5.2: Extensión de la ontología para la representación de coaliciones.

En resumen, la ontología propuesta está compuesta por 16 conceptos y 5 acciones, cuyos atributos se detallan en en Anexo B y los ejecutable (.class o .jar) están disponibles via web para cualquiera que desee usarlos.

Tabla 5.1: Adaptación de la ontología para la arquitectura multi-agentes a diferentes escenarios

	Conferencias	**eTourism**	**Feria SIMO**	**Universidad**
Participantes	Asistente, Expositor	Turista, Comerciante	Visitante, Vendedor,Ponente	Alumno, Docente, Personal administ. y de servicio
Productos	Libros publicaciones	Ticket de museo, Menús de Restaurantes	Móvil, PDA, IPhone	
Servicios	Demos Poster Exposiciones	Tomar un taxi Citytour	Planes de telefonía móvil	Calificaciones Bibliografía Horarios
Ubicación	Prox. a la sala de posters, a la de conferencias	Prox. a centros históricos, estaciones de medios de transporte, restaurante	Posición de los stands en los que el usuario tiene mayor interés	Localización de las aulas a las que debe asistir un alumno
Región espacial	Centro de conferencias Salas	Ciudad, Centro comercial, centro histórico, camping	Recinto ferial pabellones, stands	Edificio, piso pasillo, aula
Lugares	Sala A	Restaurante La Espuela	Satand Nokia	Aula 1.1.A.21

5.2 Aplicación de Bases de Datos Espacio-Temporales

Tanto los valores de las preferencias del usuario, como la localización de los mismos, o cualquier información de contexto, debe ser almacenada en una base de conocimiento. El usuario puede llevar un registro de la información espacio-temporal en tablas de su perfil público, como se verá luego en el capítulo 7. Por su parte la información manipulada por los agentes intermediario y localizador podría almacenarse en una base de datos (BD) dentro de un servidor. En la siguiente sección se propone el uso de un sistema gestor de bases de datos (SGBD) para almacenar dicha información, pero teniendo especial cuidado en la gestión de los datos espacio-temporales, dada la relevancia y riqueza que representan en nuestro sistema multi-agente. Como se ha expuesto en el capítulo sobre la arquitectura del sistema, de acuerdo a la ubicación del usuario, estaremos en condiciones de recoger información sobre el dominio, como así también ofrecerle servicios, sin perder de vista sus intereses.

5.2.1 Base de Datos Espacio-Temporal para Información de Contexto

De acuerdo a Schilit (Schilit et al., 1994) es necesario realizar un buen análisis de contexto, en el cual es fundamental responder a dos preguntas: *dónde y cuándo*. Respondiendo a la pregunta *dónde*, una dimensión espacial en (Hu et al., 2006) se clasifica como datos de posición geográfica (localización), datos de posición espacio-temporal y datos de posición semántica. En (Small et al., 2001) se define a la "localización" como un ejemplo de contexto, es decir, la información sobre las personas o dispositivos que pueden ser utilizados para modificar la manera en la que el sistema provee los servicios a la comunidad de usuarios. Mientras tanto, cuando hablamos sobre las variables de tiempo, respondiendo al *cuándo*, podemos realizar una

clasificación de acuerdo a:

- La duración:
  - Instante: ha ocurrido algo
  - Intervalo: algo ocurrido se mantiene
- En cuanto a su dimensión:
  - Tiempo válido: se considera como el tiempo durante el cual este hecho es considerado cierto en el mundo real.
  - Tiempo de transacción: se considera el tiempo como el momento en que se ha incluido el hecho en la base de datos.

Los datos espaciales y temporales se combinan porque en muchos casos se necesita almacenar la evolución temporal de las entidades relacionadas espacialmente (Bertino et al., 2005). En este sentido, y de acuerdo a nuestra ontología descripta en la sección anterior, vamos a mostrar cómo se puede diseñar una base de datos para manejar el conocimiento sobre la localización y el perfil de usuario, con el objetivo de aprender las preferencias del usuario y su comportamiento, datos que luego serán utilizados para realizar ofertas al usuario. Teniendo en cuenta los conceptos definidos en la ontología: localización, región espacial y región temporal, se opta por el uso de MySQL como herramienta para modela la información a almacenar. MySQL provee de un extensión espacial [3] particular que se puede utilizar para declarar los datos espaciales. Algunos tipos de datos en relación a la dimensión espacial son: *Geometry, Point, LineString y Polygon*. Al mismo tiempo, esta herramienta otorga un conjunto de funciones para manipular la información espacial, como *Contains(g1,g2)* - donde $g1$ y $g2$ representan respectivamente a la geometría 1 y a la geometría 2 - para saber por ejemplo si g1 se encuentra dentro de $g2$, o *Distance(g1,g2)*, que devuelve el número que representa la distancia más corta entre los puntos de ambas geometrías. En cuanto a la gestión del tiempo, esta herramienta posee los tipos de datos tradicionales (*time, date, datetime*) aunque agrega algunas funcionalidades para manejar intervalos de tiempo. Teniendo en cuenta la información anterior, diseñamos el modelo de datos para soportar la información espacial en relación a las actividades del usuario y las posibilidades de realizar ofertas de productos y servicios. Este modelo se muestra en el diagrama de la Fig. 5.3. Las entidades que hemos seleccionado de acuerdo a nuestra ontología, son: *spatial_temporal, areas, segments, locations y places* para representar los datos espaciales y de localización. Por otro lado, para representar los eventos de los usuarios, hemos creado la Tabla *activities*, la cual incluye datos de las tablas: *participant, temporal region, product or services*.

Por otro lado, se necesita representar las preferencias del usuario dentro del SGBD. Con este propósito, podemos utilizar la técnica desarrollada en (Stefanidis, 2005). En este trabajo, las preferencias son modeladas en cubos de datos y el valor de cada preferencia se calcula teniendo en cuenta al usuario, el lugar y la característica a evaluar. El número de cubos de datos es igual al número de características que se desean evaluar (por ejemplo, costo, calidad y distancia). En el diagrama anterior, las entidades *Dynamic_Preferences y Static_Preferences*,

[3] "http://dev.mysql.com/doc/refman/6.0/en/spatial-extensions.html"

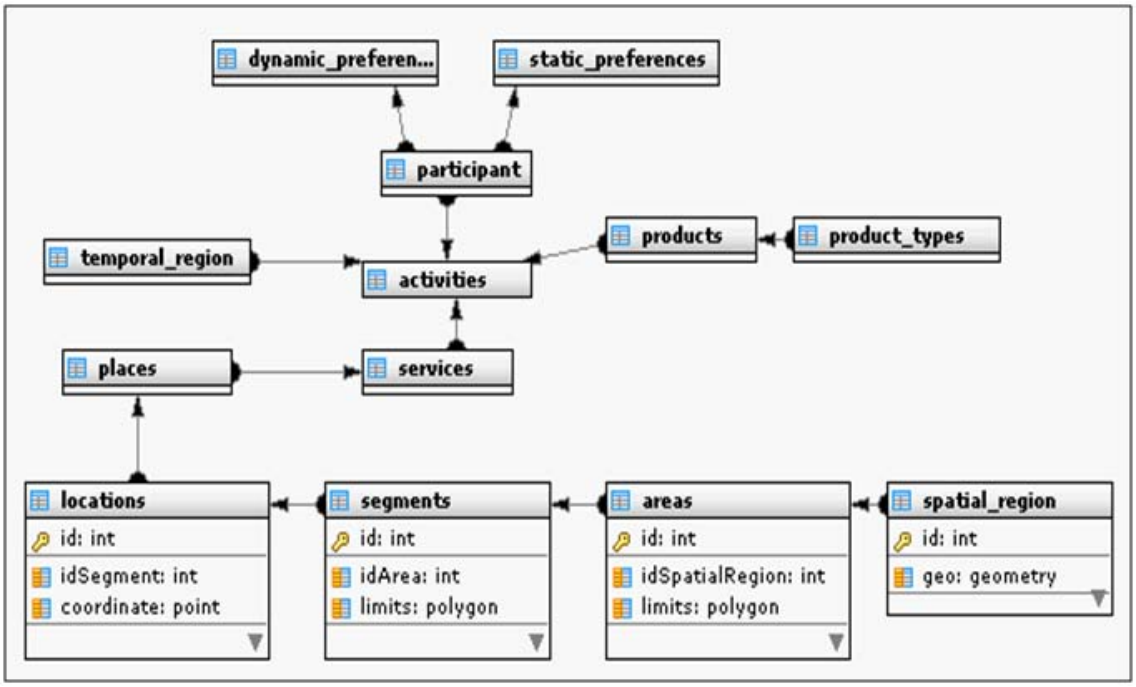

Figura 5.3: Diagrama de Base de Datos Espacio Temporal para sistemas basados en contexto y localización

almacenan el valor por cada usuario sobre cada característica. Mediante el uso de la técnica de hipercubos, podemos calcular las preferencias reales sobre cualquier aspecto del contexto, pero no es de incumbencia en esta tesis.

5.2.2 Adaptación de Información Espacio-Temporal a un Escenario Real

Para evaluar el modelo, se toma como ejemplo el escenario turístico, descripto en en el capítulo anterior. Se supone la presencia de un turista llamado Juan que va a encontrarse con otro turista, Pedro, en el centro comercial de la ciudad. *Planean almorzar juntos y luego visitar el Museo de Antropología que se encuentra a 600 mts. Puesto que no se conocen, Juan decide usar su asistente personal para que le consulte al asistente digital de Pedro sobre sus preferencias para el almuerzo, y para que luego el agente de localización pueda dar precisión sobre el sitio para la comida.* En el mapa de la Fig. 5.4 se observan ambos sitios remarcados con rectángulos verdes, y con una línea del mismo color el recorrido entre los dos puntos.

Dado el predicado `(Located(person(Pedro),location(-24.7808091, -65.4021304))`, los conceptos utilizados de la ontología son: `Predicate=located, person=Pedro and location= (-24.7808091, -65.4021304)`. Con este predicado es posible saber si Pedro se encuentra dentro del centro comercial. Puesto que la información además se almacena en la base de datos espacio-temporal, es posible realizar consultas sobre actividades pasadas de los usuarios registrados en el sistema. Veamos los siguientes casos.

Caso 1: Como se utiliza esta estructura de datos si se desea saber dónde estuvo Pedro en la

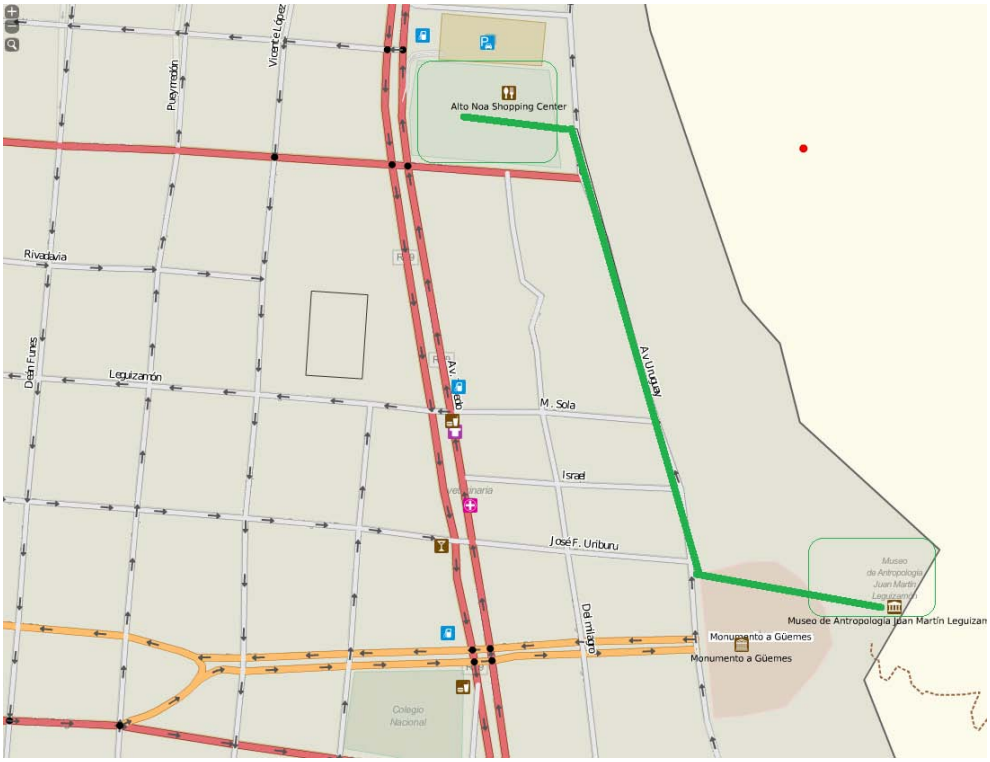

Figura 5.4: Región especial y localización en un circuito turístico.

mañana. Para representar la pregunta se puede utilizar la ontología de esta manera: (Located (person(Pedro), TemporalRegion(this morning))). Tenemos el conocimiento almacenado en la base de datos, por lo tanto, podemos ejecutar una consulta al sistema gestor de base de datos la cual retorne el conocimiento: "El realizó el circuito histórico".

```
SELECT activity.*
FROM . . .
WHERE person.name = 'Pedro'
AND temporal_region.time='during morning';
```

Caso 2: Consultar sobre si se encuentran en la misma región espacial el museo y el centro comercial. Para responder a esta pregunta, se puede realizar otra consulta, por ejemplo: "¿el museo de Antropología está cerca del centro comercial?", lo cual se representa en la ontología como: *(AreClosed(Place1(museo_Antropologia), Place2(commercial_center)))*. Utilizando una consulta a la base de datos espacio-temporal:

```
Set @g1 = 'Polygon((-24.7863838 -65.3975242, -24.7863848 -65.3975232), (-24.7863848
-65.3975252, -24.7863852 -65.3975242))'; *ubicación del museo de Antropología*
Set @g2 = 'Geometry((-24.7808081,-65.4021304),(-24.7808091,-65.4021314)'; *ubicación
del centro comercial*
SELECT Places.Name, Location.Coordinate
FROM . . .
WHERE Distance(g1,g2);
```

5.3 Conclusiones de la Estructuración de la Información

En este capítulo se ha construido una ontología para dar soporte a la comunicación entre las diferentes entidades que componen el sistema. La ontología fue construida en Protégé y contiene 16 conceptos y 5 acciones. A partir de esta estructura de datos, los agentes presentados en el capítulo anterior, podrán comunicarse entre ellos y razonar sobre el contexto. Se ha mostrado de qué manera la ontología permite el manejo de la formación de coaliciones.

Se ha propuesto la gestión de las preferencias del usuario, como así también de toda la información de contexto relevante en el sistema (específicamente la localización), a través de un sistema gestor de bases de datos, con soporte específico para datos del tipo espacio-temporal.

Se estudió de manera paralela cómo se utiliza la ontología en relación a la base del conocimiento, haciendo hincapié en la información de localización, mediante el uso de tipos de datos espaciales que permiten definir a priori una geometría, sobre la cual se realizarán consultas posteriormente, es decir, inferencias razonando sobre el contexto.

6

Adquisición de Perfiles de Usuario

El enfoque de esta tesis es el estudio de entornos inteligentes en cualquier dominio (el hogar, la oficina o espacios públicos), en donde las personas puedan estar interesada en obtener ayuda a través de agentes inteligentes con el objetivo de llevar adelante sus tareas de acuerdo a sus preferencias. Este es un punto importante, pues asumiendo un entorno cerrado y la posibilidad de adquirir las preferencias del usuario, es posible realizar sugerencias al usuario respecto de sus actividades cotidianas.

El sistema utiliza técnicas de computación evolutiva para realizar el aprendizaje automático de las preferencias del usuario, tal como se explica en el estado del arte en la sección 2.3. Con el uso de reglas de heurísticas a priori es posible editar secuencias de información en las que el usuario tiene interés. Después del análisis de los resultados, el sistema busca los intereses del usuario y detecta nuevos intereses con esta información. El uso de heurísticas de algoritmos genéticos permite conseguir que las preferencias del usuario (ya que es información privada del usuario, para obtener lo que quieren y para ofrecer servicios de su interés) y generar una lista de tareas priorizadas de acuerdo a sus intereses para ayudar en la toma de decisiones. Para la aplicación de AG, se deben evaluar las posibles soluciones, así que hemos desarrollado un método de evaluación que minimiza los costes de evaluación debido a que una generación completa tiene que ser evaluada mediante la elección del usuario.

En esta tesis se combinan técnicas de algoritmos genéticos (en adelante AG) para predecir las preferencias del usuario de una manera similar a (Chen et al., 2004c) y con el objetivo final de obtener intereses inmediatos en un entorno AmI como el perseguido por (Nijholt, 2007). Estos pueden ser considerados los más cercanos precedentes en el enfoque innovador de este capítulo.

Algunos de los conceptos descritos en la ontología del SMA, en el capítulo anterior (cap. 5), son las preferencias estáticas y dinámicas del usuario, que le sirven como referencia a la hora de tomar una decisión respecto de la acción a realizar. En este apartado se aborda la temática de adquisición de los perfiles de usuarios, con el fin de obtener las preferencias del mismo de manera dinámica.

Cada uno de nosotros juega distintos roles en la vida cotidiana, que se pueden resumir en: el individuo privado, el profesional y el participante público. Wright describe cada uno de ellos (Wright et al., 2008):

- El rol de individuo privado es aquel que juega un usuario cuando está atento a sus pasatiempos y/o a sus responsabilidades como padre o miembro de la familia. Las personas en este rol tienen requerimientos especiales que involucran entretenimientos, ocio, y otras actividades como ir de compras, e intereses particulares en aspectos de salud y educación.

- El rol profesional se lleva adelante en las acciones laborales del día a día. Las habilidades profesionales están en la naturaleza de la persona de poder comunicarse con sus pares de oficina o en cualquier lugar del mundo, en tener infinidad de información a manipular y en su capacidad innata de tomar decisiones. Cada una de estas habilidades puede ser altamente explotada con el uso de distintas herramientas de la AmI.

- El rol público, es decir las necesidades de las personas como ciudadanos. Los ciudadanos demandan seguridad, privacidad, confianza (o confiabilidad) de acuerdo a la situación en la que se encuentren. Muchas veces ellos no son consientes de las computadoras, redes y sistemas de monitoreo que los rodean. Por lo tanto, se les debe advertir sobre la posibilidad de que algunas entidades pueden utilizar sus datos para hacer ingeniería social (muy de moda actualmente a través de las redes sociales) o que de cierta manera alguien está amenazando su espacio privado. Es por ello, que el usuario tiene que tener la posibilidad de decidir el nivel de privacidad, seguridad, confianza y anonimato en cada circunstancia.

La AmI debe cubrir cuidadosamente estos roles, que abarcan gran cantidad y variadas ramas de investigación en el ámbito científico. Aspectos relevantes a la hora de crear perfiles, en entornos de Inteligencia Ambiental, son los que se detallan a continuación, expuestos en (Wright et al., 2008):

1. Factores humanos y de seguridad: factores como la exaltación pueden hacer que los individuos se olviden de corregirlos (correr la puerta, revisar los accesos a servicios online). Los errores humanos repetidos constituyen la mayor debilidad en cuanto a seguridad.

2. Pérdida de control: la AmI no puede saberlo todo por nosotros. Algunas actividades necesitan de nuestra confirmación.

3. Levantar sospechas: las debilidades al construir un perfil de usuario llevan a equivocaciones relevantes. Al reaccionar de forma irracional en el manejo de la información, se deja abierto un campo para levantar sospechas.

4. Exclusión: no todos los servicios de la AmI estarían disponibles para todo el mundo, dependerá mucho de la tecnología adquirida por el usuario. Por ejemplo, puede ser que una persona no cuente con dispositivo de localización.

El objetivo de este capítulo es la adquisición del perfil de usuario de manera no intrusiva, motivado por el hecho de que el usuario es resistente a otorgar su información privada, pero esa información es muy rica para el sistema, dado que permite a los agentes proveedores ofrecer servicios personalizados a los clientes.

A continuación se exponen algunas técnicas de adquisición de perfiles y preferencias, para posteriormente presentar un algoritmo de predicción de preferencias del usuario basada en

características, a través del uso de heurísticas de algoritmos genéticos (AG), con el fin de ofrecer al usuario servicios apropiados; además, el algoritmo hace posible la obtención de una lista de tareas priorizadas, con el fin de asistirle también, en la toma de decisiones.

6.1 Administración de Perfiles

Las personas tienen distintos intereses, los cuales pueden ser a corto, mediano o largo plazo. Nuestro objetivo es trabajar con los intereses del usuario a corto plazo, y por lo tanto es necesario tener en cuenta dos tipos de intereses: uno acerca del dominio, y el otro acerca de los intereses del usuario como se describe en (Nijholt, 2007). En este sentido, el sistema puede realizar sugerencias al usuario sobre las tareas a ejecutar, de acuerdo a sus intereses inmediatos del pasado. Por lo tanto, es posible utilizar los algoritmos genéticos para obtener una lista de tareas de acuerdo a sus preferencias para que el usuario elija cuales desea ejecutar. Existen varios caminos de adquisición del conocimiento. Una posibilidad es preguntar al usuario acerca de los productos y servicios sobre los que tiene actualmente. Otra manera, es utilizando cuestionarios para obtener de manera indirecta las preferencias del usuario. Esto permite el aprendizaje sobre las características del usuario (personalidad, costumbres, etc.) y su relación con las decisiones del usuario en algunas circunstancias. También es denominado perfil del usuario "basado en conocimiento", el cual es estático, intrusivo y consume su tiempo. Mientras que el perfil de usuario "basado en comportamiento" utiliza el propio comportamiento del usuario para construir y mejorar el perfil dinámicamente (Fuentes et al., 2006b). Por su parte, de acuerdo a (Samper Márquez, 2005), los perfiles pueden crearse a partir de diferentes métodos, tal como se indico en el estado del arte, capítulo 2:

- Explícito: los datos son introducidos directamente por el usuario.
- Colaborativo: el perfil se crea y modifica a partir de su interacción colaborativa con otros perfiles con los que se relaciona, recurriendo al conocimiento específico del dominio y heurísticas inteligentes.
- Implícito: se crea y modifica el perfil automáticamente, recurriendo a técnicas de inteligencia artificial.

Es común, que el usuario desee obtener una respuesta rápida del sistema, y, por lo tanto, que considere una pérdida de tiempo al hecho de completar un formulario con datos personales. Por este motivo, en la siguiente sección del trabajo, se trata la obtención del perfil de usuario de manera implícita, observando sus comportamientos.

6.2 Aprendizaje de Preferencias usando Algoritmos Genéticos

Un algoritmo genético (AG) es un proceso iterativo que consiste en generar de forma aleatoria un grupo de posibles soluciones, llamados población, luego darles un valor de aptitud con una función de evaluación, posteriormente seleccionarlos y cruzarlos de acuerdo a los operadores de selección y cruce definidos para el AG, de tal forma que se creen nuevas generaciones de

individuos más aptos, hasta encontrar una solución aceptable en la optimización del problema (Roa et al., 2005).

Los AG tienen ventajas indudables (Renders and Bersini., 1994), en especial la capacidad de aproximación de un problema real por el hallazgo de las regiones más prometedoras de todo el espacio de búsqueda. Los AG pueden ser más poderoso cuando encajan en un método heurístico tradicional (como ser una búsqueda local) (Chen et al., 2004c). Entonces los AG se usan para la búsqueda global en una unidad, y el método heurístico ejecuta la búsqueda local en el cromosoma.

Los intereses personales se modelan y luego se codifican en un cromosoma. La operación de cruce permite el descubrimiento de nuevos intereses, gracias a la búsqueda de los mejores intereses de los padres (Chen et al., 2004c).

La operación de mutación es también un buen método para predecir nuevos intereses (Chen et al., 2004c). El sistema utiliza reglas de heurísticas a priori para modificar la secuencia de información en la cual está interesado el usuario. Y tras el análisis de lo encontrado, el sistema encuentra los intereses mas relacionados al usuario, luego detecta nuevos intereses usando esta información. Después de la generación de nuevos hijos a través de la operación genética, el sistema retorna los mejores puntos rápidamente a través de la búsqueda heurística.

Las dos operaciones descritas anteriormente son las teorías principales de los algoritmos genéticos. Los AG no solo optimizan el sistema con precisión y en tiempo real, sino que también encuentran los intereses de los usuarios de acuerdo a alguna información existente.

6.3 Predicción de Preferencias basada en Características

Dentro del sistema multi-agente, el agente intermediario (*Broker Agent*) puede proporcionar una lista de tareas al agente usuario (*User Agent*) de acuerdo a sus preferencias (ver Fig. 6.1).

Tomamos como ejemplo la llegada de un usuario al aeropuerto de Barajas, quien viajará por razones laborales. El usuario deseará ante todo realizar las tareas imprescindibles para abordar, pero también dedicará su tiempo a comprar regalos, comer o beber algo.

El agente intermediario envía mensajes a los distintos agentes proveedores de servicios acerca de las tareas a realizar por el usuario.

El agente intermediario hará una correspondencia (matching) de los servicios ofrecidos por los agentes proveedores registrados en el aeropuerto (check-in, shop, cafetería, etc.) con el perfil del usuario acerca de sus gustos personales.

El agente localizador del aeropuerto puede indicarle al usuario por medio de un mapa en su dispositivo móvil, la ubicación de cada oficina para los trámites aeroportuarios, así como el orden que debe seguir y cuales son las menores distancias desde donde realizarlos. En su perfil de usuario, cargado en su dispositivo móvil, existirá información respecto de sus preferencias sobre las tareas cotidianas: si es una persona que exige calidad, si le es relevante el tiempo que lleva una acción, si le da o no importancia al dinero por gastar, etc.. El agente usuario se encarga de presentar al usuario, de acuerdo a estas preferencias, una lista de tareas priorizadas que podría realizar de acuerdo a su perfil y localización. Por lo tanto, el usuario puede consultar y visualizar en su dispositivo móvil un conjunto de tareas y elegir cuales quiere ejecutar.

Figura 6.1: Algoritmo genético ejecutado por el agente intermediario.

El agente usuario es quien contiene el perfil del usuario (Fig. 6.2), es decir sus preferencias sobre cada tarea. Una tarea estará definida por un conjunto de características relevantes con su peso, lo que denominaremos preferencia, la cual representa la importancia o interés del usuario sobre dicha tarea.

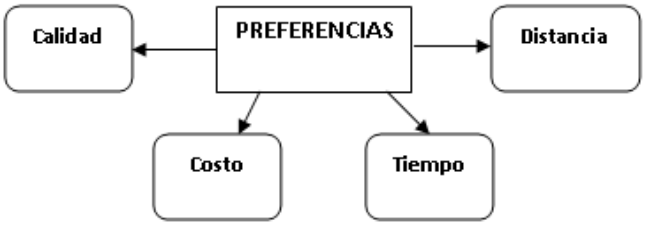

Figura 6.2: Preferencias del perfil de usuario.

El algoritmo que a continuación se presenta, tiene por resultado una lista de actividades priorizadas de acuerdo a las preferencias del usuario. El agente usuario de cualquier persona, aprende los gustos personales de su dueño, según el comportamiento que tiene en cada posición y momento del día.

El sistema envía una lista de tareas disponibles al agente del usuario de acuerdo a sus intereses. Podemos representar esa lista de tareas como:

$$\{T_i\}_{i=1}^{n} = T_1, T_2 ... T_n \tag{6.1}$$

donde cada tarea se define como un conjunto de características c_i :

$$T = \{C_i\}_{i=1}^{n} \tag{6.2}$$

A su vez, cada característica representa la calidad, el tiempo, el costo, o cualquier otra característica que una persona tiene en cuenta en el momento de realizar una actividad. Habitualmente, los usuarios asignan a esas tareas una prioridad, basada en la evaluación de las características descriptas anteriormente. Definimos entonces el interés del usuario como un conjunto de preferencias:

$$P = \{W_i\}_{i=1}^{n} \tag{6.3}$$

donde w es el peso del usuario sobre cada característica $c_1, c_2, ..., c_n$.

Estas preferencias pueden ser evaluadas sobre cada tarea mediante la importancia que el usuario le asigne. Dado que las preferencias son desconocidas, deben obtenerse a través de un algoritmo genético. Por lo tanto, se pueden definir las preferencias sobre cada tarea como:

$$P(T_i) = \sum_{i=1}^{n} w_i c_i \tag{6.4}$$

Dado que el AG necesita información de entrada, se opta por cargar en el sistema las n tareas disponibles junto con los pesos de las características de cada tarea. Por ejemplo, se crea en el sistema un vector que representa la tarea 1 $(T1)$, cuyos elementos representan las características calidad (q), costo (c), tiempo (t) y distancia (d), y los valores de cada elemento son $q = 2, c = 9, t = 9$ y $d = 7$. Cada valor representa la importancia de la característica en el sistema para $T1$. Luego, con el uso de un AG, se encuentran individuos con valores de características similares a los valores de las características del sistema. Se calcula el valor de preferencia para la tarea, que servirá luego para priorizar la tarea en la lista que se le presenta al usuario.

Por lo tanto, mediante el uso de un algoritmo genético se obtienen las preferencias del usuario y, de acuerdo a la preferencia obtenida sobre cada tarea, se construye una lista de tareas priorizada para presentar al usuario, y ayudarlo en la toma de decisiones. El hecho de poder aprender las preferencias del usuario (siendo esta información, información privada del usuario) permite a los agentes proveedores la oferta de nuevos servicios o productos de acuerdo a los intereses del usuario.

El AG genera entonces una población inicial de individuos aleatoria. Se repite el proceso por cada individuo, es decir, se calculan las preferencias para el individuo por cada tarea, y se ordena la lista de tareas obtenida del individuo según esas preferencias obtenidas. A continuación, a través de la función de evaluación, el algoritmo genético compara la posición de cada tarea en la lista del usuario con la lista del individuo. El valor óptimo de la función de evaluación, es '0', pero el algoritmo se detendrá cuando haya llegado al número de generaciones definido en el sistema, para lo cual, en cada iteración, realizará la reproducción de individuos, de acuerdo a los parámetros configurados (tipo de reproducción y de selección, probabilidad de mutación, etc).

Se tiene un individuo, representado en la Ec. 6.5, con un cromosoma con tantos genes como características deben ser evaluadas. Cada gen representa el valor w de cada característica ci:

$$Individuo \rightarrow \bar{w} = \{w_i\} \tag{6.5}$$

dónde cada w_i se codifica con 10 bits en un rango del $[0 - 9]$.

Antes de presentar la lista de tareas al usuario, se construyen los perfiles de usuario que definen las preferencias sobre cada tarea. El proceso de recomendación de tarea puede empezar una vez que se ha adquirido el perfil del usuario. A partir del usuario A dado, se debe encontrar un conjunto de individuos con perfiles similares a M. Nosotros utilizamos el AG para obtener de manera evolutiva los pesos que cada usuario asigna a sus preferencias, para adquirir una lista de tareas priorizada. Entonces, por cada individuo, realizamos una priorización de las tareas y luego utilizamos la "distancia" para escoger cual de los individuos seleccionados es el más cercano al perfil del usuario M.

La distancia puede ser de dos tipos:

- Genotípica: mide la distancia de codificación de dos individuos
- Fenotípica: mide lo que se parece a las soluciones generadas por cada uno de los individuos a comparar. Dependiendo del dominio del problema, se suele usar la función "Euclídea".

Utilizamos la distancia fenotípica para medir cual es la solución generada por cada individuo a comparar. La confrontación de la lista de tareas generada por cada individuo y las actividades seleccionadas por el usuario, puede ser llevada a cabo a través de una distancia euclídea modificada, la cual tiene en cuenta las distintas características c_i. Asumimos que la aplicación de heurísticas de AG ha evolucionado los valores de las características de las preferencias de las tareas, y ha obtenido una lista de tareas ordenada, gracias al individuo. Por lo tanto, la función euclídea (M, I), mostrada en la A.3, es la distancia entre el usuario M y el individuo I, en donde la sumatoria desde $i = 0$ hasta n se realiza según las n tareas definidas en el sistema. La función de fitness se calcula en relación a la posición j de la tarea i en la lista.

$$Distancia - Euclidea(M, I) = \sum_{i=1}^{n} \sqrt{Dif(jM_i, jI_i)^2} \tag{6.6}$$

donde

M es la lista de tareas elegida por el usuario,
I es la lista de tareas dada por el proceso de selección, además $I \neq M$
i es la tarea, donde el Perfil(M, i) y el Perfil (I, i) existen
n es la cantidad de tareas que el usuario puede realizar
j indica la posición de la tarea en la lista de tareas priorizada.
dif representa la diferencia entre la posición de la tarea i entre el usuario M y el individuo I.

La función de evaluación, calcula la distancia de cada individuo en la población (todas las listas de tareas propuestas) con la lista de tareas construida por el usuario. La distancia entre

las dos lista es 0 cuando se ha encontrado la solución óptima, lo cual representaría una lista de tareas idéntica a la elegida por el usuario.

Por ejemplo, suponemos la evaluación de las características: calidad, costo, tiempo y distancia. La valoración de cada característica para el usuario M es un vector con los valores siguientes: calidad, costo, tiempo y distancia respectivamente. Para cada tarea se introduce en el sistema un valor de cada característica (por ejemplo, tarea 1 con calidad=2, costo=9, tiempo=9 y distancia=7). Una vez introducidos estos parámetros en el algoritmo genético como "pre-procesados", se generan los individuos pertenecientes al perfil de usuario, calculando la diferencia entre las preferencias de las tareas del usuario (usuario M) y las preferencias de las tareas de cada individuo obtenido. Seguidamente, con las preferencias obtenidas por cada tarea, el AG genera una lista de tareas ordenada de acuerdo a los valores de las preferencias de cada tarea, y finalmente aplica el fitness para evaluar cuan diferente es la lista del usuario de la lista obtenida a partir del mejor individuo de la población. Una diferencia cercana a 0 hablará de listas similares.

Tabla 6.1: Aprendizaje de tareas de los individuos I1, I2 e I3 obtenidos por el AG en relación al usuario. A

Usuario M	I1	I2	I3
T1	T1	T1	T1
T3	T2	T2	T3
T2	T3	T3	T2
T4	T5	T4	T4
T5	T4	T5	T5
Diferencia entre A-I	4	2	0

6.4 Algoritmo

El algoritmo definido para la resolución del problema se muestra en la Fig. 6.3. En principio se inicializan los valores para calidad, costo, tiempo y distancia del usuario A y las mismas características con los valores correspondientes a cada tarea definida en el sistema. Luego se calculan las preferencias del usuario para cada tarea del sistema, y se las introduce en una lista, la cual es ordenada de mayor a menor, posteriormente, a modo de realizar una priorización en las tareas de acuerdo a los gustos del usuario.

El sistema genera aleatoriamente una población de individuos inicial. Luego por cada individuo, se realiza el mismo procedimiento que se realizó para el usuario, es decir, calcular las preferencias para el individuo respecto de cada tarea y ordenar la lista de tareas obtenida para dicho individuo. No debemos perder de vista el objetivo del AG de arrimar las preferencias del individuo a las del usuario.

Luego, mediante la función de evaluación, se compara la posición de cada tarea en la lista del usuario respecto a la lista del individuo. El valor óptimo de la función de evaluación, como se dijo anteriormente, es '0', pero el algoritmo se detendrá cuando haya llegado al número de

generaciones definido, para lo cual, en cada iteración, realizará la reproducción de individuos, también acorde a los parámetros configurados (tipo de reproducción y de selección, probabilidad de mutación, etc).

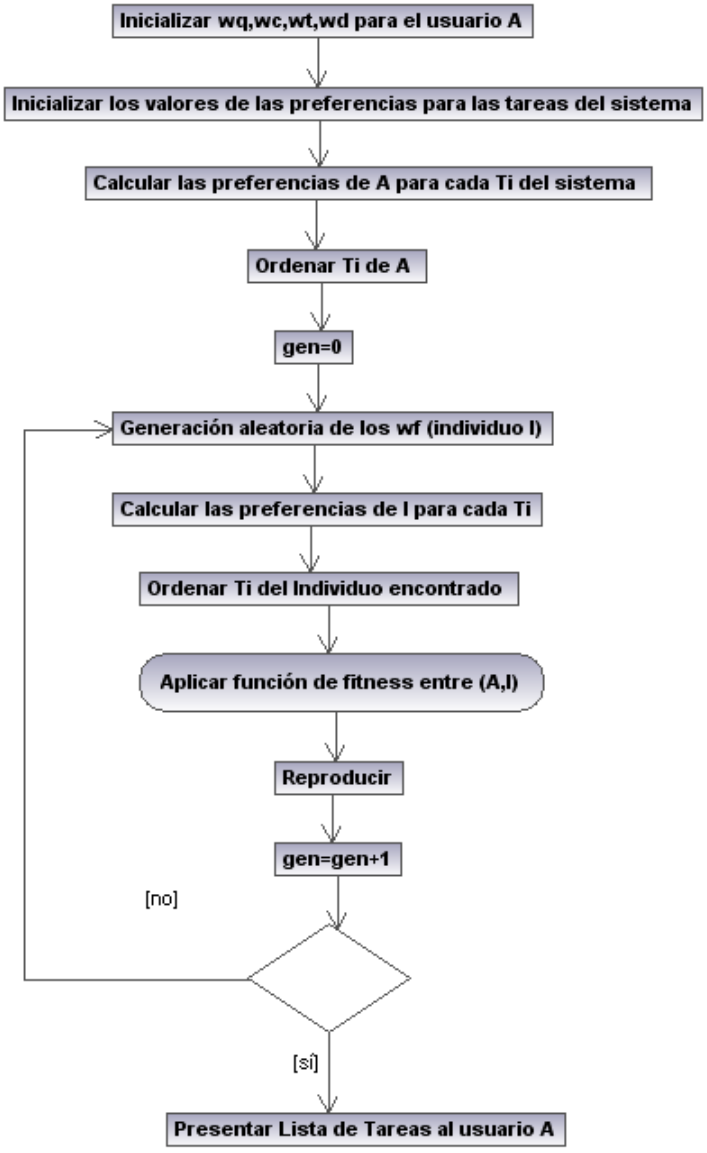

Figura 6.3: Algoritmo para el aprendizaje de preferencias.

6.5 Herramienta GOAL

GOAL es una herramienta de optimización basada en algoritmos genéticos la cual puede resolver una gran variedad de problemas de optimización [1]. GOAL puede manejar problemas

[1] http://www.geocities.com/geneticoptimization/GOAL.html

con variables reales y binarias, restricciones y complejas funciones objetivo escritas con Visual Basic. Este software tiene un editor que permite fácilmente la formulación de un problema de optimización, se muestra su interfaz en la Fig. 6.4. Es una aplicación de distribución gratuita.

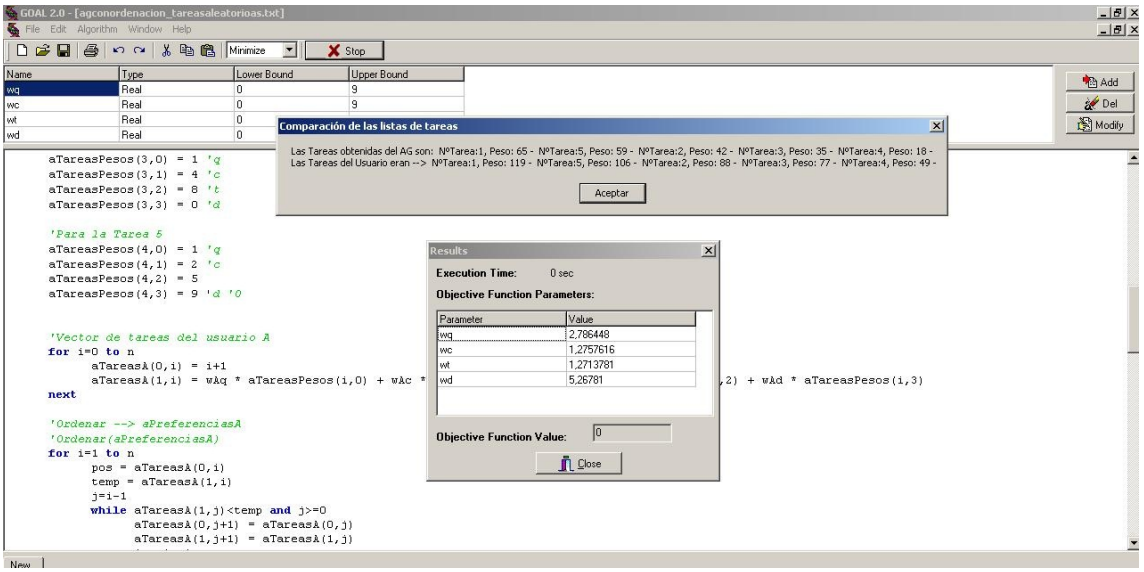

Figura 6.4: Herramienta GOAL.

GOAL permite trabajar introduciendo las siguientes opciones: tamaño de la población, número de generaciones, tipo de reproducción, probabilidad de reproducción, selección, probabilidad de mutación, elitismo (individuos a preservar), período de refresh de la población y porcentaje de población refrescada.

Para definir un problema en esta herramienta, es necesario seguir los siguientes pasos, definiendo:

1. los parámetros, los cuales son variables $x1, x2...x_n$ de la función objetivo y de las restricciones, como así también su tipo de dato y los valores mínimo y máximos permitidos,
2. la función objetivo $f(x)$ ha optimizar,
3. las restricciones $g(x)$,
4. las funciones de pre-procesamiento y de post-procesamiento, las cuales son funciones que solo se evalúan al inicio o fin de la ejecución del algoritmo, respectivamente.

La Fig. 6.5 es la imagen de la pantalla de configuración de parámetros de la herramienta.

Una vez introducido el problema en GOAL, para ejecutarlo se le debe indicar si la función objetivo ha de ser minimizada o maximizada. A partir de allí genera (evoluciona) valores para los parámetros introducidos, ejecuta el algoritmo y muestra por pantalla los resultados óptimos para los parámetros y la función objetivos, como así también genera un reporte (ver Fig. 6.6) en el que se pueden analizar los valores obtenidos en cada generación.

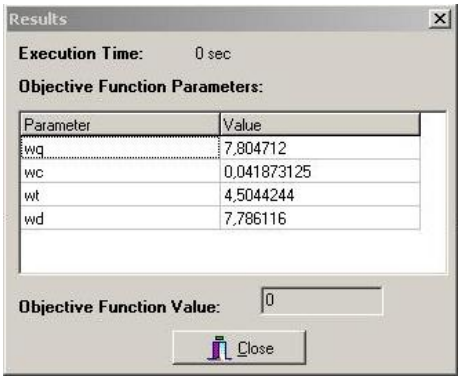

Figura 6.5: Configuración de parámetros en GOAL.

Generation	No. Valid Solutions	Mean	Std Deviation	Mutations	Reproductions	Optimum genotype	Optimum fenotype	Optimum
0	6	-3,3333	2,133333333	4	2	1,01E+127	7,804712; 0,66997; 4,50445; 255135,449088;	2
1	2	-0,33333	0,4	5	3	1,01E+127	7,804712; 4,8524584; 0,9567285; 7,542904;	2
2	2	-0,33333	0,4	1	2	1,01E+127	4,4066856; 0,66997; 4,50445; 255135,449088;	0
3	3	-0,33333	0,466666667	2	2	1,01E+127	7,804712; 0,041873125; 4,50445; 7,786116;	0
4	4	-0,33333	0,533333333	1	3	1,01E+127	7,804712; 0,041873125; 4,50445; 7,786116;	0
5	5	0	0	2	2	1,01E+127	7,804712; 0,041873125; 4,5044244; 7,786116;	0
6	6	-0,33333	0,666666667	4	3	1,01E+127	5,70756; 0,041873125; 4,50445; 7,786116;	0
7	6	-0,33333	0,666666667	1	3	1,01E+127	7,804712; 0,041873125; 4,5044244; 7,786116;	0
8	6	0	0	5	3	1,01E+127	7,804712; 0,041873125; 4,5042452; 7,786116;	0
9	5	0	0	3	3	1,01E+127	7,804712; 0,0418725; 4,5042452; 2,288133843211 76E-38;	0
10	5	-1,3333	1,866666667	0	3	1,011E+127	7,816552; 0,041873125; 4,5044244; 7,786116;	0

Figura 6.6: Información del reporte retornado por GOAL .

6.6 Escenario de Ejemplo

María (M) llega de vacaciones a la ciudad de Salta. Durante su visita desea realizar actividades relacionadas a la naturaleza y actividades culturales típicas de la ciudad.

Se supone el siguiente listado de tareas posibles en el sistema:

- Tarea 1: visita al parque nacional El Rey
- Tarea 2: excursión hacia las Ruinas de Quilmes
- Tarea 3: paseo de dos días por los viajes Calchaquíes
- Tarea 4: asistir a la fiesta tradicional de la Pachamama (Madre Tierra)
- Tarea 5: recorrido por el mercado artesanal

Asimismo, el listado de las características que el usuario evalúa antes de ejecutar una tarea es:

- Calidad (q)
- Costo económico (c)
- Tiempo (t)
- Distancia (d)

El agente intermediario (BA) envía la lista de tareas que puede ejecutar la persona, al agente usuario del dispositivo móvil de María. Ella elige qué tareas ejecutar y en qué orden hacerlo. A su vez, el algoritmo genético, ejecutado por el BA, tratará luego de obtener una lista de tareas similar a la que finalmente ejecuta María. De esta manera, el agente intermediario obtiene las preferencias de María, a través de la observación de sus movimientos, comunicados por el agente localizador. En el algoritmo genético, cada lista de tareas es un individuo. Por ende, el resultado de su ejecución es un conjunto de individuos, de los cuales uno de ellos es el mejor o el más cercano a la lista de tareas que el usuario ha elegido (6.7).

En otro momento, quizá después de ejecutar una tarea, o cuando el usuario se encuentre en otro dominio, el sistema presentará la lista de tareas obtenidas anteriormente, es decir, el mejor individuo del AG. El usuario puede elegir ahora que tareas ejecutar y el orden de las mismas, y el algoritmo genético se ejecutará nuevamente. Se observa como la entrada al AG es la lista de tareas del individuo. El algoritmo genético evoluciona las preferencias del usuario y retorna un conjunto de listas de tareas.

Figura 6.7: El AG obtiene el perfil de usuario a partir del orden de las tareas dado por el usuario.

Si nos centramos específicamente en la tarea "Excursión a las Ruinas de Quilmes", las preferencias de María sobre cada característica son las que se muestran en la Fig. 6.8 a continuación.

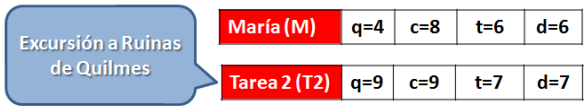

Figura 6.8: Caracterización de la tarea "Excursión a las Ruinas de Quilmes"

A María, le encanta conocer nuevos sitios, detesta el viaje hasta el lugar que visitará. Sufre de tener que estar tanto tiempo sentada. No es exigente con la calidad, ella piensa que lo más importante es conocer. Eso si, no es una persona que malgaste el dinero, por el contrario, suele cuidar mucho sus gastos durante las vacaciones. Ya le parece bastante el costo de los billetes hacia el destino elegido.

Dentro del sistema la excursión a las Ruinas de Quilmes ($T2$) es una de las de más alta calidad, dentro de todas las que ofrecen para conocer los alrededores, y por ende la más costosa. En el sistema se caracteriza esta visita con un tiempo $t = 7$ puesto que normalmente lleva dos días de excursión, y una distancia $d = 7$ porque son cerca de 300 km de distancia solo de ida. Allí hay un lujoso hotel en donde pasar la noche.

Al evaluar las preferencias del usuario sobre cada característica obtenemos:

$$P(T_4) = w_q * T_{4,q} + w_c * T_{4,c} + w_t * T_{4,t} + w_d * T_{4,d}$$

$$P(T_4) = 4 * 9 + 8 * 9 + 6 * 7 + 6 * 7 = 192$$

El valor obtenido, es decir 192, es el valor que le da María a la tarea número 2 "Excursión a las Ruinas de Quilmes".

Si hacemos este cálculo, basado en las preferencias del usuario, por cada acción que planea llevar adelante, se obtiene las preferencias de María por cada tarea:

$$P(T_1) = 165$$
$$P(T_2) = 192$$
$$P(T_3) = 170$$
$$P(T_4) = 100$$
$$P(T_5) = 92$$

A continuación se muestra, en la Fig. 6.9, como de acuerdo a las preferencias sobre cada tarea, el AG obtiene un individuo similar a la lista de tareas ordenada, según María:

(M)	T2	T3	T1	T4	T5
(I)	T2	T3	T1	T4	T5
Posición	0	1	2	3	4

Figura 6.9: Posición de cada tarea en la lista del usuario y en el individuo.

A través de la función de evaluación propuesta en el modelo, se mide cuan diferente es el individuo (I) a la lista de tareas original (M) de María, considerando la posición que ocupa cada tarea en la lista, como se representa en la Ec. 6.7.

$$Distancia - Euclidea(A, I) = \sqrt{(2-2)^2 + (0-0)^2 + (1-1)^2 + (3-3)^2 + (4-4)^2} = 0 \quad (6.7)$$

El resultado de la función de fitness es cero. Este resultado indica que el algoritmo genético ha encontrado un individuo exactamente igual a la lista de tareas ejecutada por el usuario.

Tabla 6.2: Comparación de preferencias del usuario con los individuos obtenidos, evaluando 4 características y 5 tareas, en 10 generaciones.

Usuario		**Ind. Malo**		**Ind. Bueno**		**Ind. Óptimo**	
Tarea	Peso	Tarea	Peso	Tarea	Peso	Tarea	Peso
1	119	1	111	1	125	1	111
5	106	2	100	2	90	5	101
2	88	4	57	5	89	2	79
3	77	5	45	3	71	3	72
4	49	3	44	4	56	4	44

6.7 Experimentos

6.7.1 Evaluando la Cantidad de Tareas

Es necesaria la configuración de un conjunto de parámetros para definir el problema con AG, estos son las variables $x1, x2...xn$ de la función objetivo y de las restricciones, como así también su tipo de dato y los valores mínimo y máximos permitidos, la función objetivo $f(x)$ ha optimizar, las restricciones $g(x)$ y las funciones de pre y post-procesamiento, las cuales son funciones que solo se evalúan al inicio o fin de la ejecución del algoritmo, respectivamente. Además debemos definir el tamaño de la población, número de generaciones, tipo de reproducción, probabilidad de reproducción, selección, probabilidad de mutación y elitismo (cantidad de individuos a preservar). Se trabajó sobre una población inicial de 6 individuos. Se ha elegido desarrollar un AG elitista que solo preserva 1 individuo de la población anterior y con un punto de cruce. La selección realizada fue la selección por entrenamiento, la probabilidad de mutación fue de $0,005$, la de reproducción de $0,89$ y la de selección de $0,85$. Los resultados que se muestran a continuación, muestran la tendencia en la predicción por medio de técnicas de AG.

El concepto de convergencia está relacionado con la progresión de la uniformidad: la generación converge cuando al menos el $95\,\%$ de los individuos en la población comparten el mismo valor para ese gen (Nijholt, 2007). Se asume que la población converge cuando todos los genes han convergido. La población ha evolucionado al máximo global.

Se han efectuado pruebas en donde la lista de tareas del sistema fue variando en un número de $5, 10, 20, 50$ y 100 tareas. Para cada caso se corrió el programa con 1000 generaciones, como se observa en el gráfico de la Fig. 6.10.

La optimalidad viene dada por un fitness de valor 0, lo cual indica que se ha encontrado un individuo (una lista de tareas idéntica) a la que eligió ejecutar el usuario. Del total de las pruebas realizadas, en el $99\,\%$ de los casos se llega a la solución óptima, el $0,1\,\%$ fueron resultados muy malos y el resto $(0,9\,\%)$ soluciones buenas, aunque no óptimas.

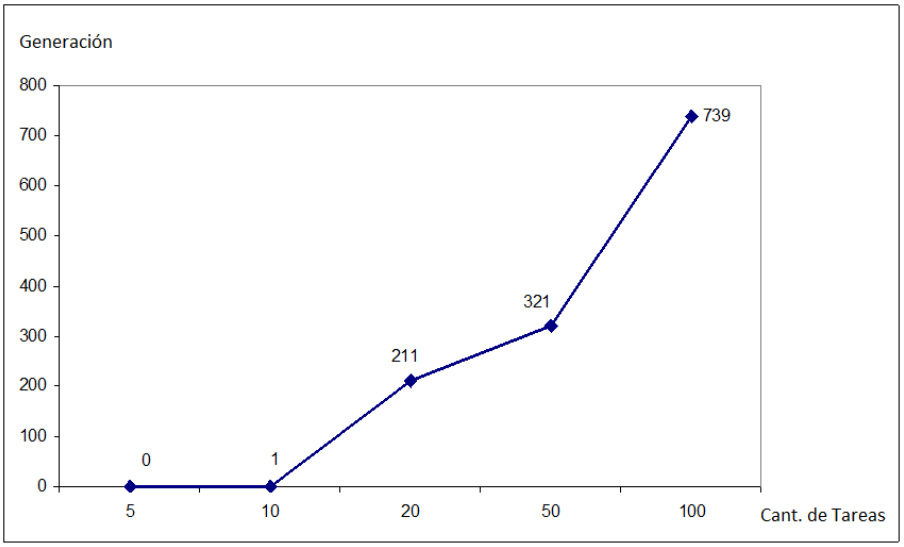

Figura 6.10: Convergencia del AG para 4 características incrementando el número de tareas.

En la Fig. 6.11 se muestra la evolución del algoritmo genético a medida que se incrementa la cantidad de tareas. Se observa que para un problema con 10 tareas (experimento 1) el AG converge rápidamente (en la segunda iteración). En el experimento 2, para 20 tareas el algoritmo converge luego de 211 iteraciones, mientras que para 50 tareas (experimento 3) lo hace después de 338 iteraciones. Para un problema de 100 tareas el AG converge en la iteración 739. En promedio con 100 tareas el AG converge más rápidamente que con 50 Tareas.

El algoritmo construido tiende a obtener la solución en muy pocos segundos por más que crezca la cantidad de tareas de cinco a cien. Luego de ejecutar numerosas simulaciones del programa, se concluye en que tiene un margen de error muy pequeño a la hora de buscar el mínimo, y este problema se soluciona aumentando la cantidad de individuos de la población inicial. Asimismo, en las ejecuciones con 5, 20, 50 y 100 tareas se obtuvo la solución óptima en un rango de cero a un segundo. Mientras que al ejecutarlo para 500 tareas lo hizo en 24 segundos, lo cual es bastante bueno.

6.7.2 Evaluando las Características

Para evaluar el comportamiento del algoritmo en base a la cantidad de características a considerar para calcular las preferencias, se realizaron algunos experimentos modificando la cantidad de tareas y de características del entorno. Los experimentos se realizaron para listas de 5 y 10 tareas, variando la cantidad de características entre 5, 15, 30 y 50. Para cada configuración de los experimentos, combinando la cantidad de tareas y características, se ejecutaron 25 simulaciones. En las Fig. 6.12 y 6.13 se observan los resultados obtenidos para listas de 5 y 10 tareas respectivamente. Sobre el eje x se refleja el número de simulación, cada una de ellas contempla 100 generaciones de la población, mientras que el eje y refleja el fitness obtenido. Las líneas verdes representan a los mejores individuos obtenidos durante la simulación, es decir, aquellos individuos que minimizan el fitness. Con rojo se muestran los

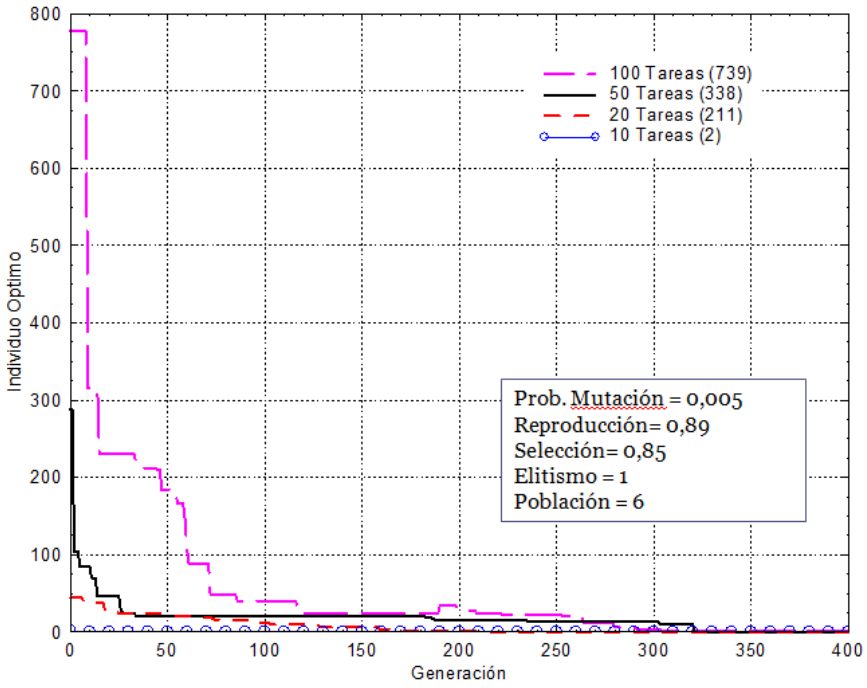

Figura 6.11: Evolución del AG con 10, 20, 50 y 100 tareas para 4 características.

peores individuos. Y finalmente con azul, el valor de fitness medio. La media se encuentra más cercana a los óptimos (mejores individuos de la simulación). Los resultados son lo esperado, pues el sistema puede considerar muchas características, y eso es trasparente al usuario.

El algoritmo ejecutado para una lista de 5 tareas (ver Fig. 6.12) obtiene sus peores fitness en un valor máximo de 4. Pero, en particular para los casos de 5 y 15 tareas, coincide el mejor fitness con el peor, es decir, un fitness óptimo igual a 0. Este comportamiento se observa por ejemplo en las simulaciones 1 y 2, y de la 5 a la 10 para el gráfico de 5 tareas y 5 características. En muy pocas ocasiones el algoritmo no encuentra el fitness óptimo, por ejemplo, en la simulación 16 para 5 características, en la 14 para 15 características, en las simulaciones 16 y 24 para 30 características y en la simulación número 11 para 50 características. Este fitness califica como "bueno" y por lo general alcanza un valor $fitness = 2$, menos para las 50 características en donde $fitness = 4$. Lo cual se interpreta como que muy pocas tareas no aparecen en el orden en la lista en el que deberían estar.

En lo que respecta a la configuración del algoritmo para 10 tareas y las diferentes cantidades de características evaluadas (Fig. 6.13), se observa que en muy pocas situaciones se obtiene el fitness óptimo de $fitness = 0$. En particular, para 30 y 50 características la diferencia entre los mejores y la media, y los peores y la media, es muy similar. Obsérvese que en estos gráficos los peores fitness alcanzan valores mucho mayores a los obtenidos para las simulaciones con 5 tareas. Por ejemplo, para 5 características, se observa un pico en las simulaciones 18 y 20 en donde los peores fitness alcanzan valores superiores a 10. Casos extremos se dan para los experimentos con 50 características en donde para las simulaciones 10 y 14 el peor fitness supera los 20. En lo que respecta a los mejores individuos obtenidos, por lo general para 100 generaciones se obtiene valores entre 2 y 10 con lo cual, al aumentar la cantidad de generaciones, se asegura la

obtención de un máximo óptimo.

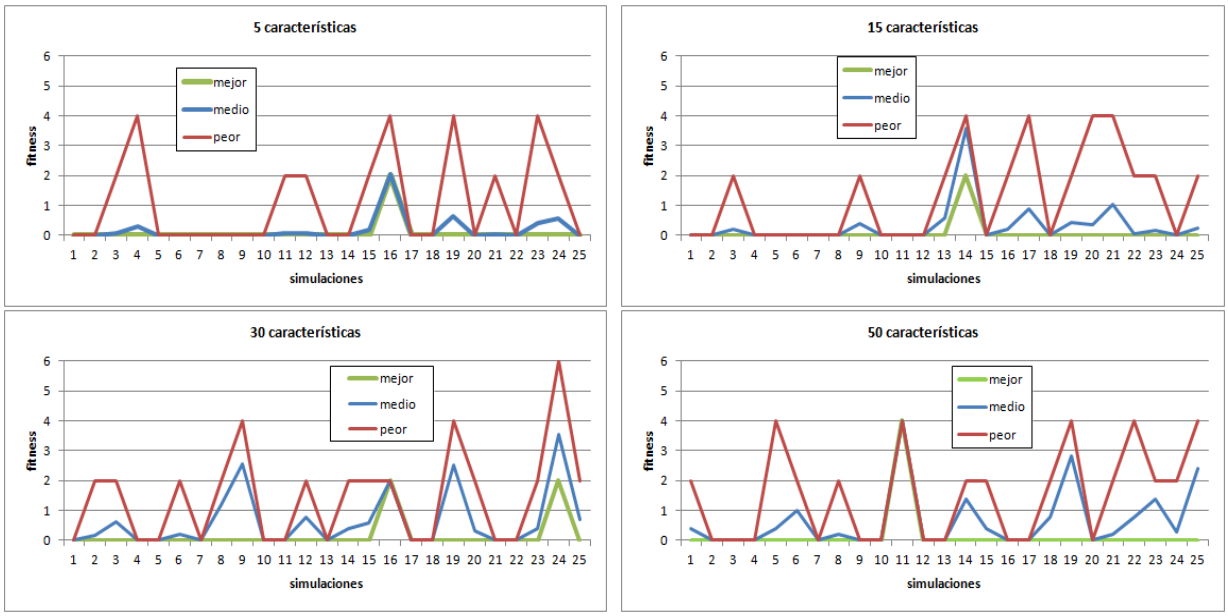

Figura 6.12: Comportamiento del AG para 5 tareas incrementando la cantidad de características que se evolucionan.

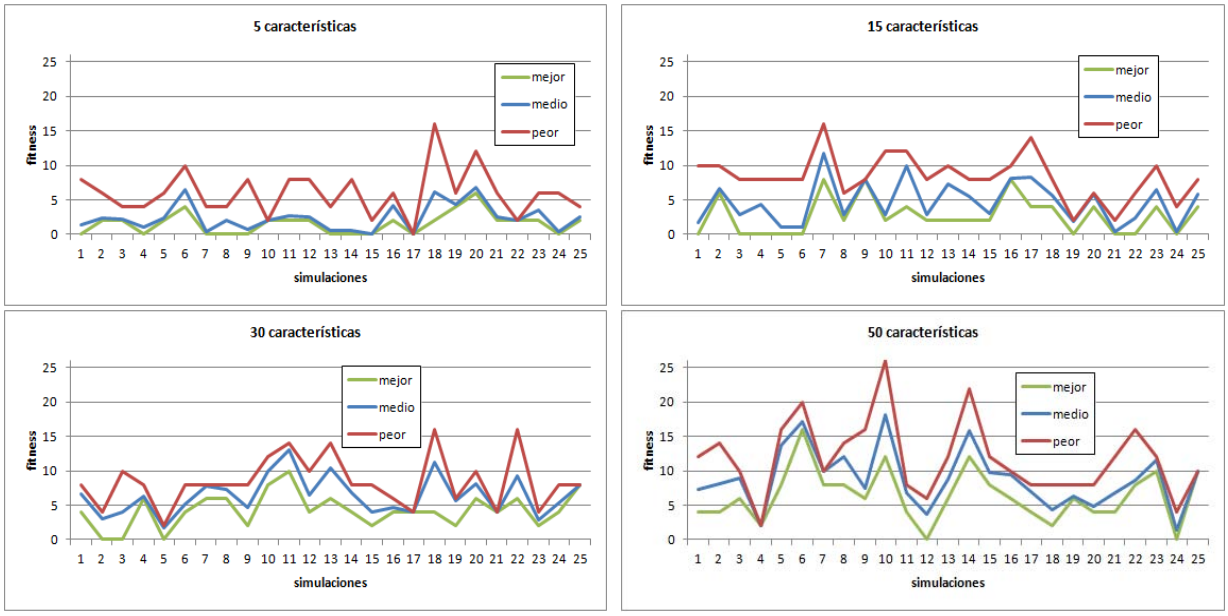

Figura 6.13: Comportamiento del AG para 10 tareas incrementando la cantidad de características que se evolucionan.

También se realizaron experimentos para 70 características, pero en estos casos, el algoritmo genético no presentó una buena respuesta. En algunas simulaciones se obtenían máximos

locales, y en otras simulaciones no se encontraba una solución, pues es muy complejo resolver un problema con tan pocas tareas y tantas características. En la Fig. 6.14 se muestra el comportamiento del genético para una configuración de 5 tareas y 70 características para 10 simulaciones. Obsérvese como en algunos casos el algoritmo no puede obtener una solución, lo cual se indica por una flecha y la leyenda "infinito". Por lo tanto, sin un estudio profundo, podría decirse que hasta 70 características o quizá menos, es un número aceptable para el buen funcionamiento del AG.

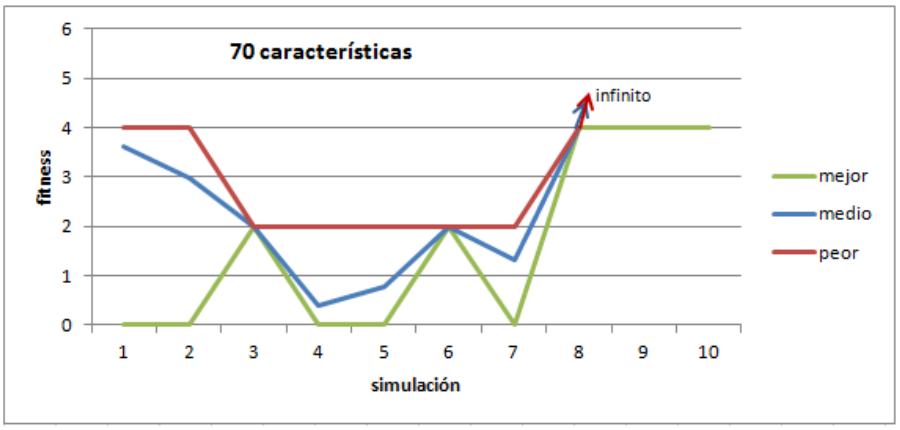

Figura 6.14: Evolución del AG para 5 tareas y 70 características. Para esta cantidad de características el genético no puede encontrar la solución.

Para todos los casos, la configuración del algoritmo fue la siguiente:

- Población $= 10$
- Generaciones $= 100$
- Tipo de reproducción $=$ Cruce uniforme
- Tipo de selección $=$ torneo
- Elitismo $= 5$
- Probabilidad de mutación $= 0{,}005$
- Probabilidad de reproducción $= 0{,}85$
- Probabilidad de selección $= 0{,}85$

En la siguiente figura (Fig. 6.15) se muestra la convergencia del genético ejecutado para las diferentes configuraciones de características y considerando los casos de 5 y 10 tareas. Al incrementar la cantidad de tareas, el algoritmo necesita de más generaciones para converger. Asimismo se observa la veloz convergencia que posee al ejecutarse para 5 tareas.

La Tabla 6.3 muestra la evolución del genético a través de las distintas generaciones para cada par de configuración tareas-características, tomando los datos a partir del promedio de

25 simulaciones para cada caso. En los gráficos de la Fig. 6.15 se observa sobre el eje x la cantidad de generaciones y sobre el eje y los valores de fitness óptimo por generación. Con línea azul se representa al genético ejecutado para 10 tareas y con línea de color verde, para 5 tareas. La convergencia del algoritmo para 10 tareas sigue al comportamiento que posee para 5 tareas, solo que de una forma más escalonada, debido a que es más complejo el problema al incrementar la cantidad de tareas. De estos gráficos se deduce que para 5 tareas son necesarias menos de 25 generaciones para las diferentes combinaciones de características, mientras que para 10 tareas, el algoritmo requiere de mayor cantidad de generaciones para converger.

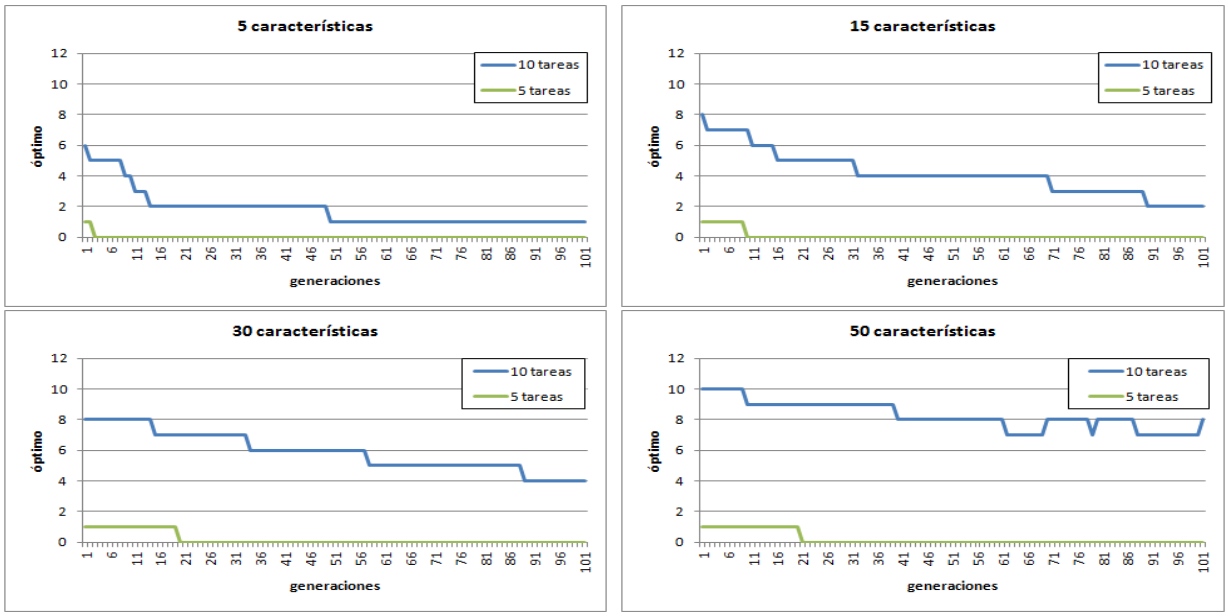

Figura 6.15: Convergencia del AG para 5 y 10 tareas variando las características en 5, 15, 30 y 50.

6.8 Conclusiones de la Aplicación de AG en la Obtención de Preferencias

En este apartado, se ha presentado el uso de heurísticas de AG para saber qué es lo que realmente desea el usuario. Se ha construido un algoritmo que aprende los pesos que el usuario da a cada característica sin que tenga el usuario que decir nada. El usuario lo único que ve es una lista de tareas y las ordena. Se extrae su perfil basado en la importancia que tiene para él cada característica. Las solución está pensada para ejecutarse diariamente, es decir que las tareas pueden ser las básicas que se hacen todos los días, y coger datos de la hora, el clima etc., de manera que cada vez que el usuario selecciona las tareas, el sistema aprende. Específicamente, se genera una lista de tareas priorizadas de acuerdo a las preferencias del usuario.

Para realizar la extracción de las preferencias se optó por una técnica que no utilizara

Tabla 6.3: Fitness medio por generación para 25 simulaciones de cada par características-tareas.

	5 características		15 características		30 características		50 características	
$N°$ de Generación	5 tareas	10 tareas	5 tareas	10 tareas	5 tareas	10 tareas	5 tareas	10 tareas
1	1	6	1	8	1	8	1	10
5	0	5	1	7	1	8	1	10
10	0	4	0	7	1	8	1	9
15	0	2	0	6	1	7	1	9
20	0	2	0	5	0	7	1	9
25	0	2	0	5	0	7	0	9
30	0	2	0	5	0	7	0	9
35	0	2	0	4	0	6	0	9
40	0	2	0	4	0	6	0	8
45	0	2	0	4	0	6	0	8
50	0	1	0	4	0	6	0	8
55	0	1	0	4	0	6	0	8
60	0	1	0	4	0	5	0	8
65	0	1	0	4	0	5	0	7
70	0	1	0	4	0	5	0	8
75	0	1	0	3	0	5	0	8
80	0	1	0	3	0	5	0	8
85	0	1	0	3	0	5	0	8
90	0	1	0	2	0	4	0	7
95	0	1	0	2	0	4	0	7
100	0	1	0	2	0	4	0	7

información adicional a lo seleccionado por el usuario, es decir que sea independiente de la complejidad del problema pudiendo variar el número de características y de tareas y el algoritmo ser el mismo. Por tal motivo se decidió usar algoritmos genéticos.

Se realizaron experimentos variando la cantidad de tareas y de características involucradas en diferentes problemas. Se evaluó cada aspecto por separado. Se realizaron simulaciones para ver el comportamiento del algoritmo genético para una configuración básica de 4 características incrementando el número de tareas, y se observó la velocidad de convergencia. Asimismo, se realizaron experimentos considerando listas de tareas de 5 y 10 tareas, variando la cantidad de características en 5, 15, 30, 50 y 70. Se observó cómo la media del fitness obtenido para 5 tareas con distintas cantidades de características siempre está cercano al óptimo, mientras que para 10 tareas, a mayor cantidad de características la brecha entre los mejores individuos y la media, es similar a la de los peores individuos y la media. También se evaluó la convergencia para estas configuraciones, concluyendo que en tan solo 25 generaciones o menos para el caso de 5 tareas, el genético converge. En el caso de las simulaciones para 10 tareas, es necesario más de 100 generaciones para obtener los óptimos.

La aplicación de algoritmos genéticos permite obtener las preferencias implícitas del usuario en algunas características de las tareas de una lista ordenada de estas tareas, con lo cual a nivel de agentes puede resumirse como obtener la información de perfil privada, que el agente usuario no quiere dar a conocer. El agente intermediario puede ejecutar entonces el AG para obtener los gustos del cliente. Mediante la aplicación de un AG en el agente intermediario, podemos ayudar a nuestros agentes usuarios a tomar decisiones respecto de las tareas a ejecutar, según la evaluación de sus preferencias.

Se ha utilizado el software GOAL para ejecutar el AG que obtiene, de forma evolutiva, los valores que cada usuario asigna a sus preferencias. Por último, se analizaron diferentes ejecuciones de este AG con el objetivo de especificar en qué número de generación el algoritmo converge cuando el número de tareas aumenta. Llegamos a la conclusión de que este método es más eficaz cuando el número de tareas a considerar se encuentra comprendido entre 10 y 100.

7

CALoR: Modelo de Reputación basado en Contexto y Localización

EXISTEN ciertas ocasiones en las que la falta de confianza sobre un individuo con el que queremos interactuar nos lleva a pedir recomendaciones. En estas circunstancias, normalmente uno piensa en consultar a las personas conocidas. Hoy en día, gracias a las facilidades de la telefonía celular, sería posible seleccionar algunos contactos del teléfono móvil y enviarles un mensaje pidiendo información respecto del proveedor que nos interesa. La persona que recibe el mensaje puede evaluar su experiencia pasada sobre el individuo en el que estamos interesados. Es en este momento cuando el SMA entra en funcionamiento. De alguna manera, le indicamos a nuestro agente usuario que busque cuales de nuestros contactos han tenido relación con el individuo y les solicite información al respecto. A tales fines, es necesario un modelo de confianza dentro de los sistemas multi-agentes para de esta manera evitar fraudes en la interacción de los agentes y permitirles realizar negociaciones más eficientes. Lo que proponemos es un sistema mixto que tiene parte centralizada y parte distribuida. Hay una parte del perfil que es adquirida mediante la observación del patrón de comportamientos de los agentes de usuario y que a partir de este perfil público se pueden hacer recomendaciones porque se infiere la reputación y la confianza de un agente a partir de las similitudes entre perfiles públicos. Habitualmente, se considera perfil público al que explícitamente el usuario ha decidido compartir, pero en el caso explicado en la presente sección, el perfil público es implícito, es el usuario quien ha producido con sus actos (y movimientos) y que el sistema multi-agente, presentado en secciones anteriores, ha inferido de forma no intrusiva (sin preguntarle al usuario, solo observándole). Pero eso no es incompatible con que cada agente usuario tenga un perfil privado e inaccesible, que es una expresión de los gustos del usuario al que el agente representa y que no comparte con nadie, pero que puede aplicar localmente para generar recomendaciones y para confiar en otros agentes comparando este perfil privado con otros perfiles. Las recomendaciones basadas en el perfil privado serán más precisas (está usando exactamente los gustos del usuario, y no la interpretación de los mismos que se obtiene al observar sus actos), pero requiere de la participación activa del usuario (tiene que querer usar su perfil para recomendar), en cambio las recomendaciones basadas en el perfil público, aunque más indirectas, permiten recomendar sin que el usuario participe directamente. Además de la reputación directa dada por un agente sobre otro, un dato esencial para el cálculo de la reputación es el dato de la posición del agente se basa en que a mayor distancia cliente-proveedor, menor capacidad de discernir la calidad del

servicio proporcionado por el proveedor. Pues no es lo mismo valorar la Gioconda en el Louvre (a distancia y con un cristal y personas de por medio) que las Meninas en el Prado. Es importante obtener un valor que represente la relación entre la posición y la evaluación de una interacción. Con este propósito y tomando como base la publicación (Bishr and Mantelas, 2008), donde se propone un modelo de confianza y reputación para clasificar y filtrar información geográfica aportada colaborativamente, se presenta un modelo de confianza que considera la posición de los agentes (usuarios y proveedores) para el cálculo de la reputación. Pero el peso de este dato tendría muy poco sentido si no se lo asociara al momento en que realizó la valoración. Que queremos decir con esto, no es lo mismo haber asignado un 8 de reputación directa a un agente hace mas de 10 meses, que haberle asignado el mismo puntaje hace un par de días o semanas. Esto se debe a que la calidad con la que se provee un servicio puede cambiar a lo largo del tiempo. A mayor tiempo pasado, mayor probabilidad de que haya cambiado. Por lo tanto, es necesario establecer una relación entre el tiempo en que se asignó un valor de reputación directa y cuan reciente es este dato. En este contexto, se presenta seguidamente el modelo de reputación CALoR (Context-Aware and Location Reputation Model).

7.1 Extensión de la Ontología para el Modelo de Reputación

En el sistema descrito en el capítulo 4, los usuarios poseen dos perfiles:

- Un perfil público: adquirido observando los patrones de comportamiento de los agentes usuarios. El perfil público está asociado a las preferencias explícitamente publicadas por el usuario, en esta tesis se considera el perfil publico cómo aquel que se produce automáticamente a través de la observación de sus actos (movimientos) por el agente localizador. Por otro lado, el agente intermediario, BA, es quien infiere las preferencias del usuario a través de sus movimientos de una manera no intrusiva. A través de este perfil público, el usuario intermediario puede decidir qué agente proveedor se adapta a las preferencias de usuario inferidas.
- Un perfil privado: cada agente usuario, UA, tiene además un perfil privado que es desconocido para cualquier otro agente. Este perfil privado es una expresión explícita de las preferencias del usuario, que él las configura en su agente de forma manual. Estas preferencias, nunca se comparten, pero si son utilizadas de forma interna para tomar ciertas decisiones: evaluar interacciones, considerar los valores de reputación y generar recomendaciones. Debido a esto, el perfil privado puede ser más específico y correcto, pero requiere de la activa participación del usuario

El perfil publico es parte de la ontología de agentes usada en el sistema. Consiste en un conjunto de interacciones de un agente usuario en particular, UA, denominadas *PastInteraction*. Cada interacción se define como una tupla $\langle PA, t, L, ie \rangle$, compuesta por los siguientes atributos:

- providerAgent: es el identificador de cualquier agente que brinde productos o servicios en la comunidad de agentes, con el que interacuó en el pasado. Instancia del concepto *Participant* en la ontología.
- Time: fecha y hora en la que ocurrió la interacción.

- L location, compuesta por un par de coordenadas (x,y), donde la coordenada x indica la latitud y la y, la longitud, de la posición del UA en el momento de la interacción. Se utilizan sistemas parecidos a los desarrollados por OpenStreetMap y GoogleMaps, es decir, sistema de posiciones por coordenadas geográficas.

- InteractionEvaluation (ie): este concepto representa la opinión de un agente solicitante sobre la calidad de interacción provista por el PA. Es un valor de reputación calculado por UA sobre el PA. Puede ser un valor en el intervalo de $[1, 10]$.

De manera adicional, un agente solicitante, UA, debe gestionar otro concepto, el cual representa las recomendaciones recibidas. Por cada tupla de interacción mencionada anteriormente, se tienen varias tuplas *ReceivedRecommendation*: $\langle RA, PA, ar \rangle$, donde RA es el UA que actúa como un agente recomendador, PA es el proveedor y ar (*Aggregated Recommendation*) representa la ganancia de la información recolectada que envía RA sobre el PA. Este valor muestra cual es la reputación del PA para el agente recomendador RA. El valor de ar lo calcula el RA utilizando las interacciones pasadas con PA que tiene almacenadas, y, luego envía este dato al UA respondiendo su solicitud de recomendación. Se debe tener en cuenta la frescura y la distancia entre el PA y RA de cada interacción. *ReceivedRecommendation* es la imagen reputacional que el agente recomendador tiene sobre un agente proveedor y que comparte con el agente solicitante.

Por lo tanto, con estos conceptos, cada UA actuando como agente recomendador, debe calcular el valor de *Aggregated Recommendation* sobre el agente proveedor, el cual incluye los siguientes conceptos adicionales:

- InteractionFreshness (if): se calcula como la diferencia entre el tiempo actual y el tiempo de una interacción dada. El valor de frescura de la interacción (interaction freshness) está en el rango $[0, 1]$.

- LocationDistance (ld): este valor se computa como la diferencia entre la posición del PA y el UA al momento de la interacción. El valore de ld puede ir de 0 a miles.

- InteractionEvaluation (ie)

Finalmente, los usuarios agentes, UA, actuando como agentes solicitantes, necesitan de otros dos conceptos a los fines de evaluar las recomendaciones recibidas y a los propios recomendadores:

- RecommenderReputation (rr): es el valor de reputación, la calidad del recomendador. Es decir, con cuánto éxito ese UA (en su rol de agente recomendador RA) me ha recomendado en el pasado a otros agentes proveedores PAs. Es similar al valor *Witness Reputation* del modelo FIRE (Hunyh, 2006) Este valor es calculado por el UA cada vez que recibe una recomendación del recomendador, a los fines de converger en la calidad del proveedor. Es un valor dentro del rango [1,10].

- Reputation (R): es el comportamiento esperado del proveedor en un intervalo de [1,10], el agente usuario UA asigna este valor al agente proveedor PA, en la interacción considerando todas las recomendaciones recibidas. Para ello utiliza todos los valores de ar (*Aggregated*

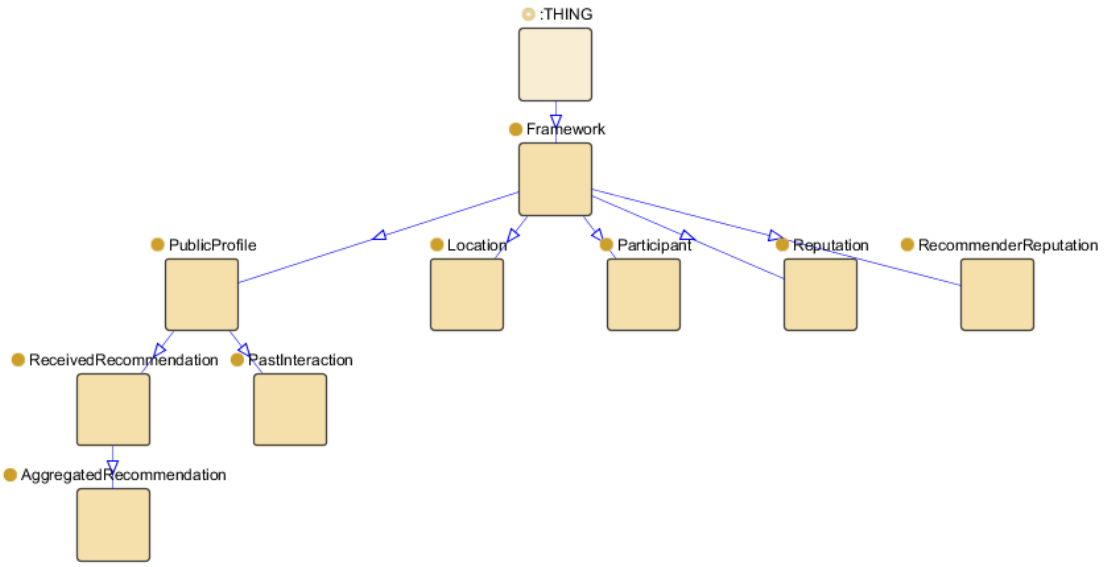

Figura 7.1: Extensión de la ontología para dar soporte a CALoR

Recommendation) recibidos, y los pesas con el valor de rr (*Recommender Reputation*) que él mismo calcula sobre cada agente que le recomienda.

La Fig. 7.1 resume la ontología utilizada por estos agentes para el sistema de reputación en dominios de inteligencia ambiental. Esta ontología es una extensión de la ontología expuesta en (Venturini et al., 2010). Se han agregado ha esta ontología, detallada en el capítulo 5, 6 nuevos conceptos: *PublicProfile, RecommendationReceived, PastInteraction, AggregatedRecommendation, RecommenderReputation y Reputation*, cuyos atributos pueden verse en el Anexo B.

Dado un agente del tipo BDI, se tiene que un agente usuario posee un perfil público compuesto por una tabla de datos local en la cual conserva los datos de las interacciones con otros agentes a lo largo del su vida. En esta se almacenan tuplas del tipo $\langle PA, t, L, ie \rangle$, que son instancias del concepto public profile definido en la ontología, donde PA es el agente proveedor con el que interactuó; t es el tiempo real expresado en fecha y hora con formato (yy,mm,dd,h,m,s); L es la posición (en relación al PA) en donde se encontraba el agente usuario (UA) que había solicitado la recomendación en el momento que se reporta la valoración ie. En cuanto a la localización, se trabaja con coordenadas (latitud, longitud). Y, finalmente, ie (interaction evaluation) es la valoración sobre PA dentro de un rango de satisfacción entre [1,10], en caso de que se haya concretado la interacción. Esta descripción se ejemplifica en la siguiente Fig. 7.2:

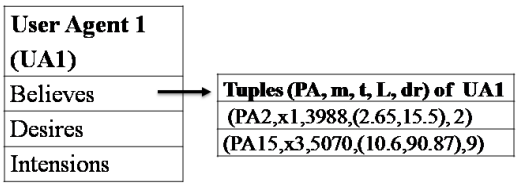

Figura 7.2: Creencias del agente usuario.

7.2 Protocolo del Sistema CALoR

A continuación se describe el protocolo implementado en la provisión de un servicio por un proveedor PA. En la Fig. 7.3 se muestra un diagrama AUML del protocolo de recomendación. Cada fase se explica en detalle en las secciones siguientes. De manera ilustrativa, suponemos un agente solicitante UA, quien pregunta sobre recomendaciones a otros agentes usuarios (UAi), quienes actúan como recomendadores del proveedor PAj. Los pasos a seguir son:

1. El agente solicitante pide recomendaciones a los recomendadores potenciales: UA pregunta a otro UAi sobre recomendaciones de PAj (Fig. 7.4).

2. Los agentes recomendadores recolectan sus interacciones directas pasadas con el proveedor en cuestión: UAi buscan interacciones con el PAj en sus perfiles.

3. Los agentes recomendadores agregan sus propias experiencias de las interacciones directas en la recomendación que envían al agente solicitante: cada slot ar de un agente recomendador UAi se calcula en el concepto recomendación enviado al agente solicitante UA (Fig. 7.5).

4. El agente solicitante recibe y almacena las tuplas *ReceivedRecommendation* de los agentes recomendadores: UA almacena una tupla $\langle UAi, PA, ar\rangle$ por cada recomendación recibida de UAi.

5. El agente solicitante calcula la reputación de los agentes recomendadores: UA computa el valor *RecommenderReputation* (rr) del UAi, quien envió el valor de ar en el slot del concepto recomendación (Fig. 7.6).

6. El agente solicitante sintetiza las recomendaciones de todos los UAi en el valor de reputación R utilizando como peso el valor rr, recientemente calculado por cada UAi (Fig. 7.7).

7. El agente usuario UA toma la decisión de actuar o no con el proveedor PA, en base al resultado de reputación obtenido en R.

8. Si la interacción se lleva a cabo, el agente usuario UA asumiendo su rol de solicitante almacenará una tupla $\langle PA, t, L, ie\rangle$ en su tabla *PastInteractions*.

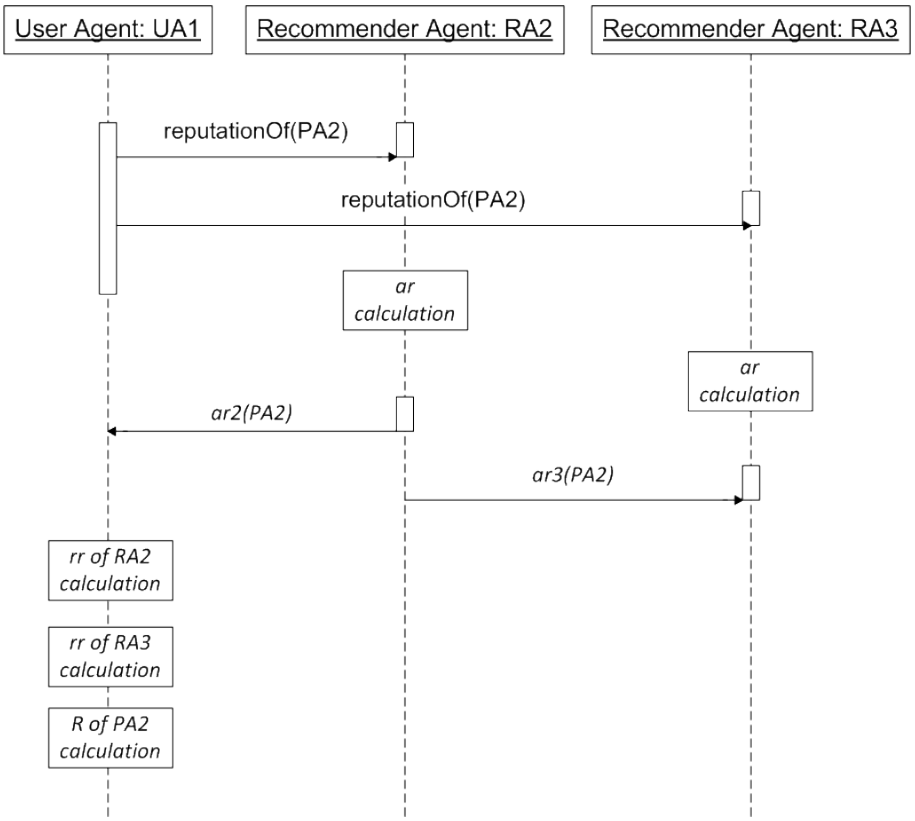

Figura 7.3: Diagrama de interacción AUML del protocolo de recomendación

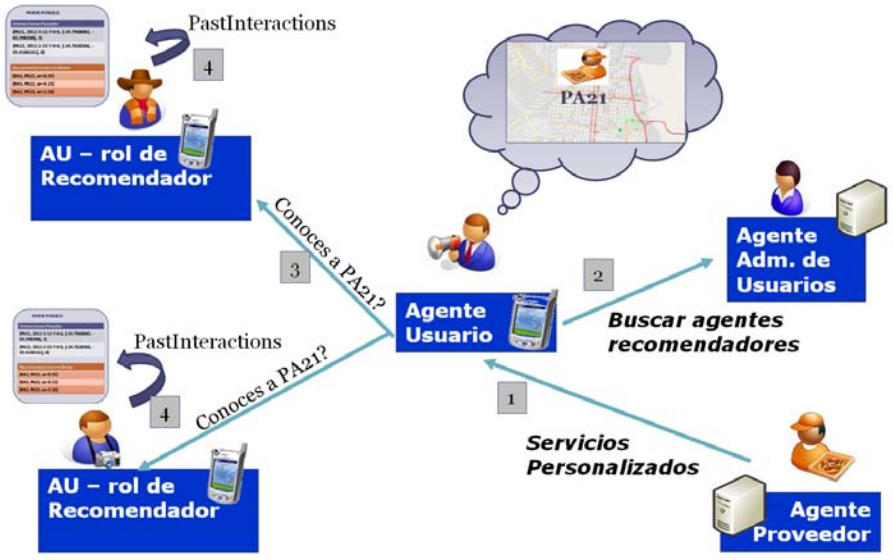

Figura 7.4: El agente usuario solicita recomendaciones a sus contactos. Los agentes recomendadores consultan las interacciones pasadas en su perfil público.

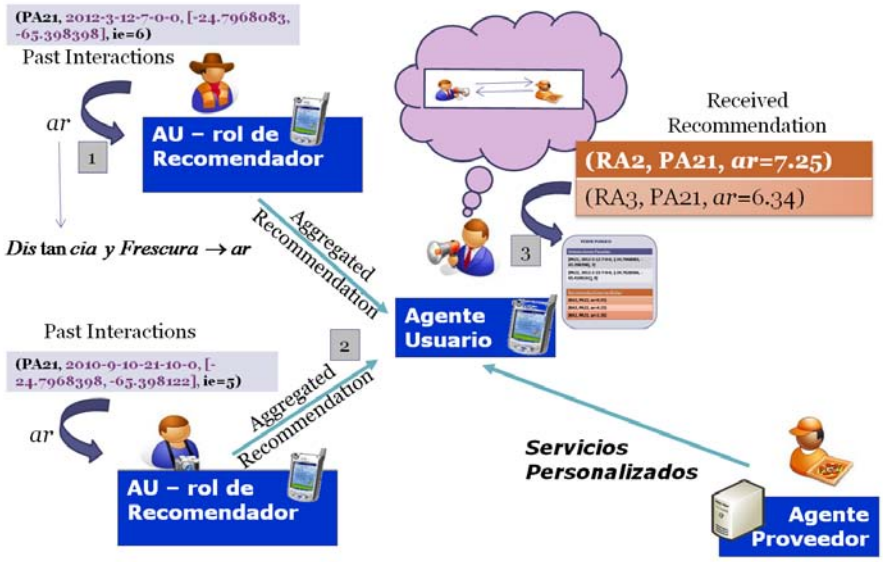

Figura 7.5: Los agentes recomendadores calcúlan el valor aggregated recommendation y se lo envían al agente usuario.

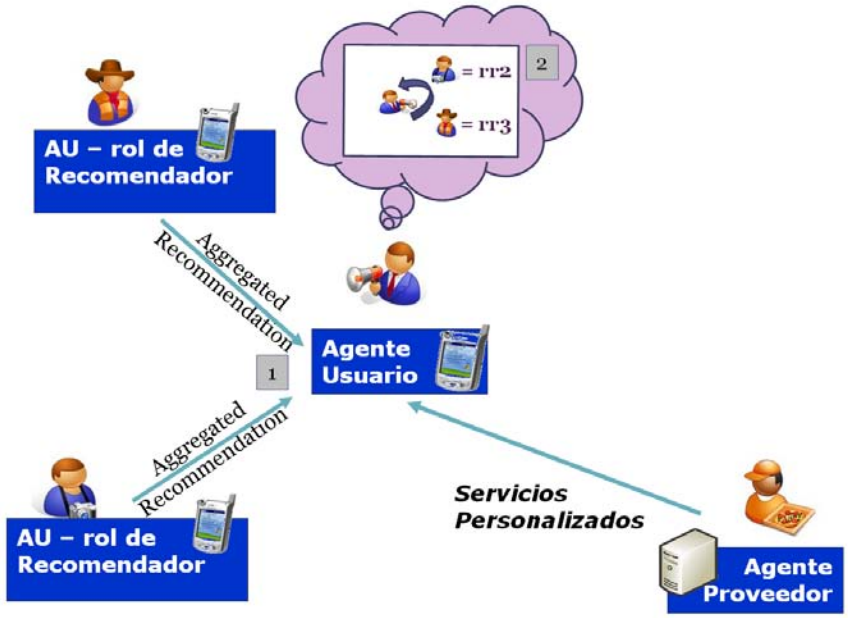

Figura 7.6: El agente usuario computa la reputación rr de los recomendadores que enviaron sus recomendaciones.

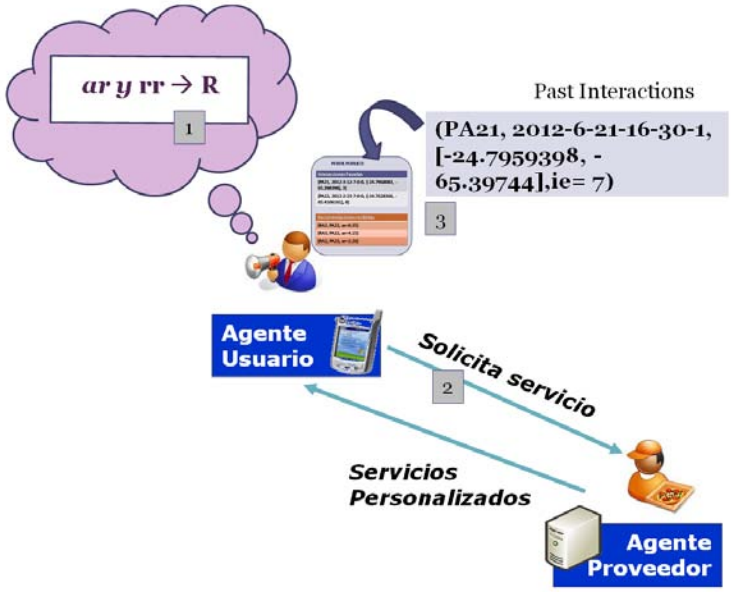

Figura 7.7: El agente usuario obtiene un valor de reputación sobre el proveedor, con el cuál decide si negociar o no.

En la sección siguiente se definen los cálculos necesarios para llevar adelante los pasos 3, 5 y 6 del protocolo.

El agente administrador de usuarios (UMA) juega un rol en el proceso de recomendación. Por ejemplo, cuando el agente solicitante (quien consulta sobre la reputación de otro agente), no tiene información previa sobre agentes de confianza para pedirles recomendación, el UMA puede asumir ese rol filtrando agentes usuarios (UA) que pueden actuar como recomendadores potenciales. Caso contrario, esto puede hacerlo directamente el agente usuario, contactándose directamente con los recomendadores.

7.3 Formalización del Modelo de Reputación

7.3.1 Generando la recomendación para el usuario (Paso 3)

Cada recomendador genera la recomendación a enviar al usuario agregando las interacciones directas que el recomendador tuvo con el proveedor estudiado.

En este paso se trata de formular una ecuación que incorpore situaciones espaciales y temporales de interacciones directas pasadas. Para ello, en primer lugar se consideran la ecuación sugerida en el modelo FIRE (Hunyh, 2006) que incluye el dato de la frescura de la información, y la ecuación de (Bishr and Mantelas, 2008), la cual incluye la distancia espacial.

La *Frescura* de la información se calcula en el modelo FIRE (Hunyh, 2006) utilizando la

siguiente ecuación:

$$f(ie_i) = e^{(-\frac{\Delta t(ie_i)}{\lambda})} \tag{7.1}$$

Donde $\triangle t(ie_i)$ es la diferencia de tiempo entre el tiempo actual y el tiempo en el que se almacenó el valor de direct reputation i. Mientras que λ es quien indica la unidad de tiempo en base a la aplicación, debiendose configurar a priori. Es una unidad de tiempo en la que se expresa el tiempo proporcional que se elige como referencia en el sistema. Por ejemplo, si se quisiera expresar el tiempo en días, proporcional a un año sería 365, pero si el tiempo en días está en proporción a un trimestre, esta unidad de tiempo, descripta con lambda en el modelo, es de 90 días.

Se opta por una función exponencial dado que se asume que nuevos valores de experiencia directa reflejan el funcionamiento real del agente con mayor precisión que los viejos valores de experiencia directa.

Ahora bien, así como el tiempo en que se produce la valoración sobre el proveedor en cuestión, es un factor importante para dar una opinión general, para que la información sea más precisa, hoy por hoy, dada la globalización del mercado, se necesita poder contar con información de la ubicación geográfica en la que se encontraba el agente usuario respecto del agente proveedor, al momento de llevar adelante la transacción. Es por esto que se considera el modelo de (Bishr and Mantelas, 2008) a los fines de medir la distancia entre la posición física de un agente usuario en relación a un agente proveedor al momento de la interacción. En este trabajo los autores presentan una métrica sensible a la distancia geográfica, Ec. 7.2, donde t_m es el rating/valor de confianza de un actor n basada en los ratings dados por los actores que están conectados a él en la red (sus vínculos), y r(i,m) es el rating dado por los k actores sobre la entidad m_h en relación a la distancia c_i. Se observa el uso del logaritmo en base 10 como penalización de la función puesto que llega un punto (dependiendo del contexto de aplicación serán 50, 100, 1000 o más metros o kilómetros) en que esa distancia es tan grande, tan lejana, que se hace insignificante entre los agentes que interactúan.

$$t_m = \sum_{i=1}^{k} \frac{t_i \cdot r(i,m)}{log(c_i)} \, donde \, c_i > 1 \tag{7.2}$$

El uso de una distancia logarítmica podría ayudar a dar mayor peso a pequeñas diferencias si la distancia desde la que se interactúa es pequeña, mientras que se daría menor relevancia a las mismas diferencias si la distancia desde la que se interactúa es mayor. Una fórmula clásica para representar la idea planteada es la función $1/log(x)$. Por ejemplo, al evaluar la densidad de una red de datos: la cantidad de paquetes perdidos es mayor para un cable largo que para una cable corto (distancia corta); el logaritmo será 1 si el cable está cerca del origen. Es decir, el uso de logaritmos nos permite delimitar las distancias de acuerdo al dominio, lo cual es de particular interés en los entornos de AmI. Pues el dominio de interacción puede ser regional, nacional o internacional. Por lo tanto, en el cálculo de la reputación de la Ec. 7.3, consideramos el peso del logaritmo para la distancia. Con este cálculo, se obtiene un valor final de recomendación que no se ve fuertemente penalizado por la influencia de la distancia, no hay una relación lineal.

$$w = \frac{1}{a + b \, \cdot \, log(d)} \tag{7.3}$$

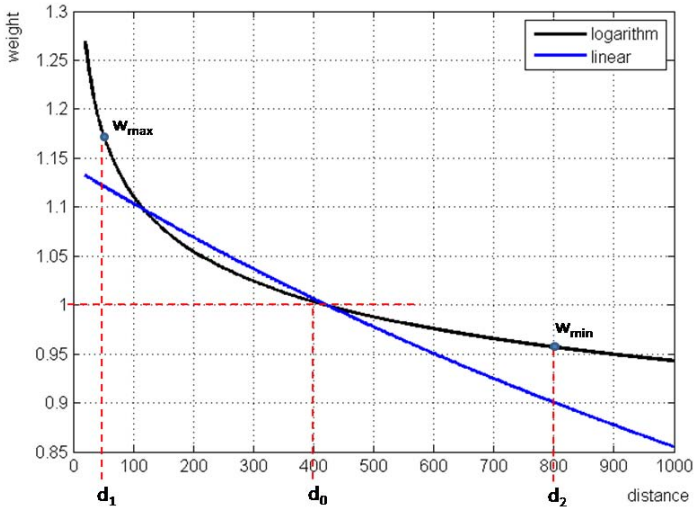

Figura 7.8: Función de pesos del logaritmo.

A partir del gráfico del peso del logaritmo (Fig. 7.8), se observa como se suavizan las variaciones bruscas de la ecuación. La línea negra representa el logaritmo y la línea azul es una función lineal. Veamos un ejemplo. Se supone la siguiente configuración de distancia: $d_1 = 40$, $d_2 = 800$ y $d_0 = 420$ (d_0 es elegido como el valor central de la distancia). Obsérvese cómo para $d_2 = 800$ el logaritmo cae más suavemente que la función lineal para el mismo valor. Además, para la distancia $d_1 = 40$, la diferencia entre ambas líneas es más corta; es decir, para distancias cortas el peso de w es mayor. Para el valor de distancia igual a 800, se considera que, puesto que la distancia es mayor, la confianza es menos relevante.

Para obtener los parámetros a y b se utilizan dos relaciones:

1. La relación entre w_{max} y w_{min}: el peso mínimo es el porcentaje del peso máximo;
2. Una normalización: es el punto en donde se asume que w es igual a 1;

Ambos casos se muestran en las ecuaciones 7.4 y 7.5.

$$w = 1 = \frac{1}{a + b \ \cdot \ log(d_0)} \tag{7.4}$$

$$\frac{w_{max}}{w_{min}} = \frac{1}{a + b \cdot \ log(d_1)} \cdot \frac{1}{\frac{1}{a + b \cdot \ log(d_2)}} = \frac{a + b \cdot \ log(d_2)}{a + b \cdot \ log(d_1)} \tag{7.5}$$

Finalmente, el uso de una función logarítmica (Fig. 7.9) nos permite suavizar el crecimiento exponencial de la distancia.

A partir de las ecuaciones 7.1 y 7.2 proponemos una función que evalúa la experiencia directa que tuvo el recomendador, y que permitirá producir la recomendación pedida en

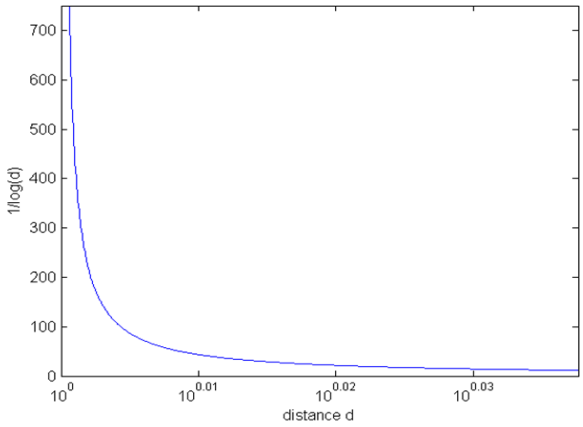

Figura 7.9: Función logaritmo.

relación a cuan reciente fue dicha interacción y su relación geoespacial con el lugar físico del proveedor de productos/servicios en el momento del intercambio. A esta métrica la denominamos *AggregatedRecommendation*, y es una suma ponderada de las interacciones del recomendador con el proveedor de interés (Ec. 7.11), normalizada a un valor relativo de referencia. La frescura de la interacción $f_i(ie_i)$ señala cuan reciente es la interacción, siendo 1 cuando más reciente es la misma. El ar será mayor cuando más reciente y más cerca se haya producido la interacción.

$$ar = \frac{a + b\,log[\alpha]}{n} \sum_{i=1}^{n} sr_i \tag{7.6}$$

Donde:

- ar (*AggregatedRecommendation*) es el valor final dado por el agente recomendador sobre el PA. Es la imagen de la reputación que posee el agente recomendador sobre el agente proveedor.

- *i* es el número de interacción entre el agente recomendador con el agente proveedor

- a *y* b son parámetros de ajuste ad-hoc necesarios por la función de distancia para normalizar los valores de ar.

- α es la distancia media en el dominio. Por ejemplo, si estamos en un entorno de AmI donde las distancias se encuentran en un rango [0..1000], el valor α podría ser 500.

- *n* es el número total de interacciones entre el agente recomendador y el agente proveedor.

- sr_j (single recommendation): Representa la relación entre el ie (interaction evaluation) del agente proveedor y la distancia y frescura de la interacción i correspondiente. Se calcula en función de la Ec. 7.7.

$$sr_j = \frac{ie_i \cdot f(ie_i)}{a + b\,log[c_i(ie_i)]} \tag{7.7}$$

Donde:

- ie_i es el valor ie (interaction evaluation) del agente recomendador sobre el agente proveedor.

- $f(ie_i)$ es la medida de la frescura de la interacción i tal como se la calcula en el modelo FIRE (Hunyh, 2006).

- c_i es la distancia (cercanía) entre la posición del agente recomendador j y la posición del agente proveedor en el momento de la interacción i

7.3.2 Generando la reputación del recomendador (Paso 5)

Luego de cada interacción (cuando se produce un nuevo valor de interaction evaluation, ie), el agente solicitante debe considerar cuan buenas han sido las recomendaciones pasadas. A estos efectos, es necesario comparar por cada agente recomendador, la diferencia entre el slot ar de las recomendaciones recibidas que correspondan sobre el proveedor, y el valor ie de la interacción final con el proveedor. Se elige una función exponencial, tal como hicieron los autores del modelo FIRE (Hunyh, 2006), para obtener valores de la reputación de los agentes recomendadores. Esta función exponencial refleja que cuanto menor es la diferencia (entre el ie producido por el agente solicitante y el ar enviado por el recomendador previamente), más confiable es el recomendador.

$$rr = 10 * e^{-\sigma} \tag{7.8}$$

En el caso en que el exponente sea cero (0), la reputación del recomendador valdrá 10, es decir, el agente recomendador hizo una recomendación perfecta en el pasado (disponible en el rango de 0 a 10) y, por lo tanto, sus recomendaciones futuras tendrán mayor peso que las enviadas por otros. El valor de σ_i está dado por:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i}^{n}(\frac{(ar_i - ie_i)^2}{ie_o^2})} \tag{7.9}$$

Con,

- ar_i es el slot ar de una *ReceivedRecommendation* (recomendación recibida) sobre la interacción i.

- i es el número de recomendación

- $ie_{i,j}$ es el valor de evaluación de la interacción i entre el agente recomendador y el agente proveedor

- ie_0 es el primer valor de ie (valor inicial por defecto utilizado) utilizado para normalizar los valores, midiendo de esta forma errores relativos a este valor inicial

- n es la cantidad de recomendaciones recibidas

De esta forma, en caso de que σ sea cercano a 0, la reputación del recomendador rr tenderá a 1, lo cual está reflejando que ha sido un buen recomendador en el pasado, teniendo más peso la información que éste nos envíe en el futuro.

7.3.3 Cálculo de la reputación final del proveedor (Paso 6)

Uso de la reputación del recomendador para agregar las recomendaciones recibidas de distintos recomendadores sobre un mismo proveedor.

Utilizando la reputación de los recomendadores, se puede calcular cuál es el valor final de reputación R de los agentes proveedores, como muestra 7.15. Se calcula la Reputación de un agente proveedor PA como una suma ponderada de las recomendaciones recibidas (sus slots ar), donde el peso de la ponderación es la reputación del recomendador (calculada en el paso 5):

$$R = \frac{\sum_j^m ar_j \cdot rr_j}{\sum_j^m rr_j} \tag{7.10}$$

Donde,

- *R* es la reputación de un agente proveedor de acuerdo a las recomendaciones recibidas. Es un valor entre 1 y 10

- ar_j es el slot ar de una recomendación recibida del agente recomendador j

- rr_j es la reputación del agente recomendador j, calculada en el paso 5 cada vez que se recibe una nueva recomendación

- *m* es el número total de recomendaciones recibidas por cada intento de interacción con el agente proveedor.

El éxito del sistema de reputación se determina en base a la similaridad de los valores de R con el comportamiento real del agente proveedor, tal como se prueba en las secciones siguientes.

7.4 Validación de CALoR a través de un Caso de Uso

En esta sección se muestra un caso de uso para ilustrar y validar informalmente la funcionalidad de la arquitectura, el protocolo y los cálculos propuestos. El dominio escogido para validar el modelo es el de los servicios turísticos en una ciudad, específicamente usaremos como escenario de prueba la ciudad de Salta, Argentina. Se considera una serie de proveedores de servicios listados en la Tabla N° 7.1. De cada proveedor se sabe su identificación, sus coordenadas geográficas y la calidad real de servicios que provee. Suponemos que un grupo de personas transitan por la ciudad. Cada uno cuenta con un dispositivo móvil provisto de un agente personal, cuyos roles posibles son los de: agente usuario o agente recomendador. En la Fig. 7.10 se puede ver la ubicación de los proveedores en el mapa de la ciudad. En orden a que sea un dominio de inteligencia ambiental significativo para nuestro propósito, asumimos que las interacciones con estos sitios turísticos se producen de una manera más precisa cuando los agentes usuarios visitan los sitios desde ubicaciones más cercanas.

Tabla 7.1: Ids de Proveedores, ubicación y nombres

Proveedor	**Id del Agente**	Longitud	Latitud
Museo Arqueológico de Alta Montaña (MAAM)	PA21	-24.7889941	-65.4111281
Museo Antropológico Juan Martín Leguizamón	PA22	-24.7863838	-65.3975242
Museo de Ciencias Naturales - UNSA	PA23	-24.7961433	-65.4023092

7.4.1 Configuración Inicial del caso de uso

Los agentes involucrados en el proceso de recomendación del caso de uso propuesto, son los siguientes:

- UA1: agente usuario 1, quien decide iniciar el pedido de recomendación sobre el PA21.
- PA21: agente proveedor del sobre el cual UA1 va a calcular la reputación.
- UA2: agente usuario 2, quien actúa como recomendador de UA1 acerca de PA21, y quien interactuó en el pasado una vez con PA21, PA22 y PA23.
- UA3: agente usuario 3, quien actúa como recomendador de UA1 acerca de PA21, y quien interactuó previamente dos veces con PA21 y una vez con PA22.

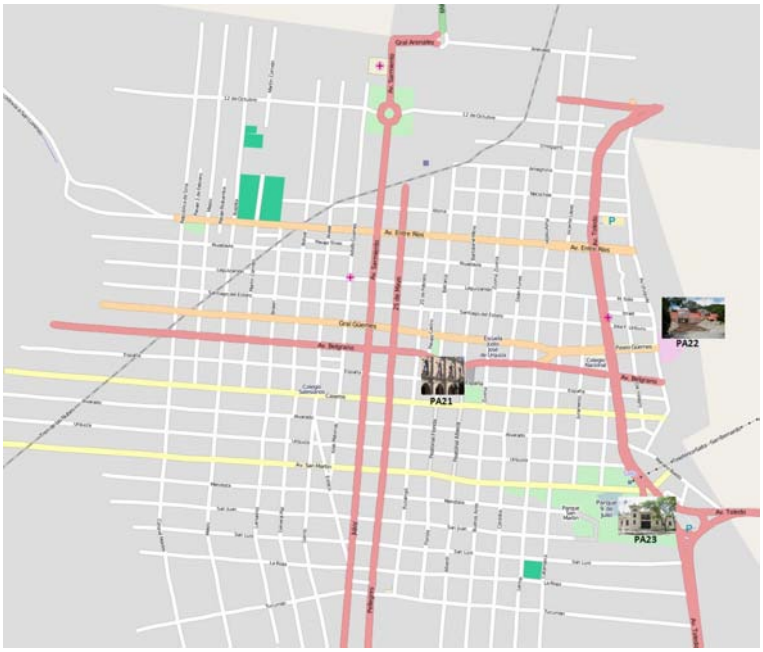

Figura 7.10: Proveedores de servicios en la ciudad turística de Salta.

Las tablas 7.2 y 7.3 de los perfiles públicos iniciales de los usuarios $UA2$ y $UA3$ muestran en detalle las interacciones mencionadas anteriormente: evaluación de la interacción (ie), ubicación donde se produjo la interacción (longitud y latitud) y frescura de la interacción (tiempos):

Tabla 7.2: Interacciones de UA2 con los proveedores

Proveedor	ie	Longitud	Latitud	Tiempo
PA21	8	-65.4130523	-24.782280	2010,12,22,7,0,0
PA22	3	-65.4027362	-24.7839892	2010,7,23,7,0,0
PA23	9	-65.4455745	-24.7760736	2011,12,30,7,0,0

Tabla 7.3: Interacciones de UA3 con los proveedores

Proveedor	ie	Longitud	Latitud	Tiempo
PA21	7	-65.4313467	-24.8004947	2010,7,6,7,0,0
PA21	3	-65.4537264	-24.8141731	2007,9,3,7,0,0
PA22	9	-65.4338991	-24.8044136	2012,1,30,7,0,0

Adicionalmente, asumimos que un agente usuario $UA1$ actuando como solicitante, ha recibido previamente recomendaciones de los agentes $UA2$ y $UA3$ sobre $PA22$ y $PA23$, correspondientes a las interacciones que han tenido con esos proveedores. La Tabla 7.4 muestra las recomendaciones recibidas de los usuarios $UA2$ y $UA3$ sobre $PA22$ y $PA23$, y los resultados de las interacciones de $UA1$ con $PA22$ y $PA23$ que tuvieron lugar después de esas

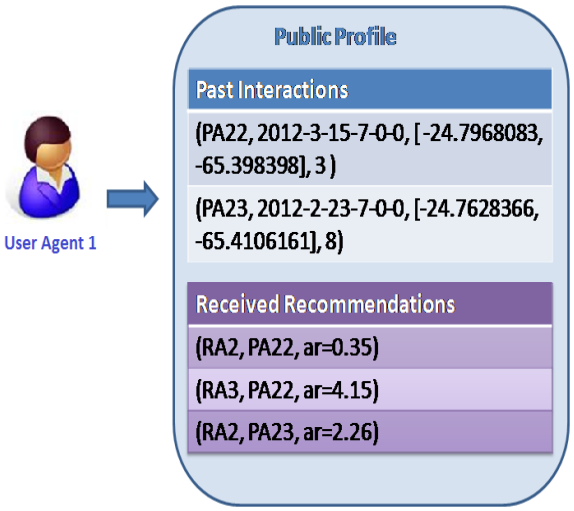

Figura 7.11: Perfil público de UA1.

recomendaciones, las cuales conforman el perfil público del $UA1$ (ver Fig. 7.11):

Tabla 7.4: Recomendaciones recibidas por UA1 de UA2 y UA3

Recomendador	Proveedor	ar
UA2	PA22	0.350227674
UA2	PA23	2.265593464
UA3	PA22	4.152888877
UA3	PA22	4.152888877

El valor ar de la última fila de la Tabla 7.5 se calcula usando los valores ie de ambas interacciones de $UA2$ con $PA22$ (de acuerdo a los tiempos mostrados en la tabla anterior 7.2) mientras que los demás valores de ar se computan a partir de los valores simples de ie.

Tabla 7.5: Interacciones previas de UA1 con los proveedores

Proveedor	ie	Longitud	Latitud	Tiempo
PA23	8	-65.4106161	-24.7628366	2012,2,23,7,0,0
PA22	3	-65.398398	-24.7968083	2012,3,15,7,0,0

7.4.2 Agentes Ejecutando una Instancia del Protocolo en el Caso de Uso

El protocolo se inicia cuando el agente usuario $UA1$ envía mensajes solicitando recomendaciones (ver Fig. 7.12) a $UA2$ y $UA3$ sobre $PA21$, ambos agentes han interactuado previamente con

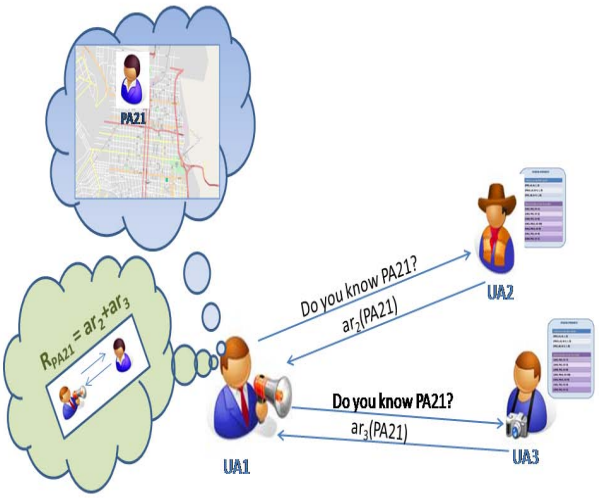

Figura 7.12: UA1 pide recomendaciones.

el $PA21$, y, también previamente, ambos han realizado recomendaciones a $UA1$ sobre $PA22$ y $PA23$.

Cada agente actuando como recomendador ($UA2$ y $UA3$) recibe un mensaje de solicitud de recomendación e inicia un proceso de cálculo interno a los fines de proveer una recomendación sobre PA21 (denotada como ar). Este cálculo interno se corresponde con el paso 3 del protocolo:

$$ar = \frac{a + b\,log[\alpha]}{n} \sum_{i=1}^{n} sr_i \tag{7.11}$$

Donde cada sr_j se calcula como:

$$sr_j = \frac{ie_i \cdot f(ie_i)}{a + b\,log[c_i(ie_i)]} \tag{7.12}$$

Por lo tanto, reemplazando las variables correspondientes con los valores de $UA2$ recomendando a $PA21$:

- a = -1.142
- b = 0.857
- Interacción: $ie_1 = 8$

Debido a que $UA2$ ha interactuado solo una vez con $PA21$, se tiene un valor de sr_j de 0.94 (y por lo tanto n=1). Luego, se reemplazan las variables con los valores correspondientes para obtener ar:

- a = -1.142

- $b = 0.857$
- Interacción pesada: $sr_1 = 0.94$
- $n = 1$
- $\alpha = 400$

Por lo tanto la recomendación que $UA2$ envía a $UA1$ sobre $PA21$ es un valor ar de: 1.033

Luego se repiten los mismo cálculos para obtener el valor de ar enviados por $UA3$ a $UA1$ sobre $PA21$. En este caso se tienen dos interacciones de $UA3$ con $PA21$:

- $a =$ -1.142
- $b = 0.857$
- Interacción 1: $ie_1 = 3$
- Interacción 2: $ie_2 = 7$

Se obtienen dos valores de sr_j de 0.0155 y 3.51016 (con n=2). A continuación, se reemplazan las variables con los valores correspondientes a los fines de obtener ar:

- $a =$ -1.142
- $b = 0.857$
- $n = 2$
- $\alpha = 400$
- Interacción Pesada 1: $sr_1 = 0.0155$
- Interacción Pesada 2: $sr_2 = 3.51016$

Por lo tanto, la recomendación que $UA3$ envía a $UA1$ sobre $PA21$ es un valor ar de: 1.9179.

$UA2$ y $UA3$ luego responden el mensaje de solicitud de recomendación enviado por $UA1$ con los valores de ar recientemente calculados, de una manera completamente transparente para el dueño del dispositivo móvil (cuyo agente está actuando bajo el rol de solicitante). Mientras $UA1$ recibe las recomendaciones de $UA2$ y $UA3$, éste calcula la reputación de cada recomendador, desde un nivel relativo de lo sucedido en las recomendaciones previas de $UA2$ y $UA3$ sobre $PA22$ y $PA23$. Este cálculo corresponde al paso 5 del protocolo.

$$rr = 10 * e^{-\sigma} \tag{7.13}$$

Donde σ se calcula como:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i}^{n}(\frac{(ar_i - ie_i)^2}{ie_o^2})} \tag{7.14}$$

Entonces, reemplazando las variables correspondientes con los valores de $UA2$ recomendando $PA22$ y $PA23$:

- Recomendaciones sobre $PA22$:
  - $ar_1 = 0.35$
  - $ie_1 = 3$
- Recomendaciones sobre $PA23$:
  - $ar_2 = 2.26$
  - $ie_2 = 8$
- $ie_o = 3$
- $n = 2$

Se obtiene un valor de σ de 4.4704, lo que da un valor de reputación del recomendador (rr) para el $UA2$ de 0.1144.

En este caso $UA3$ recomendó dos veces a $PA22$, se tienen entonces:

- Recomendación 1 sobre $PA22$
  - $ar_1 = 4.15$
  - $ie_1 = 3$
- Recomendación 2 sobre $PA22$
  - $ar_2 = 4.15$
  - $ie_2 = 3$
- $ie_o = 3$
- $n = 2$

Y se obtiene un valor de σ de 1.15, lo que da un valor de (rr) para el $UA3$ de 3.16.

Ya que se tienen los valores de reputación de ambos recomendadores (rr), se pueden agregar los valores de las recomendaciones recibidas (ar) al valor de reputación final R del proveedor $PA21$. Este cálculo se corresponde con el paso 6 del protocolo. En este paso se computa R como:

$$R = \frac{\sum_j^m ar_j \cdot rr_j}{\sum_j^m rr_j} \tag{7.15}$$

Por lo tanto, reemplazando las variables correspondiente con los valores de rr sobre $UA2$ y $UA3$, y los valores de ar recibidos de $UA2$ y $UA3$ sobre $PA21$:

- $ar_0 = 1.033$

- $ar_1 = 1.9179$
- $rr_0 = 0.1144$
- $rr_1 = 3.191$
- $m = 2$.

De este modo, se obtiene para el caso de uso planteado un valor de reputación R para el proveedor $PA21$ de 1.8872. Lo cual es un valor razonable dada la pequeña cantidad de interacciones con las que se han calculado las recomendaciones.

7.5 Implementación de CALoR: Sistema de Reputación basado en Contexto y Localización

La construcción de los agentes se lleva adelante utilizando JAVA y la librería JADE [1] , específica para la construcción de agentes. Se implementan los cinco tipos de agentes descritos en la arquitectura del sistema junto a sus comportamientos detallados en (Venturini et al., 2012b). En particular, para llevar adelante el modelo se implementan dos comportamientos para los agentes clientes. Uno de ellos es el comportamiento *Ask for reputation*, que para nuestro ejemplo será tarea del $UA1$ que desea conocer la reputación del $PA21$, y el comportamiento *Response reputation*, el cual estará activo para todos los demás agentes UA que hayan interactuado con $PA21$, suponemos que son $UA2$, $UA3$ y $UA4$. Los agentes se comunican utilizando los mensajes que se implementan del protocolo FIPA, sobre una ontología construida en Protégée[2]. Dicha ontología es una extensión de la ontología presentada en este artículo sobre reputación y la ontología que da soporte a escenarios de AmI expuesta en el artículo (Venturini et al., 2010). El $UA1$ envía un mensaje FIPA del tipo *Request* a cada agente de la lista, indicando sobre que agente proveedor quiere recibir información. Los demás agentes usuarios, al recibir este mensaje, inician un proceso de cálculo del valor del slot aggregated recommendation (ar) que generó en el pasado la interacción de ellos con dicho proveedor, aplicando las ecuaciones descritas en el modelo, de acuerdo a la información registrada en sus perfiles privados. Los agentes recomendadores envían al $UA1$ un mensaje tipo *Inform* cuyo contenido es el ar. En caso de que el $UA1$ tomara la decisión de interactuar con el PA, deberá emitir un mensaje *Request* cuyo destinatario será el $PA21$. A partir de esta fase, $UA1$ y $PA21$ intercambiarán mensajes para llevar adelante el proceso de negociación.

En resumen, se han implementado 2 comportamientos: AskForReputation.class y ResponseReputation.class, 1 archivo de XML para los perfiles públicos (publicProfile.xml) y 6 clases: publicProfile.class, ParsePublicProfile.class, PastInteraction.class, ReceivedRecommendation.class, AggregatedRecommendation.class y Reputation.class. Los ejecutable (.class) están disponibles via web para cualquiera que desee usarlos.

La Fig. 7.13 es un ejemplo de la comunicación entre los agentes de JADE durante el proceso de solicitud de recomendaciones.

[1] http://jade.tilab.com/
[2] http://protege.stanford.edu/

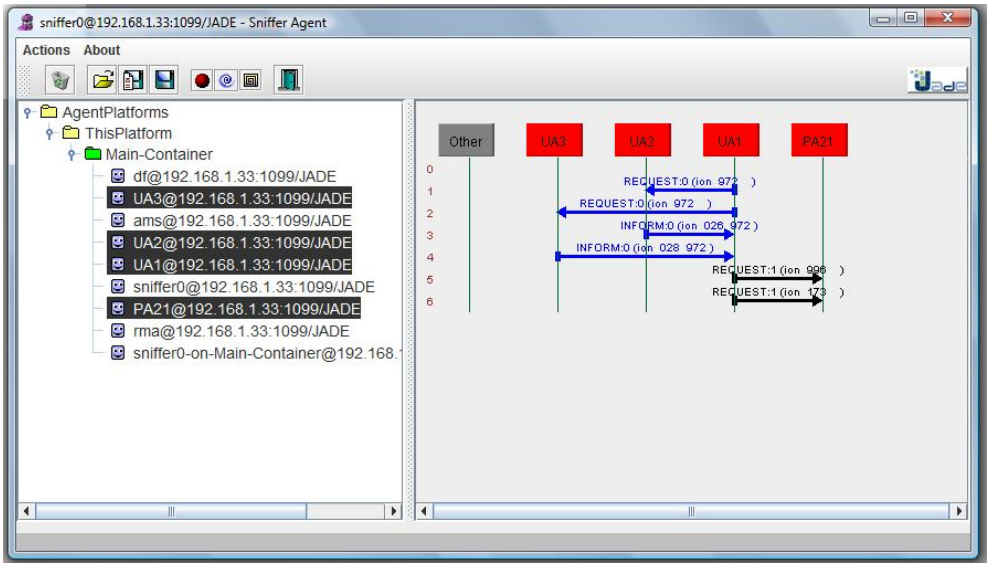

Figura 7.13: Agentes de JADE activos durante el proceso de recomendación

7.6 Evaluación del Modelo a través de Simulaciones

Cada simulación incluye 20 agentes usuarios y 5 agentes proveedores. Los agentes usuarios pueden realizar consultas sobre cualquier proveedor. Cualquier agente usuario puede, en cualquier momento, asumir el rol de recomendador. Inicialmente, los agentes no tienen información respecto a recomendaciones anteriores, pero si poseen información respecto de sus propias interacciones pasadas (una tabla con 50 interacciones iniciales de todos los agentes). Esta tabla contiene el dato del ie (evaluación de la interacción) de los agentes usuarios UA con los agentes proveedores PA. Esta tabla se creó con la siguiente información:

- UA: un número aleatorio obtenido de una distribución uniforme entre [1..20],
- PA: un número aleatorio obtenido de una distribución uniforme entre [1..5],
- ie: un número aleatorio obtenido de una distribución normal, con una media y una varianza correlativa a la calidad de servicio ofrecida por el proveedor,
- $frescura$ ($freshness$): un número aleatorio obtenido a través de una distribución gamma con $\alpha = 0{,}8$ y $\beta = 0{,}1$ para proveer valores de frescura entre [0..1],
- distancia de la interacción (interaction $distance$): un número aleatorio obtenido de una distribución gamma con $\alpha = 16$ y $\beta = 32$ para proveer valores de distancia entre [0..800].

Se asume que cada proveedor tiene una calidad inherente que determina la calidad de los servicios provistos en las interacciones ie. Esta calidad se representa mediante una distribución normal $N(\mu_p, \sigma_{p,c})$, donde la media representa la calidad del servicio provista $\mu_p \epsilon [1.,10]$, la cual depende de la distancia desde la que se proveyó el servicio, mientras que la varianza representa la variabilidad de esa calidad $\sigma_{p,c} \epsilon [0{,}6.,1{,}6]$. Diferentes valores de la varianza de ie pueden producir distintos tipos de experimentos, se pueden representar de esa manera diferentes

escenarios de inteligencia ambiental. Por ejemplo, se puede observar en la Tabla 7.6 la media y la varianza de los 5 agentes proveedores de la simulación. Allí, los valores de varianza de esa distribución normal para producir valores particulares de ie son lo suficientemente altos para definir un escenario de proveedores con la suficiente inestabilidad (alta variabilidad) en la generación de servicios.

Tabla 7.6: Calidad de servicio de los proveedores para ofrecer una evaluación de interacción, ie

Proveedor	ie media	ie varianza
1	3.00	1.217
2	5.75	1.422
3	8.13	0.831
4	7.17	1.344
5	4.33	0.992

Mientras que se ejecutan los agentes, el proceso de recomendación evoluciona. Se generan 50 simulaciones. Cada simulación consiste en un ciclo de 15 rondas de recomendaciones solicitud/respuesta. En cada ronda se selecciona de forma aleatoria (distribución uniforme) un agente usuario que actúa como solicitante, UA, y un agente proveedor PA. Asumimos que el agente usuario elegido siempre está propenso a pedir recomendaciones sobre el agente proveedor elegido. Por lo tanto, el agente solicitante, UA, mira su tabla de interacciones iniciales en busca de otros agentes que hayan interactuado previamente con el PA. Como resultado de la búsqueda en la tabla de interacciones iniciales, se obtiene un vector de recomendadores. Luego, los agentes recomendadores pueden calcular los valores de ar correspondientes sobre el proveedor PA con el cual desea interactuar el agente usuario UA. Con esos valores de ar el agente solicitante podrá finalmente calcular el valor de reputación R del proveedor PA.

Por lo tanto, en el cálculo de R, el agente solicitante debe considerar por cada agente recomendador:

- El tiempo donde el agente recomendador envía su valor para ar.
- El valor de aggregated recommendation (ar) (Ec. 7.11) la cual incluye:
  - La $frescura$ de la interación entre el recomendador y el proveedor
  - La $distancia$ entre ambos agentes interactuantes
- La reputación del recomendador, rr, Ec. 7.13

Tal como se ha explicado para el valor de varianza de ie, definiciones diferentes de los parámetro gamma de la $frescura$ y la $distancia$ pueden producir diferentes tipos de experimentos, los cuales pueden representar nuevos escenarios de Inteligencia Ambiental.

7.7 Evaluando el Rol de la Distancia

Con los valores de configuración inicial explicados anteriormente, el objetivo planteado es ver cómo la distancia influye en la reputación final del proveedor, comparando cuán cerca está la

reputación R de la calidad real del servicio en dos escenarios:

- Considerando la distancia entre los agentes que interactuan.
- Ignorando la distancia entre los agentes que interactuan. Con el propósito de calcular la reputación sin distancia, se sustrae la $distancia$ de la Ec. 7.11, y la Ec. resultante es:

$$ar_j = \frac{1}{n}\sum_{i=1}^{k} ie_i(n,m) \cdot f_i(ie_i) \tag{7.16}$$

La Fig. 7.14 muestra cuan cerca está el valor de reputación (R) de la media de la evaluación de las interacciones generadas, los valores ie, por cada proveedor. La Tabla 7.10 muestra el error relativo de esas predicciones considerando e ignorando la distancia. Se observa cómo el valor de reputación (R) considerando la distancia es más cercano a la media de ies, con un error promedio de 0.04, mientras que las simulaciones ignorando la distancia arrojan un error promedio de 0.24.

Tabla 7.7: Error relativo de las predicciones basado en la reputación considerando e ignorando las distancias

Proveedor	Error relativo ignorando	Error Relativo considerando
1	0.196667	-0.03
2	0.373217	0.171304348
3	0.224846	0.045756458
4	0.208926	-0.00223152
5	0.206236	0.047113164

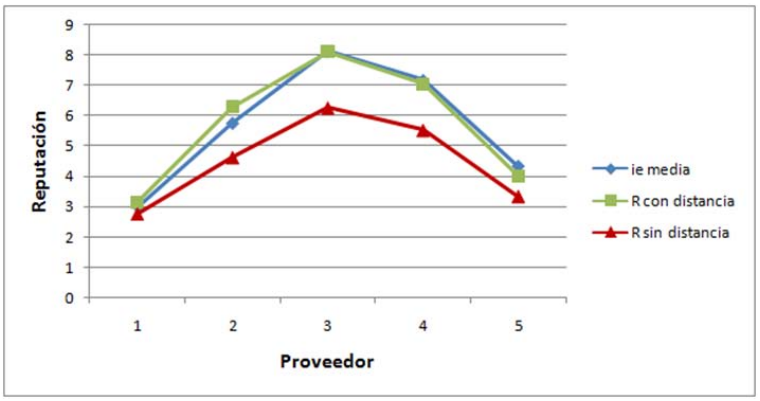

Figura 7.14: Reputación (R) de cada proveedor y media de las evaluaciones de las interacciones (ie) considerando e ignorando la distancia

Si solo nos enfocamos en el proveedor $PA2$, se puede dibujar el histograma de las evaluaciones de las interacciones (ie) con $PA2$ (dibujados en azul en las Fig. 7.15 y 7.16), y luego puede verse el promedio de las imágenes de reputación final (R) de $PA2$ con todos los usuarios

(dibujado en rojo) durante toda la simulación. Mientras que la Fig. 7.15 muestra la simulación considerando la distancia, la Fig. 7.16 muestra la simulación ignorando la distancia. Podemos observar como la reputación del $PA2$ tiene una probabilidad de suceder cercana al $15\,\%$ cuando se incluye la distancia en los cálculos, mientras que la reputación de $PA2$ cuando la distancia es ignorada tiene una probabilidad cercana al $5\,\%$.

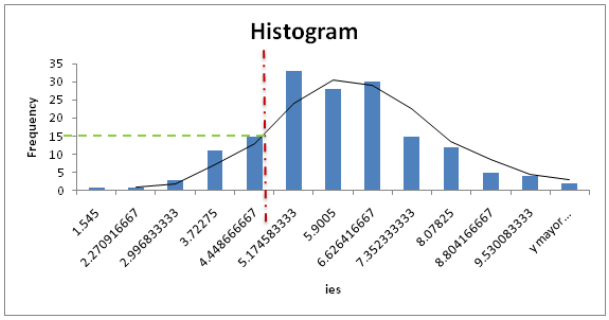

Figura 7.15: Histograma de ies de PA2 y su reputación final considerando la distancia.

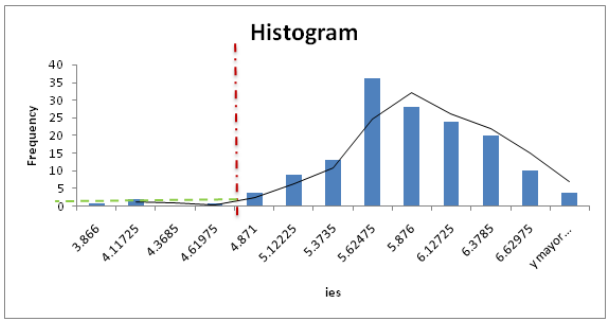

Figura 7.16: Histograma de ies de PA2 y su reputación final ignorando la distancia.

Estos resultados muestran que, si la evaluación de las interacciones dependen de la distancia existente entre los agentes que interactúan, las ecuaciones propuestas en este capítulo pueden mejorar los modelos de confianza que ignoran dicho factor.

En la Fig. 7.17 se muestra la evolución de la reputación de cada uno de los proveedores cuya calidad de servicio se expuso en la Tabla 7.6 a lo largo de 50 interacciones, comparando R considerando e ignorando la distancia, en relación al valor real de calidad de servicio provisto evaluado directamente (ie). Se observa como evolucionan desde cero los valores de reputación R para ambos casos analizados. Con línea roja se muestran los valores de R considerando la distancia y en verde los valores de R ignorando la distancia. Por otra parte, la línea azul refleja los valores de evaluación directa ie a lo largo de las simulaciones mostrando un escenario con proveedores con calidad de servicio mínimamente variables. En el gráfico del comportamiento del proveedor 1 cuya calidad de servicio es baja, rondando los 3 puntos, ambas líneas representativas de R tienen un comportamiento similar. Para los proveedores 2, 3 y 4 con valores de ie entre 6 y 8 puntos la diferencias en las curvas es más marcada. En particular para los agentes proveedores 2 y 3 el sistema de reputación considerando la distancia sobreestima la calidad de servicio del agente. En el último caso, el de la reputación del agente proveedor 5, se observa como el sistema

obtiene valores bastante más precisos, siguiendo casi en paralelo al verdadero comportamiento del proveedor. Los gráficos se generaron a partir de los datos obtenidos durante las simulaciones, expuestos en la Tabla 7.8.

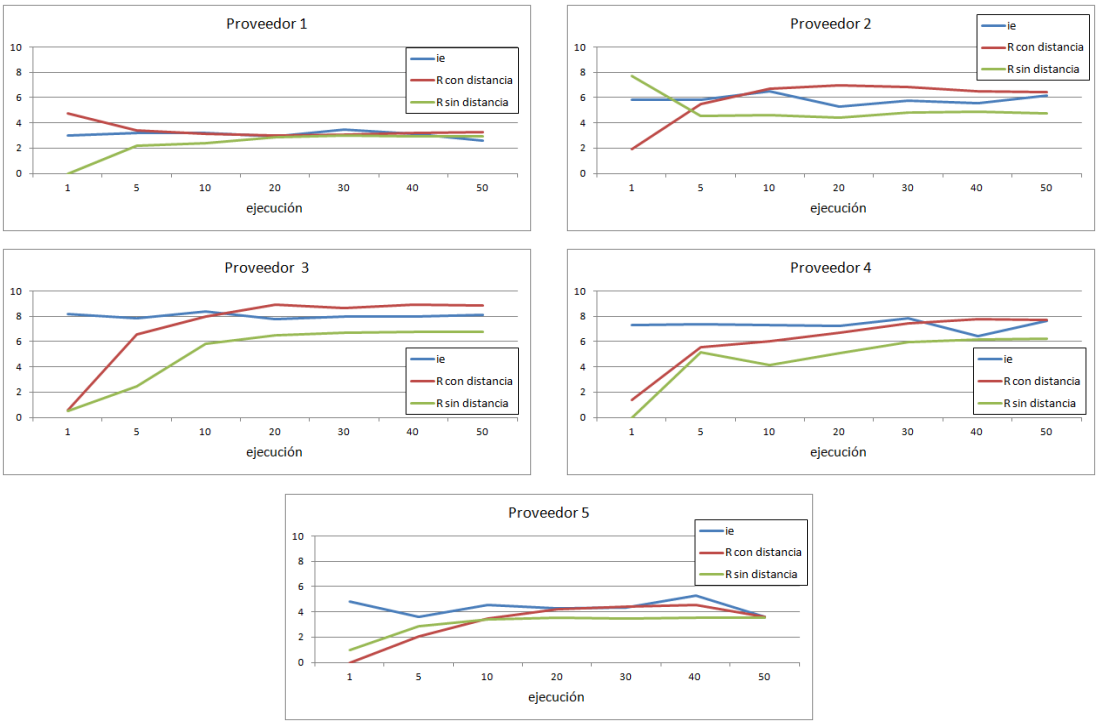

Figura 7.17: Calidad de servicio de cada proveedor durante 50 simulaciones

Al realizar el cálculo del error promedio de los proveedores durante las ejecuciones de la simulación, se observa la tendencia de bajar el error en el tiempo. Esto se debe a que la reputación de los proveedores se refina con la información generada en cada nueva solicitud de recomendación, es decir, a causa de los valores de *AggregateRecommendation* (ar) enviados por los recomendadores y las interacciones pasadas concretadas.

En la Fig. 7.18 se comparan los errores promedios de todos los proveedores para las simulaciones considerando (línea roja) e ignorando (línea verde) la distancia. En lo referente a las primeras 5 simulaciones, puesto que aún no se posee información, o existen muy pocos datos, respecto de interacciones pasadas, las tendencias de las líneas de errores se siguen. Luego se observa una brecha importante entre ambas durante las simulaciones siguientes. A pesar del seguimiento de ambas curvas, es contundente la disminución del error al incluir la distancia. Los errores promedios obtenidos se detallan en la Tabla 7.9.

Como se observa en el gráfico de la Fig. 7.19, para este escenario las distancias resultantes de aplicar la correspondiente distribución gamma son relativamente cortas. Para esta distribución de distancias gamma los pesos a y b se configuran en $a = -8/7(-1{,}142)$ y $b = 6/7(0{,}857)$, con una distancia mínima en 20 y máxima en 700, centradas en 300. Para todas aquellas distancias

Tabla 7.8: ie, R considerando e ignorando distancia para los 5 proveedores en 50 iteraciones.

	ie						R con distancia						R sin distancia					
Ejecución	1	5	10	20	30	40	50	1	5	10	20	30	40	50	1	5	10	20	30	40	50
PA 1	3.01	3.18	3.22	2.92	3.46	3.16	2.56	4.72	3.38	3.13	2.97	3.07	3.18	3.24	0.00	2.22	2.36	2.89	2.97	2.91	2.90
PA 2	5.82	5.81	6.47	5.29	5.74	5.54	6.18	1.94	5.49	6.71	6.99	6.81	6.51	6.47	7.72	4.52	4.58	4.41	4.85	4.89	4.74
PA 3	8.19	7.86	8.38	7.77	7.96	7.99	8.14	0.58	6.60	8.02	8.90	8.68	8.92	8.87	0.52	2.45	5.82	6.51	6.71	6.80	6.76
PA 4	7.28	7.37	7.32	7.26	7.88	6.42	7.62	1.40	5.57	6.05	6.72	7.44	7.76	7.74	0.00	5.14	4.15	5.06	5.95	6.14	6.25
PA 5	4.84	3.62	4.53	4.28	4.36	5.29	3.57	0.00	2.05	3.50	4.25	4.42	4.58	3.57	0.95	2.83	3.43	3.55	3.50	3.52	3.56

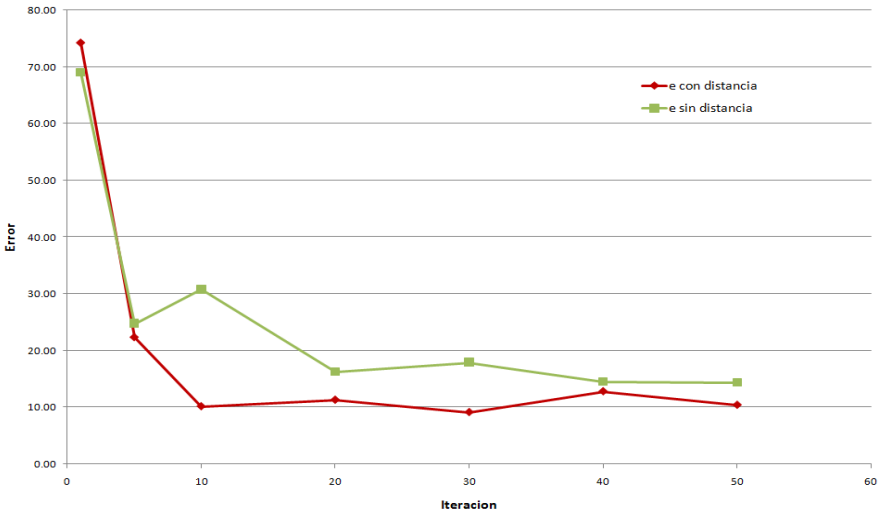

Figura 7.18: Tendencia del error promedio para 50 iteraciones.

Tabla 7.9: Errores promedios para 50 iteraciones.

Iteracion	e con distancia	e sin distancia
1	74.28	68.97
5	22.34	24.79
10	10.17	30.77
20	11.31	16.22
30	9.13	17.95
40	12.81	14.52
50	10.42	14.38

inferiores a 70, normalizamos el resultado en 70. La Fig. 7.20 muestra la distribución de la frescura a lo largo de las 50 simulaciones generadas por todos los proveedores. El Gráfico 7.21 nos muestra una distribución de interacciones realizadas recientemente.

Por su parte, la Fig. 7.22 muestra los casos de los proveedores $PA2$ y $PA4$ en particular, con calidades de servicios diferentes, $5{,}75$ y $7{,}17$ respectivamente. Se observa como a medida que pasa el tiempo, la imagen del proveedor se estabiliza. Haciendo hincapié en la iteración 32, evaluando el $PA2$, el pico hacia abajo ($R = 0$) representa aquellas simulaciones en las que no se calculó un valor de R para el proveedor, puesto que ningún agente usuario se interesó en sus servicios.

En cuanto al comportamiento de los recomendadores en el sistema, se toman dos casos especiales (Fig. 7.23). El $RA2$ tiene una reputación totalmente variable, mientras que el $RA15$ estabiliza su reputación en el tiempo. De todas maneras, el sistema de reputación propuesto converge hacia el comportamiento real del proveedor como resultado de una buena ponderación

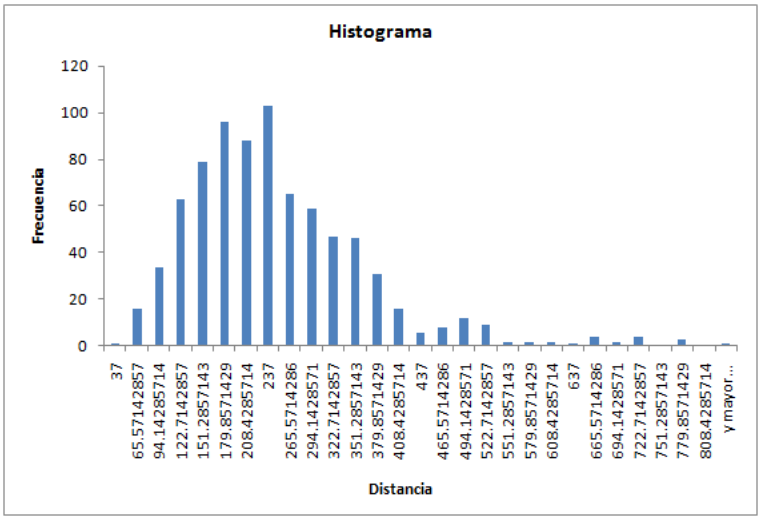

Figura 7.19: Histograma de la distribución gamma para la distancia

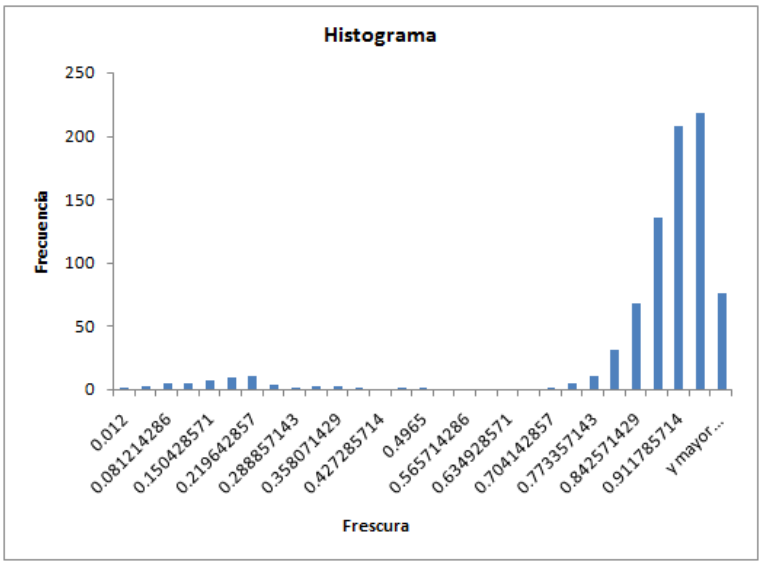

Figura 7.20: Histograma de la distribución gamma para la frescura

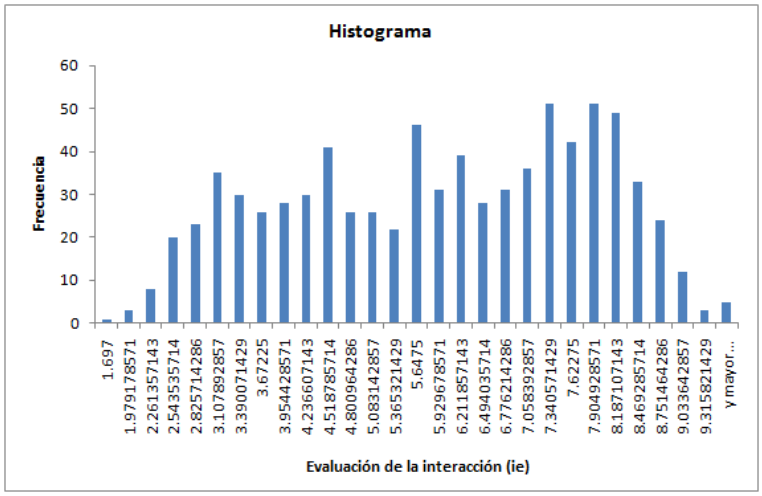

Figura 7.21: Histograma de la distribución normal para la evaluación de la interacción ie

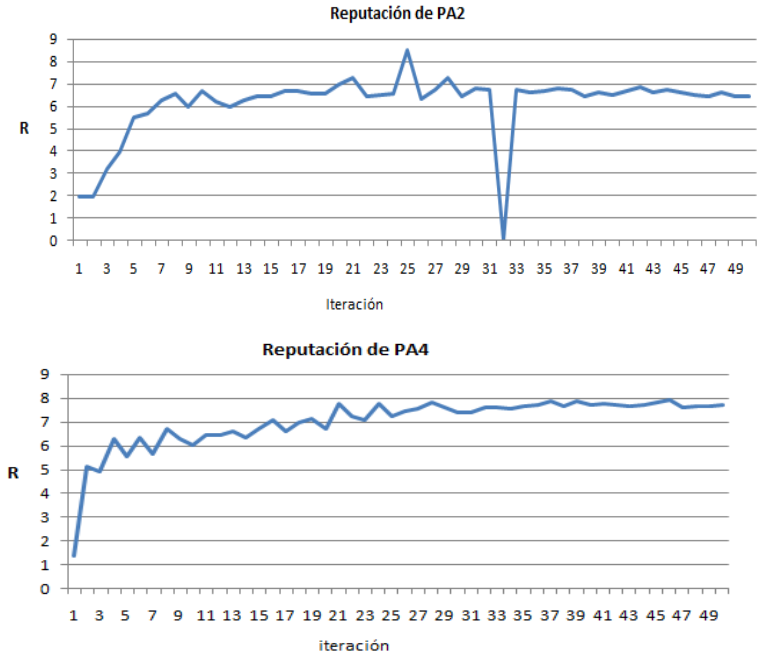

Figura 7.22: Reputación R para los agentes PA 2 y PA 4, considerando la distancia

(Ec. 7.13) de las recomendaciones realizadas.

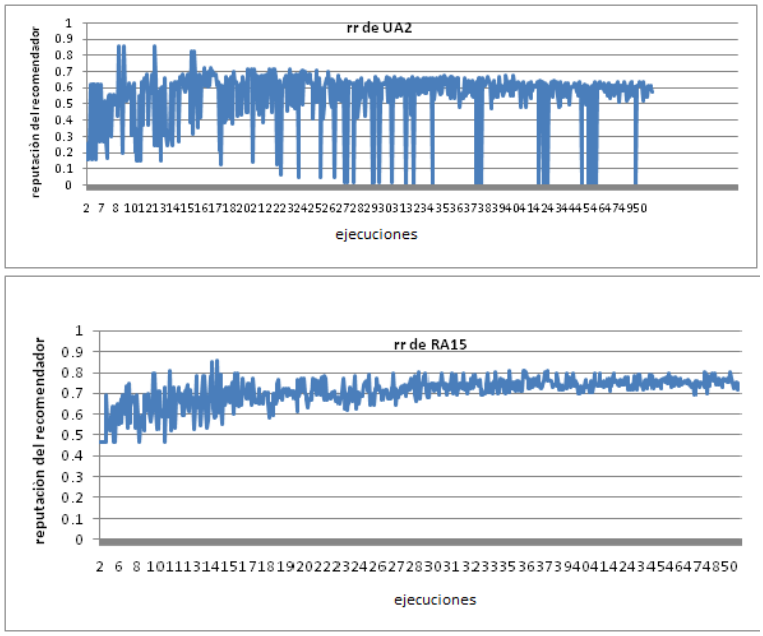

Figura 7.23: Reputación de los recomendadores UA2 y UA15 considerando la distancia

Dado que no contamos con los datos reales para probar la exactitud de los cálculos propuestos, sólo se han presentado los resultados de las simulaciones a partir de datos generados al azar. Además, dado que un modelo de confianza que considere la distancia entre los agentes intervinientes es una contribución innovadora, no hay otros modelos de confianza similares con los cuales comparar la propuesta, sólo se puede mostrar cómo la inclusión de la distancia puede mejorar las predicciones basadas en las recomendaciones en dominios de Inteligencia Ambiental.

7.8 Comparando CALoR con otros Sistemas de Reputación

A los fines de evaluar la performance del modelo CALoR, se propone una comparación con los sistemas de reputación FIRE y el de Bishr. Se realizan 50 simulaciones de 5 interacciones por ronda. La configuración para adaptar la propuesta de Bishr a nuestro experimento es la siguiente: confianza t, un valor entre 1 y 10 extraído de una distribución normal; rating r, un valor entre 0 y 1 con distribución uniforme; c la variable de distancia con distribución gamma. Estos valores se introdujeron en la ecuación Ec. 7.2. Asimismo, se configuró el modelo FIRE para los experimentos, tal como la Ec. 7.17.

$$T_k = \frac{\sum w_k \cdot T_i}{\sum w_k} \tag{7.17}$$

Donde el peso w_k es la función de frescura y T_i es el valor de confianza calculado en base al valor de reputación social. El experimento se configuró de la siguiente manera: T_i un valor

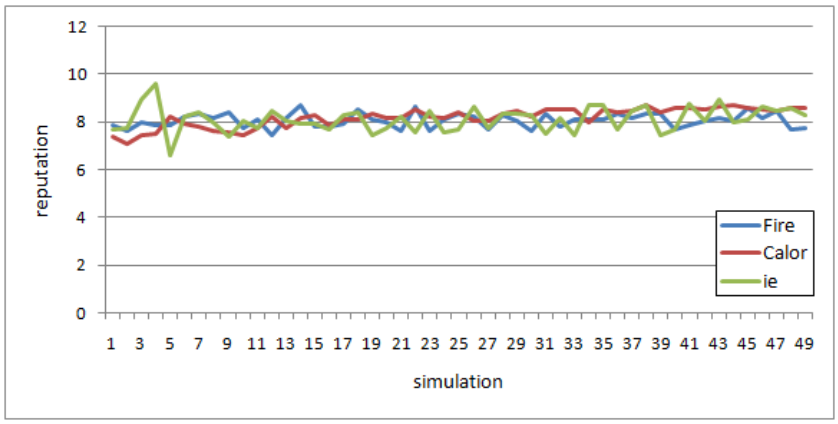

Figura 7.24: Comportamiento de CALoR y FIRE evaluando al agente proveedor PA3.

de un distribución normal dentro del rango [1,10]; w_k es un valor de la distribución gamma en un rango de [0,1].

Para comparar los tres modelos, nos centramos en el agente proveedor 3 y su valor de ie igual a $8{,}13$, es decir, su calidad real de servicio (valor extraído de la Tabla 7.6). Se puede observar en la Tabla 7.10 los valores promedios de reputación y los porcentajes de error calculados con ese valor de ie para los tres modelos de reputación. La comparación entre CALoR y FIRE respecto del proveedor 3 se muestra en la Fig. 7.24, y la comparación entre CALoR y la propuesta de Bishr en la Fig. 7.25. De las gráficas se estima que el sistema CALoR parece converger más rápidamente que los otros dos sistemas, presenta mayor estabilidad.

El modelo propuesto por Bishr presenta un error del 14.8 % mientras que el error del modelo FIRE es del 0.64 % y el error calculado con CALoR es de -0.5 %.

Tabla 7.10: Promedios y errores de los valores de reputación del agente proveedor 3.

	FIRE	CALoR	Bishr et. all
promedio	8.07755102	8.17071429	6.926632653
error	0.6451289	-0.50079072	14.80156638

7.9 Conclusiones del Modelo de Reputación

En este capítulo se ha diseñado un sistema de reputación para dominios de inteligencia ambiental denominado CALoR, sus siglas en inglés: *Context-Aware and Location Reputation System* y en español, "sistema de reputación basado en contexto y localización". Las contribución principal de este capítulo consiste en la inclusión de información geo-espacial relativa a las interacciones en conjunto con cuan reciente fueron esas interacciones (es decir, la frescura). A efectos de cumplir con este objetivo, se han definido nuevos conceptos en la ontología propuesta por la

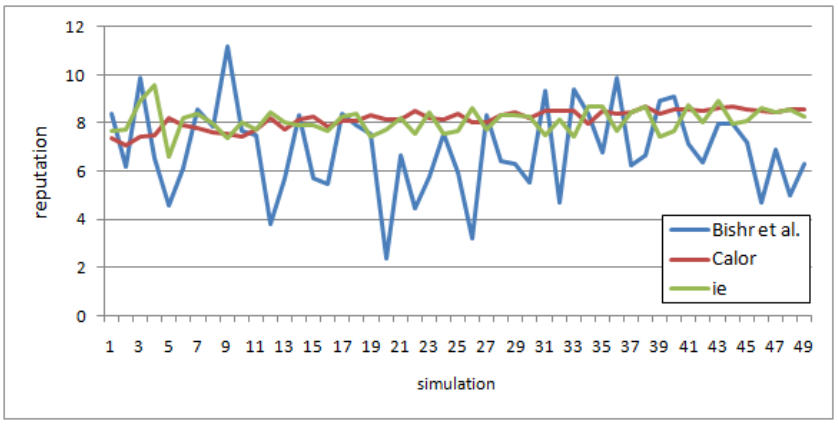

Figura 7.25: Comportamiento de CALoR y Bishr evaluando al agente proveedor PA3.

tesista en el trabajo (Venturini et al., 2010), para dar soporte a la comunicación de la reputación y a la los cálculos del modelo.

Esta extensión de la ontología en Protégée incluye el perfil público del agente usuario definido de forma no intrusiva a través de la observación de los movimientos del individuo. Este perfil contiene dos conceptos estructurados relevantes que se deben generar e intercambiar en el protocolo de recomendación: PastInteractions y ReceivedRecommendation.

Se ha sugerido una formulación específica para la inclusión de referencias espacio-temporales considerando otros trabajos del estado de la cuestión, capítulo 2. El sistema de reputación ha sido validado utilizando un caso de uso ilustrativo. Además, el sistema fue implementado utilizando la plataforma JADE. Y, finalmente, se han ejecutado varias simulaciones para mostrar las ventajas del sistema propuesto considerando el factor distancia en problemas de Inteligencia Ambiental.

III

Conclusiones, Trabajos Futuros y Anexos

8

Conclusiones

En un mundo en el cual el uso de las tecnologías móviles crece a pasos agigantados, donde el usuario es un consumidor de servicios innato sin importar el dominio donde se encuentre, y la posibilidad de conexión continua a internet, hicieron propicio el escenario para la implementación de nuevas tecnologías en el campo de la computación, dando lugar a la Inteligencia Ambiental. El uso de dispositivos móviles, que acompañan al usuario durante sus actividades cotidianas, hizo posible el uso de agentes personales, que lo asisten diariamente. Asimismo, el concepto de la exigencia del usuario en la personalización de servicios requirió de una infraestructura multi-agente capaz de distribuir el procesamiento de datos de diferentes fuentes: localización, preferencias del usuario, servicios ofrecidos. Un aspecto no menor, dentro de las ventajas que proporciona el uso de sistemas multi-agentes, es la capacidad de los agentes de poder aprender y razonar sobre el entorno, considerar las tareas a ejecutar en el mismo y tomar decisiones. Hecho por el cual los agentes son ideales para estos nuevos entornos inteligentes, que requieren de la constante adaptación de su funcionalidad.

8.1 Contribuciones

En esta tesis se ha realizado un diseñado metodológico siguiendo AUML y GAIA, cuyo resultado fue un Sistema Multi-Agente basado en contexto, localización y reputación para dominios heterogéneos de Inteligencia Ambiental. Se han desarrollado los agentes usando JADE_LEAP siguiendo las especificaciones FIPA. Se modelaron cinco tipos de agentes: User, Provider, Broker, Locator y User Manager, con sus respectivos roles y servicios, es decir, las funciones que pueden ejecutar. La arquitectura propuesta se evaluó frente a la arquitectura para sistemas context-aware de (Fuentes et al., 2007) utilizando un método consistente en el conteo de mensajes, pesados de acuerdo a la importancia de su contenido, dependiendo del protocolo implementado. Se concluyó que la descomposición del Agente Central propuesto por (Fuentes et al., 2007) en tres agentes de la arquitectura presentada (Locator, Broker y User Manager Agent) no incrementa el costo comunicativo de los agentes en cuanto a la cantidad de mensajes necesarios para comunicarse. Una evaluación subjetiva se ha considerado para la evaluación final de la arquitectura, comparando la misma con otras arquitecturas genéricas como AMiRA, CONSORT y OOA, demostrando la robustez y flexibilidad que presenta nuestra arquitectura.

Además, para hacer posible dicha comunicación, se ha propuesto una ontología genérica de alto nivel para manejar el conocimiento de cualquier dominio, que sirve como vocabulario para los agentes que conforman el sistema. Esta ontología permitió la manipulación de información contextual, de localización y de reputación. Se construyó la estructura de datos utilizando la herramienta Protegeé que permite la generación de código JAVA, para su incorporación al sistema multi-agente. Asimismo, se planteó el uso de una base de datos espacio-temporal para que los agentes Broker y Locator puedan almacenar información de los movimientos (interacciones) de los agentes usuarios respecto de los proveedores.

Mediante el uso de técnicas de inteligencia artificial, específicamente heurísticas de algoritmos genéticos, se desarrolló un algoritmo para la obtención de las preferencias de usuario, basado en características. Esto último permitió que el agente proveedor, a través de la advertencia del agente intermediario, pueda ofrecer servicios personalizados a los usuarios. Este algoritmo devuelve una lista de tareas priorizadas según las preferencias del usuario. Se evaluó el algoritmo mostrando como a medida que se incrementa la cantidad de tareas en el sistema, es más útil y necesaria su aplicación. Asimismo, se realizó una evaluación basada en características, mostrando el comportamiento del algoritmo para 5 tareas, y como converge rápidamente al incrementar la cantidad de características de 5 a 70. Representando este último, un límite superior, puesto que para esa configuración, el algoritmo presenta un fitness que tiende a infinito puesto que le es imposible resolver la función objetivo con muchas características considerando tan pocas tareas.

Por último, se creó un modelo matemático de reputación basado en localización (CALoR) en el cual el agente recomendador se fija en las interacciones pasadas almacenadas en su perfil público, y en base al momento, lugar y evaluación de la interacción (ie) realizada, calcula el valor de recomendación del proveedor (ar, AgreggatedRecommendation), y se lo envía al agente usuario. Por su parte, el agente usuario, por cada recomendación recibida, calcula la reputación del recomendado) (rr), es decir, calcula cuan buenas fueron las recomendaciones que le hizo el agente recomendador en el pasado. Además, el agente usuario almacena cada recomendación recibida en su tabla *ReceivedRecommendation* del perfil público. Por último, el agente usuario calcula el valor de reputación final (R) del proveedor en el cual estaba interesado mediante una suma ponderada de las recomendaciones recibidas (ar) pesadas con el valor de reputación de cada recomendador (rr), y normalizada en la sumatoria de rr. El modelo se ha evaluado considerando e ignorando el factor distancia, mostrando como en promedio el error ignorando la distancia es del $0{,}24$ mientras que, considerando la distancia, el error promedio es del $0{,}04$.

8.2 Trabajos Futuros

Como trabajo futuro se propone la evaluación de la arquitectura desde el punto de vista de la performance (rendimiento) del sistema tal como se describe en el trabajo (Bouck´e and Holvoet, 2007), y evaluando la escalabilidad del sistema al aumentar el número de agentes en interacción. Además, se deberá aplicar la arquitectura propuesta en otros dominios, para lo cuál se considerará la ampliación y especificación de la ontología, si fuere necesario.

En lo referente al cambio de dominio de un usuario, en este trabajo el usuario debía autentificarse en cara a que el agente usuario comprendiera que cambió de entorno, y los agentes proveedores del nuevo dominio llegarán a considerarlo cliente potencial. Una línea de

investigación actualmente activa es la de contextos multidominios. En este sentido, no haría falta desloguearse de un sistema para ingresar a otro, sino que se utilizaría el mismo sistema. Estaría muy relacionado al concepto de computación continua expuesto en (Plaza., 2004). El enfoque propuesto consiste en el procesamiento centrado en el usuario. Así como el usuario cambia de contexto, su agente personal recibe los datos correspondientes a lo que percibe a través de los servicios contextuales. Los agentes en las aplicaciones basadas en contexto deben ser capaces de adaptarse a un nuevo contexto y aprender de la satisfacción del usuario, luego de realizar cada tarea. Moverse de un contexto a otro requerirá de un esfuerzo de estandarización de los servicios.

En cuanto al modelo de reputación, cómo trabajos futuros se propone evaluar distintos escenarios de AmI con el modelo de reputación CALoR. Para ello las simulaciones a realizar contemplan la variación de la distancia y la frescura, generando escenarios con, por ejemplo, distancias largas e interacciones recientes o distancias cortas e interacciones antiguas, para estudiar como influyen ambos factores juntos y cada uno por separado. Además, sería factible también realizar un estudio sobre agentes maliciosos con el modelo propuesto, para ver como infieren en la decisión final de los agentes que solicitan recomendaciones, y analizar de qué manera detectar a un agente malicioso para penalizar o no considerar sus recomendaciones.

Finalmente, el modelo de reputación que incluye los factores distancia, frescura e evaluación directa de la interacción, podría incorporar la adquisición de preferencias como factor adicional. Pensando en aquellas situaciones en las que un usuario no posee un alto grado de preferencias sobre un proveedor, y este nos envía una recomendaciones sobre ese proveedor. Es decir, si estamos solicitando recomendaciones para visitar un museo de arte contemporáneo, y quien hace la recomendación es un fanático de los deportes extremos y la vida al aire libre que detesta los museos, pero solo asiste para acompañar a su pareja, quizá su valoración sobre el museo sea muy subjetiva. Por ende, el modelo de reputación debería darle un peso menor a este tipo de recomendaciones en donde quien recomienda posee una preferencia mínima sobre el proveedor, y en caso de que sea de su agrado, el valor de la recomendación se debería ver beneficiado. Por la tonta, se podría pensar en agregar un factor de preferencia que en principio multiplicase el valor de recomendación por cada interacción, es decir, la función *SingleRecommendation* del modelo de reputación, obteniendo una función lineal. Luego de analizar los resultados podría reemplazarse dicho factor por una función exponencial o logarítmica sobre la preferencia, siempre multiplicando al valor de recomendación.

8.3 Publicaciones Obtenidas

1. Venturini, V., Carbó, J., and Molina, J. M.. Learning user profile with genetic algorithm in ami applications. In Proceedings of Third International Workshop, HAIS, pages 124-131, Burgos, España, 2008.

2. Venturini, V., Carbó, J., and Molina, J. M.. Using temporal spatial data base in context-aware location-based system. In Proceedings of the Second International Workshop on User-Centric Technologies and Applications, Madrinet, pages 154-162, Salamanca, España, 2008.

3. Venturini, V.. Inteligencia ambiental y nanotecnología: el paso del bit al átomo. Cuaderno

No4 de la Facultad de Ingeniería e Informática, pages 62-82. Universidad Católica de Salta, Salta, Argentina, 2009. ISSN 1852-7094.

4. Venturini, V., Carbó, J., and Molina, J. M.. An ambient intelligent platform based on multi-agent system. XI Workshop of Physical Agents, 2010.

5. Venturini, V., Carbó, J., and Molina, J. M.. Methodological design and comparative evaluation of a mas providing ami. Expert Systems With Applications Journal, 2012.

6. Venturini, V., Carbó, J., and Molina, J. M.. Multiobjective Optimization Problem in Multi-Agent System for AmI scenarios. Intelligent systems for context-based information fusion, Iberamia. Cartagena de Indias, Colombia, 2012.

7. Venturini, V., Carbó, J., and Molina, J. M.. La confianza y la reputación en los sistemas multiagente. Novatica, 2012.

8. Venturini, V., Carbó, J., and Molina, J. M.. Calor: Context-aware and location reputation model in ami environments. En 2º revisión, 2012.

A

Escenarios de Inteligencia Ambiental

EN este anexo se podrán encontrar distintos escenarios de aplicación de la arquitectura propuesta en el capítulo 4.

A.1 Escenario de Conferencias

A continuación se explica un escenario dentro del dominio de conferencias descrito en (Bravo et al., 2005). En el se describe un entorno de conferencias, en donde hay varias ponencias, sesiones de posters y exposiciones de prototipos. Los oyentes reciben mensajes filtrados de acuerdo a sus intereses, acerca de qué se está mostrando en ese momento cerca de ellos o si es que se está por dar inicio a una exposición de su incumbencia. Un usuario llamado Juan llega al sitio de la conferencia. Recibe en su GPS una alarma sobre un sitio vacío en el parking. Estaciona su vehículo y se dirige rápidamente a la recepción. Como Juan ya se ha registrado algunos días atrás, una vez identificado, recibe directamente servicios personalizados en su dispositivo móvil. A continuación, puede ver en este la lista de las sesiones siguientes, sin necesidad de echar un vistazo a la pantalla de RFID del punto de información. Juan camina hacia la sala de demos. Puesto que la demo ya ha empezado, puede ver en su móvil información contextual sobre el expositor (a qué universidad pertenece, cuál es su grupo de investigación y el show completo de la demo). Como no le suena interesante la demo, deja este espacio. Se ubica en la cafetería, lo que hace que sea localizable para cualquier persona que desee hablar con él. Empieza a recibir algunos mensajes en el móvil de gente que lo desea contactar por diferentes razones y por ende, se da inicio a una serie de charlas informales sin ninguna introducción comprometedora. En medio de los mensajes, recibe una sugerencia para unirse a un grupo (coalición) que visitará un sitio de particular interés en la ciudad por la tarde y una recomendación personal sobre un expositor entrante. El acepta la invitación del grupo y corre hacia el hall de la conferencia donde en unos minutos se dará una charla interesante. Cuando llega a la sala de sesión, su móvil le advierte acerca de la presencia de un colega de su universidad. El usuario queda sorprendido ante tanta tecnología.

La arquitectura para este contexto de conferencias, mostrado en la Fig. A.1 cuenta con un servidor que maneja toda la información contextual: datos de las ponencias, las conferencias planificadas, noticias, posters y demos. Los espacios físicos que podemos hallar son: la sala de sesiones, el área común, la recepción. Cuando una persona que atiende una conferencia pasa

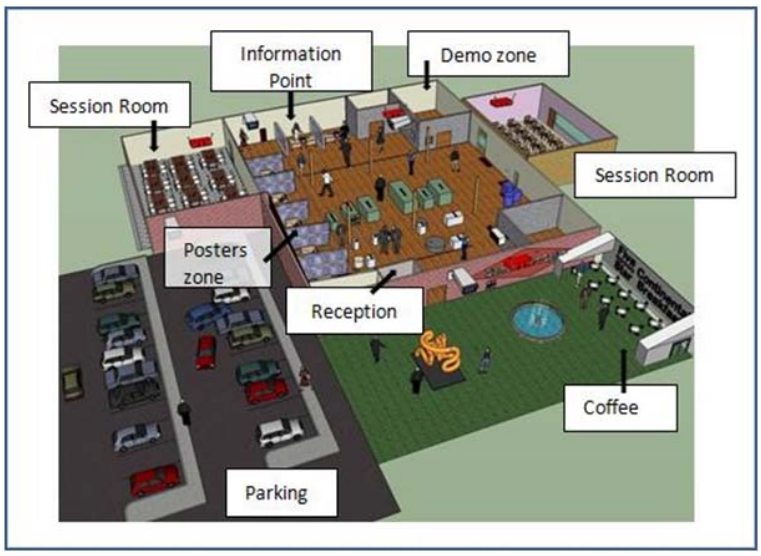

Figura A.1: Contexto de Conferencias

cerca de la puerta de la sala de sesión, ésta lee la información sobre su identificación, perfil y el código de su trabajo, en caso de que la persona deba presentar un artículo. En la sala común se encuentra la sesión de posters y demos. Los posters se muestran cuando el autor está cerca del mismo. Las demos se presentan en PCs situadas en el área, el tiempo en el que se presentan depende de la planificación. La recepción también está equipada con sensores los cuales permiten la identificación de los asistentes evitando colas y aglomeraciones. Esto también permite recibir consultas de los asistentes a las conferencias y enviar las respuestas correspondientes. En caso de que una persona haya colgado un poster, y no se encuentra cerca de él, cuando los asistentes se arriman, el expositor recibe una alarma requiriendo su presencia. También puede suceder que el interesado reciba, en su dispositivo móvil, información adicional a la del poster.

Un sitio de conferencias, un entorno de ferias, un centro comercial, un circuito turístico, todos estos contextos, son especialmente ricos en cuanto a información contextual se refiere, lo cual permite y facilita la provisión de servicios basados en contexto.

A continuación se muestran dos casos de uso para el escenario de conferencias propuesto. En el primero caso se estudia la llegada de una persona al entorno de conferencias y el registro de usuario en el sistema que le permitirá luego recibir información filtrada de sus intereses. El segundo caso ejemplifica la situación en la que una persona está interesada en la exposición de posters, cuáles son las tareas que el usuario del sistema seguirá para estar habilitado en el sistema para recibir información sobre un poster o sobre un expositor en particualr. Los diagramas de ambos casos de uso se muestran en las Figs. A.2 y A.3.

A.1.1 Caso de Uso 1: Actividades del Asistente

Objetivo: Asistir a la conferencia sobre Sistemas Context Aware *Pre-condiciones:* el Usuario debe identificarse en el sistema. En el sistema se han dado de alta todos los ponentes y las

temáticas que han de presentar, como así también los horarios de sus presentaciones. El perfil del usuario incluye los intereses acerca de los seminarios que se van a dictar. *Post-condiciones:* el Asistente ha podido escuchar las conferencias, asistir a la sesión de posters y demos de acuerdo a sus preferencias aprovechando el tiempo correctamente.

Roles intervinientes:

- Localizador: Localizador de Usuario.
- Administrador de Usuarios: Registro de Usuario.
- Intermediario: Correspondencia entre perfil de usuario y servicios que ofrece el proveedor.
- Usuario: Recibir servicios y Realizar consultas.
- Proveedor: Ofrecer servicio y Responder consultas.

Descripción de los roles:

- El Usuario se registra al ingresar en el entorno de la conferencia.
- El Localizador verifica su ubicación.
- El Localizador informa al Usuario que se encuentra cerca de la sala de conferencia.
- El Intermediario advierte al Proveedor que está por exponer, acerca de la presencia del Asistente.
- El Proveedor envía el sumario de su charla al Usuario.

Flujo alternativo 1: El Localizador informa al Usuario que se encuentra en la sala común y muestra en su dispositivo móvil un listado de los póster que pueden interesarle, como así también de las demos.

Flujo alternativo 2: El Usuario se encuentra escuchando la conferencia, efectúa una consulta al Proveedor:

- Puesto que tiene interés en un producto que ha presentado, se inicia un proceso de negociación entre ambos: que costo tiene la licencia, si ellos se encargan del mantenimiento, el costo de las modificaciones, si se adapta para un tipo de empresa particular.
- Dado que tiene interés en un tema particular y no ha comprendido un concepto. En este caso puede el Agente Proveedor recibir la consulta a en su correo electrónico y responder luego.

A.1.2 Caso de Uso 2: Actividades del Ponente de un Poster

Objetivo: Exponer un poster en la conferencia sobre Sistemas Context Aware *Pre-condiciones:* En el sistema se han dado de alta todos los ponentes y las temáticas que han de presentar, como así también los horarios. *Post-condiciones:* El disertante ha expuesto su trabajo llegando a los asistentes que tenían este interés en su perfil.

Roles intervinientes:

- Localizador: Localizador de Usuario.
- Administrador de Usuarios: Registro de Usuario.
- Intermediario: Correspondencia entre perfil de usuario y servicios que ofrece el proveedor.
- Usuario: Recibir servicios y Realizar consultas.
- Proveedor: Ofrecer servicios y Responder consultas.

Descripción de los roles:

- El Usuario se registra al ingresar en el entorno de la conferencia.
- El Localizador verifica su ubicación.
- El Localizador informa al Usuario que se encuentra cerca del poster con la temática x.
- El Intermediario advierte al Proveedor sobre la cercanía del Usuario a su poster. El ponente le envía un mensaje para ver si quiere realizar alguna consulta y en ese caso acercase a su puesto.

A.1.3 Adaptación de la Arquitectura al Entorno de Conferencias

Inicialmente, el agente de localización (LA) establece la posición actual del usuario en el área del estacionamiento. Cuando el usuario se mueve dentro del edificio el LA detecta su posición en el sector de registro. En este momento, el UMA pregunta al agente usuario sobre el registro. El UA envía su posición actual en la sala de conferencias o en la sesión de posters al agente intermediario BA. BA cruza el perfil del usuario de la persona que concurre a la conferencia con la información de cada expositor. Luego el agente usuario recibe mensajes del agente proveedor de la sala de conferencias. El agente localizador detecta los movimientos del usuario hacia la zona de la cafetería. El agente usuario recibe nuevamente su posición. El UMA conoce la situación y le advierte sobre la presencia de otros agentes usuarios. Cualquiera de estos agentes usuarios le podrán recomendar ahora sobre paseos, sobre otros usuarios, etc. Algunos minutos después, algún agente usuario se mueve hacia el área común y recibe un mensaje en su dispositivo móvil del agente proveedor de postes. Luego, aparece en su móvil, información respecto de la línea de investigación y de otros papers. Esto es posible puesto que el agente usuario ha enviado nuevamente su posición al agente intermediario, quien se encarga de hacer la correspondencia. Un agente usuario envía una solicitud al agente proveedor de posters, iniciando de esta manera un proceso de negociación sobre una posible publicación en conjunto.

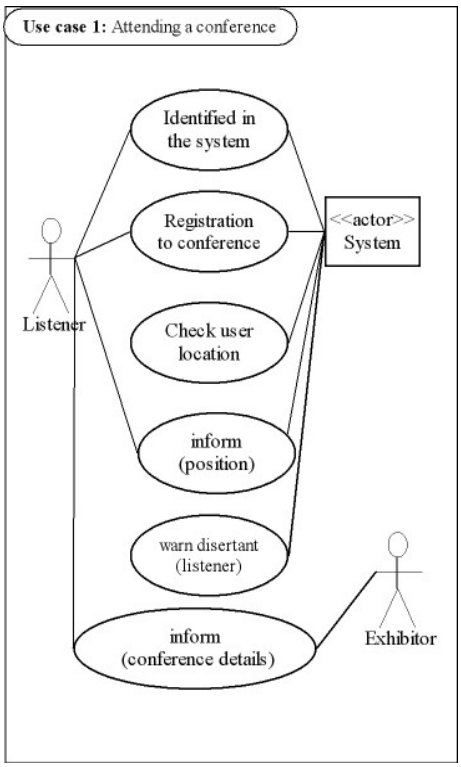

Figura A.2: Caso de Uso 1: Atendiendo una conferencia

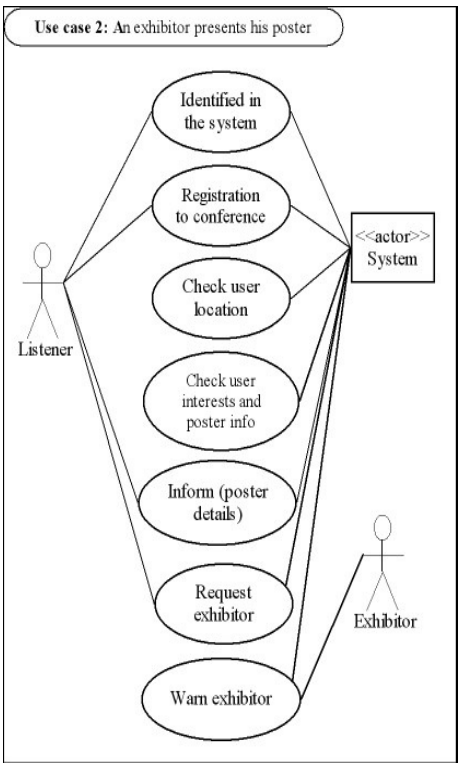

Figura A.3: Caso de Uso 2: Presentación de un poster

A.2 Escenario sobre la Vida en la Ciudad

Para evaluar el modelo, se toma como ejemplo un escenario sobre la vida cotidiana en la ciudad, descrito en (Kjeldskov and Paay., 2005). Asumiendo una ciudad o un pueblo con una estructura similar a la que se muestra en la Fig. A.4. En el esquema, podemos ver los edificios, el centro comercial y el parque de la ciudad. Se supone la presencia de un usuario llamado Tomás que va a encontrarse con María en la biblioteca ubicada en el centro comercial de la ciudad. *Tomás sabe que María está demorada, y por lo tanto decide planear un almuerzo luego de ir a la biblioteca. En este caso, él intenta encontrar una pizzeria cercana a la biblioteca, pero además necesita saber si María está de acuerdo con lo que él planeó. Con este objetivo, él usa su agente personal para preguntar al agente de María sobre sus preferencias en cuanto a comida se refiere, y para localizar la pizzeria*.

Se podría ejecutar una consulta sobre el SGBD para asistir a Tomás, de acuerdo al escenario expuesto anteriormente. En esta consulta vamos a combinar información relacionada a la localización, con información acerca de las preferencias sobre el modelo de contexto. A continuación, se presenta un ejemplo de consulta de contexto:

```
SELECT Places.Name as 'pizzeria', Location.Coordinate as
'pizzeria_geometry'
FROM . . .
WHERE distance (current_geometry, pizzeria_geometry) <200
```

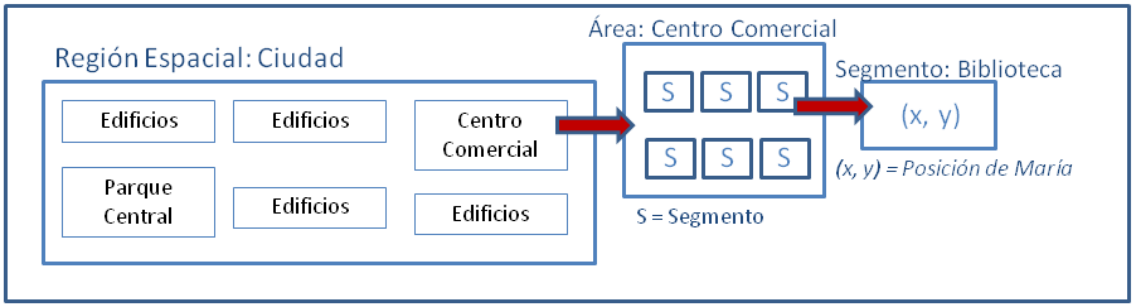

Figura A.4: Escenario de ejemplo: región especial y localización en una ciudad

```
INTERSECTION
SELECT Fact_FastFood.score
FROM . . .
WHERE person.name = 'Mary' and Fact_FastFood.score>0.5
```

Conjugando la Ontología y la Base de Datos Espacio-Temporal

Dado el predicado `(Located(person(Mary),location(2,4)))`, los conceptos utilizados de la ontología son: `Predicate=located, person=Mary and location=(2,4)`. Con este predicado es posible saber si María se encuentra dentro de la biblioteca en este momento. Pero como la información se encuentra almacenada en la base de datos espacio-temporal, es posible, además, realizar preguntas sobre el pasado. Veamos los siguientes casos.

Caso 1: Como se utiliza esta estructura de datos si se desea saber dónde estuvo María en la mañana. Para representar la pregunta se puede utilizar la ontología de esta manera: `(Located (person(Mary), location(2,4), TemporalRegion(this morning)))`. Tenemos el conocimiento almacenado en la base de datos, por lo tanto, podemos ejecutar una consulta al sistema gestor de base de datos la cual retorne el conocimiento: "Ella estuvo en la biblioteca".

```
Set @g1 = 'Point(2,4)'; it represent library location
SELECT activity.* FROM . . .
WHERE person.name = 'Mary'
AND temporal_region.time='during morning'
AND location.coordinate = @g1;
```

Caso 2: Consultar sobre si se encuentra este lugar (la biblioteca) dentro del centro commercial de la ciudad. Para responder a esta pregunta, se puede realizar otra consulta, por ejemplo: "¿Se encuentra la biblioteca dentro del centro comercial?", lo cual se representa en la ontología como: *(IsPartOf(Place1(library), Place2(commercial_center)))*. En ambos caso, estamos utilizando consultas espacio-temporales:

```
Set @g1 = 'Polygon((0 0,10 0,10 10,0 10,0 0), (3 3,5 3,5 5,3 5,3 3))'; representa la
ubicación de la biblioteca
Set @g2 = 'Geometry(0 0,1 1,5 5)'; representa la ubicación del centro comercial
SELECT Places.Name, Location.Coordinate FROM . . .
WHERE Contains(g1,g2);
```

A.3 Escenario en el Aeropuerto

Alberto (A) llega al aeropuerto de Barajas como se muestra en la Fig. A.5. Antes que nada, desea seguir los pasos necesarios para abordar, tomar un café, comprar una guía turística en la sala de abordo y llevar algunos presentes a sus colegas italianos.

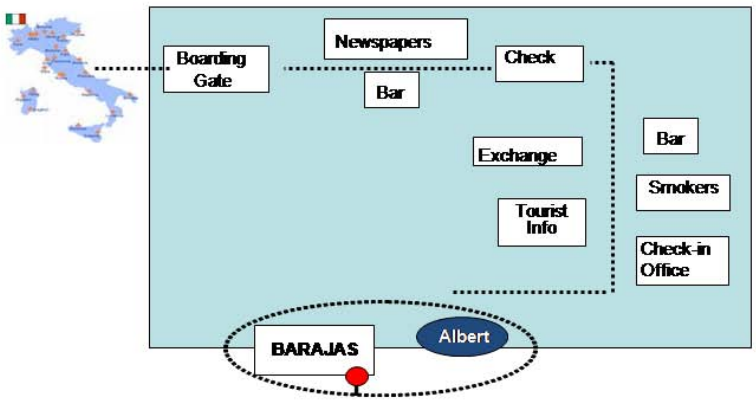

Figura A.5: Escenario de ejemplo: Aeropuerto de Barajas

Se supone el siguiente listado de tareas posibles en el sistema:

- Tarea 1: Check-in
- Tarea 2: Tomar un café en la cafetería
- Tarea 3: Comprar una guía turística de Roma (Comprar un libro)
- Tarea 4: Ir al Punto de Información
- Tarea 5: Comprar presentes

Asimismo, el listado de las características que el usuario evalúa antes de ejecutar una tarea es:

- Calidad (q)

- Costo económico (c)
- Tiempo (t)
- Distancia (d)

El agente usuario de Alberto recibe en su teléfono móvil una lista de tareas enviada por el agente intermediario (Broker Agent). Sobre esta lista, el elige qué tareas desea ejecutar y en qué orden lo hará. El algoritmo genético, que está corriendo sobre el agente intermediario, obtiene un conjunto de individuos, de los cuales uno de ellos es el mejor o el más cercano a la lista de tareas que el usuario ha elegido.

En otro momento, quizá después de ejecutar una tarea, o cuando el usuario se encuentre en otro dominio, el sistema presentará la lista de tareas obtenidas anteriormente, es decir, el mejor individuo del AG. El usuario puede elegir ahora que tareas ejecutar y el orden de las mismas, y el algoritmo genético se ejecutará nuevamente. Se observa como la entrada al AG es la lista de tareas del individuo. El AG evoluciona las preferencias del usuario y retorna un conjunto de listas de tareas.

Si nos centramos específicamente en la tarea "Comprar un libro", las preferencias de Alberto sobre cada característica son las que se muestran en la Fig. A.6 a continuación.

Figura A.6: Caracterización de la tarea "comprar un libro"

Al evaluar las preferencias del usuario sobre cada característica obtenemos:

$$P(T_4) = w_q T_{4,q} + w_c T_{4,c} + w_t T_{4,t} + w_d T_{4,d}$$
$$P(T_4) = 9 * 1 + 0 * 4 + 5 * 9 + 8 * 0 = 54$$

El valor obtenido, es decir 54, es el valor que le da Alberto a la tarea "Comprar un libro".

Si hacemos este cálculo, basado en las preferencias del usuario, por cada acción que planea llevar adelante, se obtiene las preferencias de Alberto por cada tarea:

$$P(T_1) = 125$$
$$P(T_2) = 82$$
$$P(T_3) = 92$$
$$P(T_4) = 54$$
$$P(T_5) = 49$$

Figura A.7: Posición de la tarea en la lista.

A continuación se muestra, en la Fig. A.7, como de acuerdo a las preferencias sobre cada tarea, el AG obtiene un individuo similar a la lista de tareas ordenada, según Alberto:

A través de la función de evaluación propuesta en el modelo, se mide cuan diferente es el individuo (I) a la lista de tareas original (A) de Alberto, considerando la posición (Pos) que ocupa cada tarea en la lista, como se representa en la Ec. A.1.

$$Distancia - Euclidea(A, I) = \sqrt{(0-0)^2 + (2-1)^2 + (1-2)^2 + (3-3)^2 + (4-4)^2} = 2 \tag{A.1}$$

Si el fitness fuese igual a cero, el resultado sería un individuo exactamente igual a la lista de tareas planificada por el usuario.

B

Ontología

En este anexo se podrán encontrar especificaciones de la ontología resultante de los capítulos 5 y 7.

B.1 Conceptos de la Ontología

En la tabla B.1 se detallan los atributos de los conceptos de la ontología para entornos de Inteligencia Ambiental, junto con los conceptos derivados de la extensión necesaria para el modelo de reputación. Cada atributo está acompañado del tipo de dato. Cuando el tipo de dato es 'Class', el lector deberá referirse a la tabla B.2 para su detalle.

La estructura de clases que componen el concepto Country se muestran en la figura B.1:

Cuadro B.1: Conceptos de la Ontología

Concepto	Atributo	Tipo de Dato
Coalition	Problem	Class
Device	MAC	String
	Marc	String
	Model	String
DynamicalPreferences	Week activities	list of String
	Weekend activities	list of String
	Favorites restaurants	list of String
	Entreatments	list of String
	TaskQualityPreference	Integer
	TaskCostPreference	Integer
	TaskTimePreference	Integer
	TaskDistancePreference	Integer
Location	Latitude	Double
	Longitude	Double
NegotiationParameters	CardNumber	Integer
	MyOffer	Float
	PayForm	Int (code of type of payment)
Participant	has_location	Instance of Location
	has_PublicProfile	Instance of PublicProfile
	Id	String
	Name	String
	Roles	String
	PrivateProfile	Class
Place	has_Segment	Instance of Segment
	has_Service	Instance of Service
Product	Name	String
	Price	Float
	Product_ID	String
PublicProfile	has_participant	Instance of Participant
	PastInteraction	Class
	ReceivedRecommendation	Class
RecommenderReputation	Value	Float
Reputation	Value	Float
Service	Service_name	String
	has_Product	Instance of Product
Spatial_Region	Country	Class
StaticPreferences	Music Style	list of String
	Museum	Boolean
	Theater	Boolean
	Technology	Boolean
	Literary Genre	list of String
Temporal_Region	Sector	String
	Date	String
	Hour	String

Cuadro B.2: Atributos de tipo Class

Concepto	Atributo	Tipo de Dato	Clase Superior
Problem			Coalition
	Objective	String	
	Problem	Class	
	Action	Class	
PrivateProfile			Participant
	Address	String	
	City	String	
	Country	String	
	has_location	Instance of Location	
	Nationality	String	
	Smoker	Boolean	
	Job	list of String	
PastInteraction			PublicProfile
	ProviderAgent	Instance of Participant	
	Time	Instance of temporal_region	
	Location	Instance of Location	
	InteractionEvaluation	Float	
ReceivedRecommendation			PublicProfile
	RecommenderAgent	Instance of Participant	
	ProviderAgent	Instance of Participant	
	AggregatedRecommendation	Class	
AggregatedRecomendation			ReceivedRecommendation
	InteractionFreshness	Float	
	LocationDistance	Float	
	InteractionEvaluation	Float	

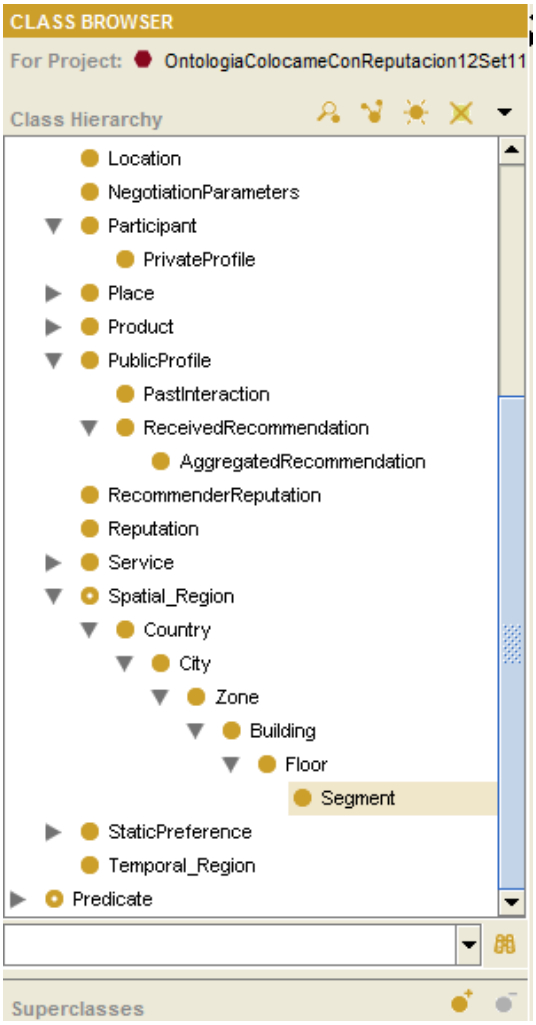

Figura B.1: Clases que componen el concepto Spatial_Region

C

Agentes en JADE

En este anexo se podrán encontrar detalles técnicos de la implementación del sistema multi-agente construido en el capítulo 4.

C.1 Comportamiento de los Agentes en JADE

Cuadro C.1: Comportamientos Agentes en JADE

Agente	Comportamiento	Tipo de Comp.	Función
Localizador	DetectClient	OneShotBehaviour	Detecta la presencia de un agente cliente
Intermediario	ReplyToRegister	CyclicBehaviour	Recibe mensajes del agente cliente para ser registrado.
Intermediario	WarnProvider	OneShotBehaviour	Advierte al agente proveedor sobre la presencia de un agente usuario.
Usuario	ReceiveMsg	CyclicBehaviour	Recibe mensajes de los proveedores u otros agentes clientes.
Usuario	AskForAgreements	OneShotBehaviour	El cliente realiza consultas al proveedor.
Usuario	MakeRecommendation	CyclicBehaviour	
Usuario	ResponseRecommendation	OneShotBehaviour	
Usuario	AskForReputation	Behaviour	
Usuario	ResponseReputation	CyclicBehaviour	
Proveedor	Register	OneShotBehaviour	Registrar un proveedor en el sistema.
Proveedor	RequestNegotiation	CyclicBehaviour	Creado para aceptar o rechazar alguna propuesta del cliente.
Proveedor	OfferRequestServices	Behaviour	Ofrece productos/ servicios a los clientes y se queda a la espera de alguna consulta para finalmente concretar o no la negociación.
Administrador de usuarios	UserFiltering	OneShotBehaviour	Creado para filtrar agentes usuarios con intereses similares

C.2 Diagramas de Clases del Prototipo en JADE

En esta sección se presentan los diagramas de clase UML de los agentes implementados en JADE, utilizando la ingeniería reversa de Netbeans [1].

[1] www.netbeans.org

La figura C.1 es un diagrama de algunas de las clases que componen la arquitectura del sistema multi-agente.

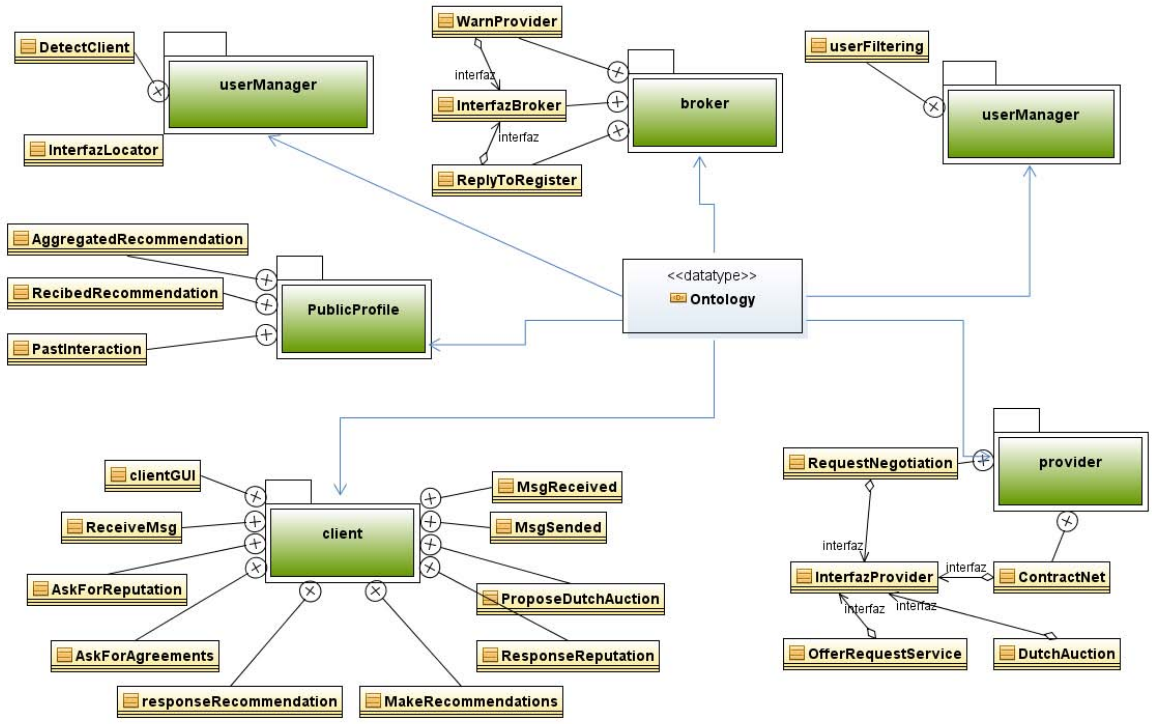

Figura C.1: Diagrama de clases construido a partir de las clases de JADE.Se muestran las clases que instancian a los diferentes tipos de agentes, las clases que representan sus comportamientos, la ontología como metaclase y el perfil público de usuario, también construido en JAVA.

C.2.1 Diagramas de Clases para la representación del Agente Usuario

Los diagramas de clase que se muestran en esta sección representan al mismo agente usuario (Fig. C.2), a la clase construida para su representación gráfica (Fig. C.3) y todos los comportamientos programados para este agente usuario: AskForAgreements (Fig. C.4), MsgReceived (Fig. C.5), AskForReputation (Fig. C.6), ResponseReputation (Fig. C.7), ProposeDutchAuction (Fig. C.8), MakeRecommendations (FigC.9), y ResponseRecommendation (Fig. C.10).

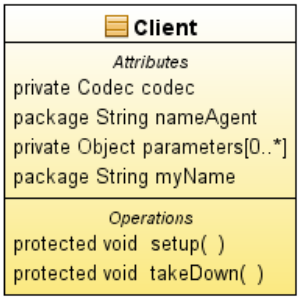

Figura C.2: Clase Cliente representando al Agente Usuario

Figura C.3: Clase para la representación gráfica del Agente Usuario

Figura C.4: Comportamiento *AskForAgreements*

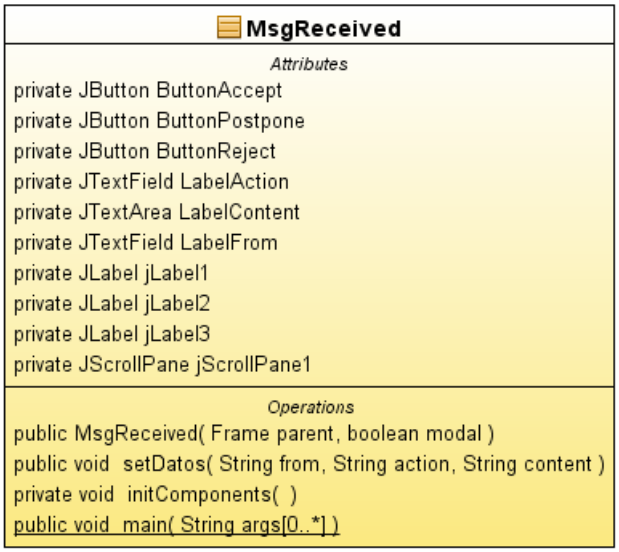

Figura C.5: Comportamiento *MsgReceived*

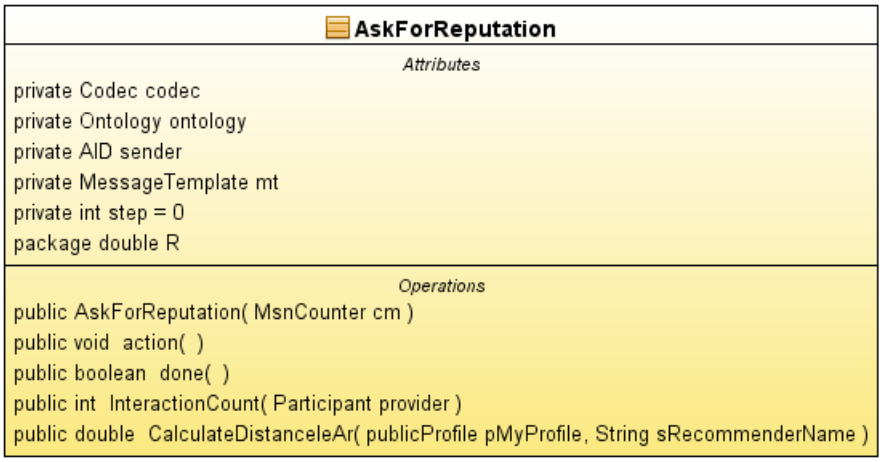

Figura C.6: Comportamiento *AskForReputation*

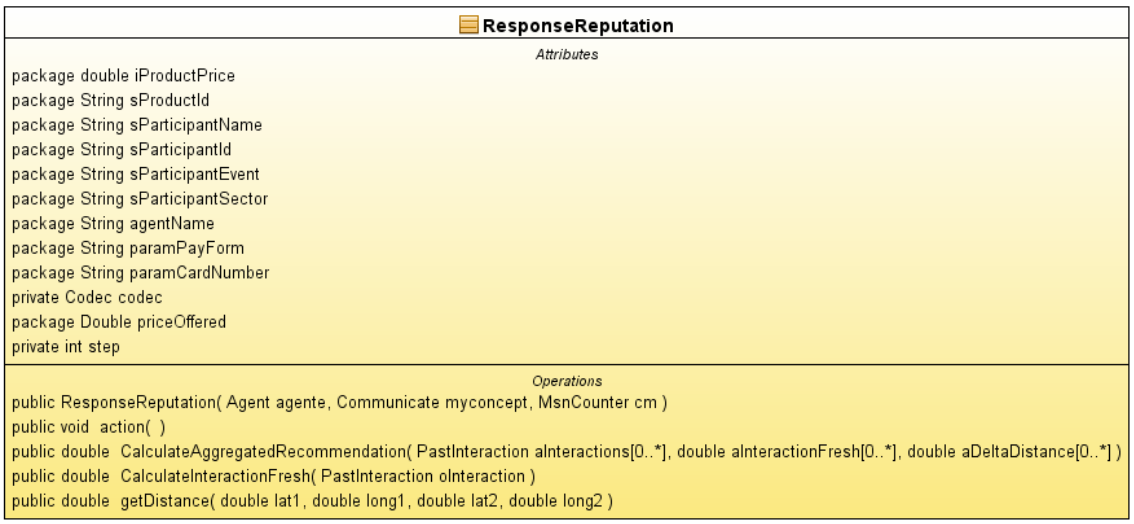

Figura C.7: Comportamiento *ResponseReputation*

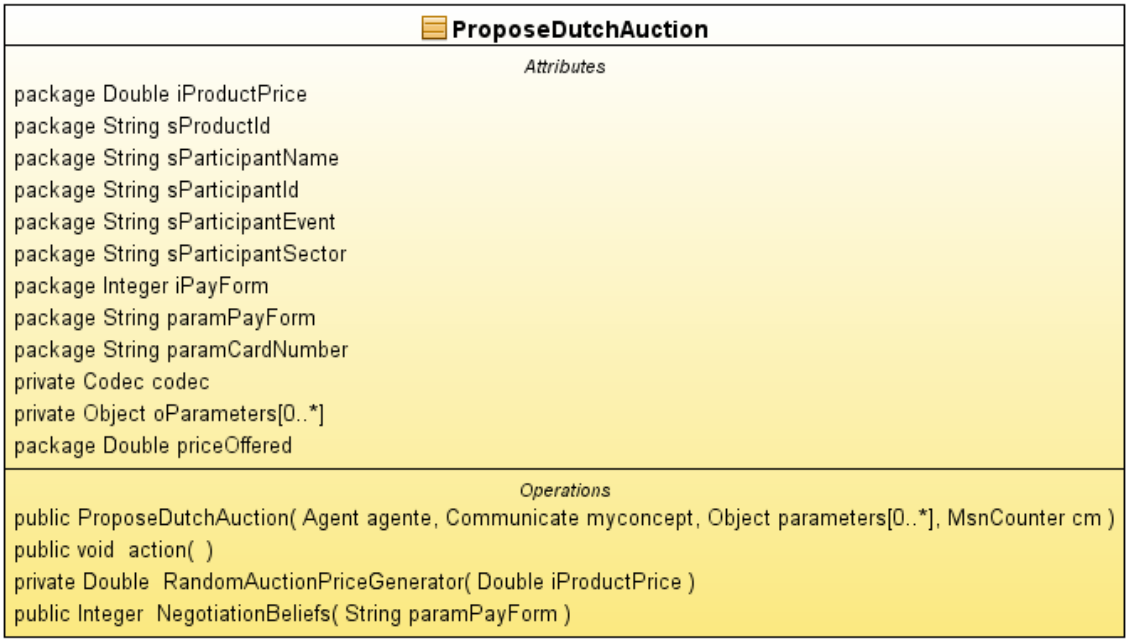

Figura C.8: Comportamiento *ProposeDutchAuction*

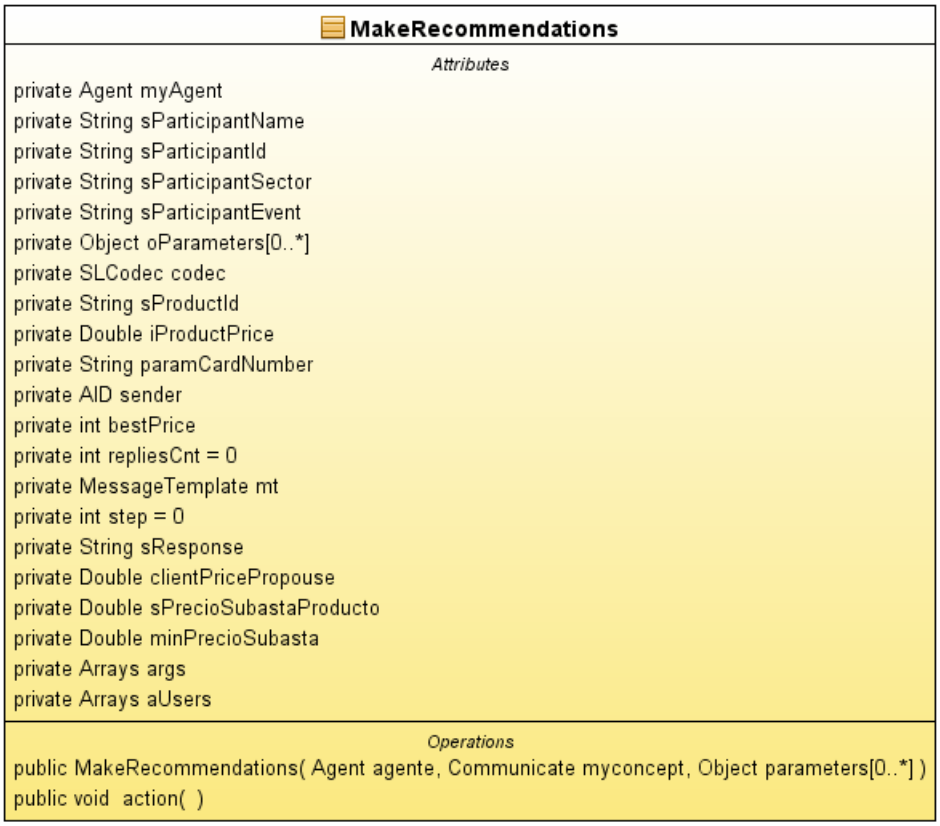

Figura C.9: Comportamiento *MakeRecommendations*

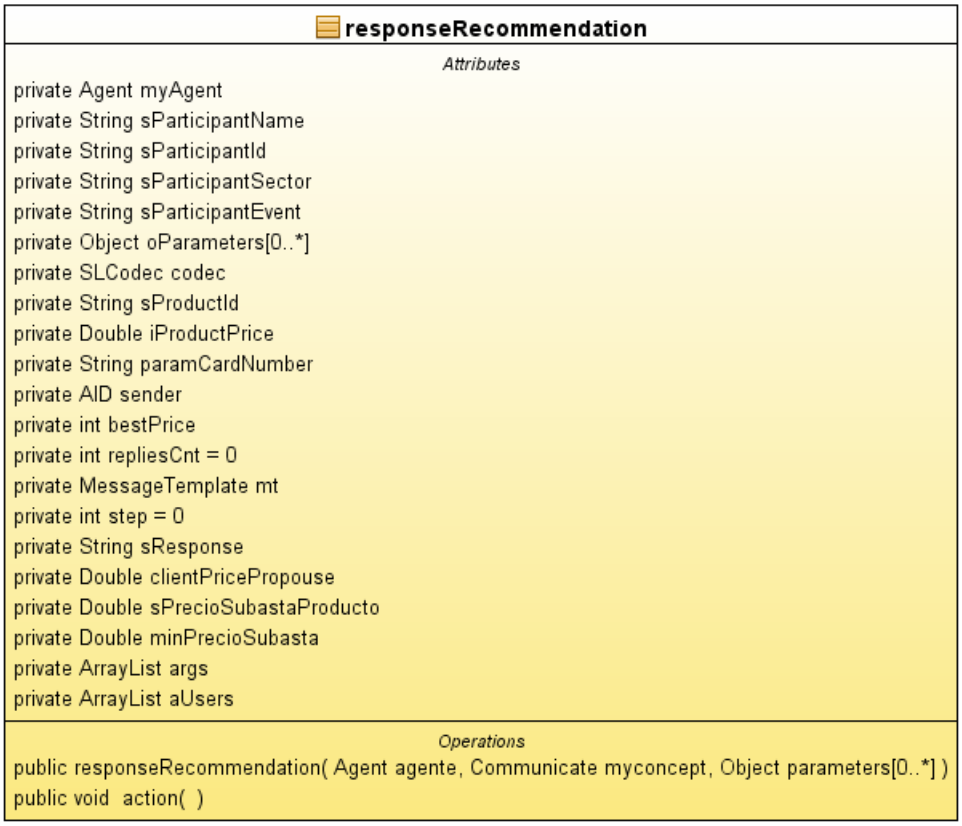

Figura C.10: Comportamiento *responseRecommendation*

C.2.2 Diagramas de Clases para la representación del Agente Proveedor

Los diagramas de clase que se muestran en esta sección pertenecen a la representación del agente proveedor (Fig. C.11), a la clase construida para su representación gráfica (Fig. C.12) y todos los comportamientos programados para este agente proveedor: DutchAuction (Fig. C.13), OfferRequestService (Fig. C.14), y RequestNegotiation (Fig. C.15).

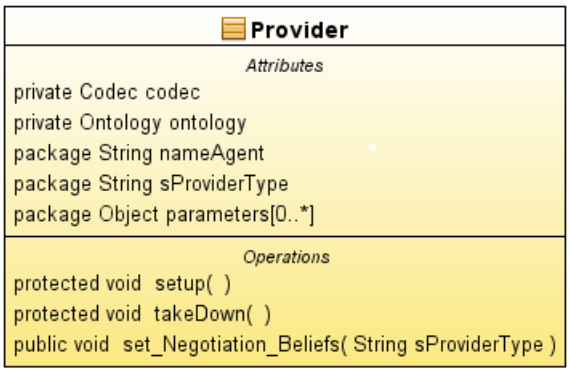

Figura C.11: Clase Provider representando al Agente Proveedor

C.2.3 Diagramas de Clases para la representación del Agente Localizador

En esta sección se muestran las clases que pertenecen al agente localizador (Fig. C.16), a la clase construida para su representación gráfica (Fig. C.17) y el comportamiento DetectClient (Fig. C.18).

C.2.4 Diagramas de Clases para la representación del Agente Intermediario

En esta sección se muestran las clases que pertenecen al agente intermediario (Fig. C.19), a la clase construida para su representación gráfica (Fig. C.20) y los comportamientos ReplyToRegister (Fig. C.21) y WarnProvider (Fig. C.22).

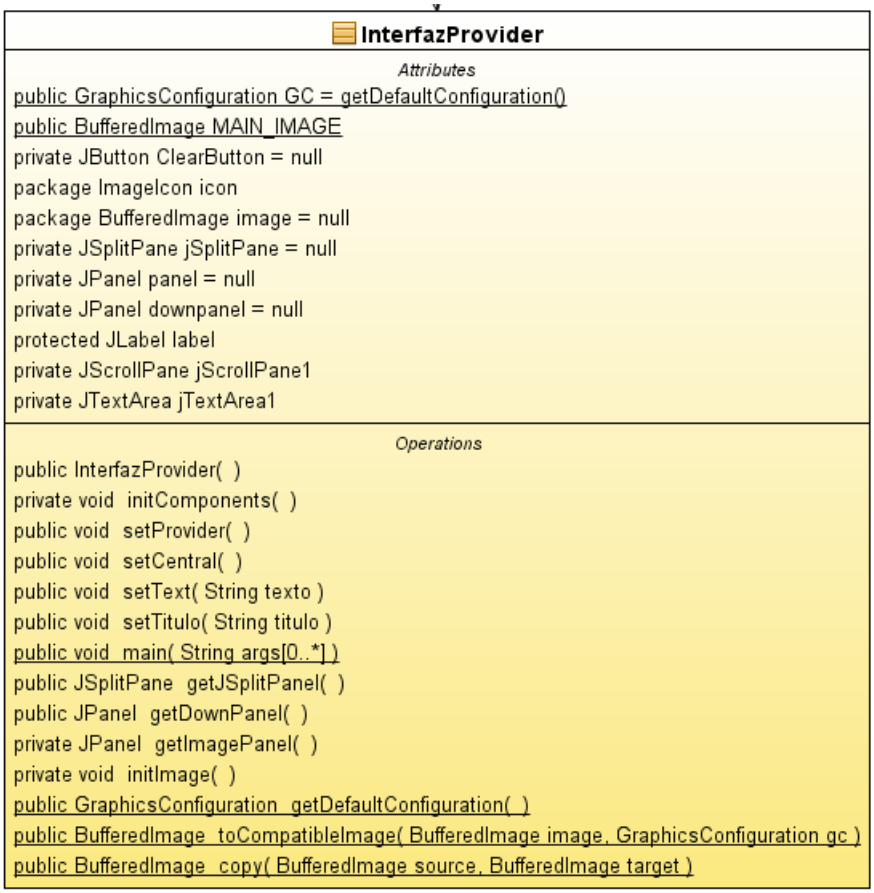

Figura C.12: Clase para la interfaz gráfica del Proveedor

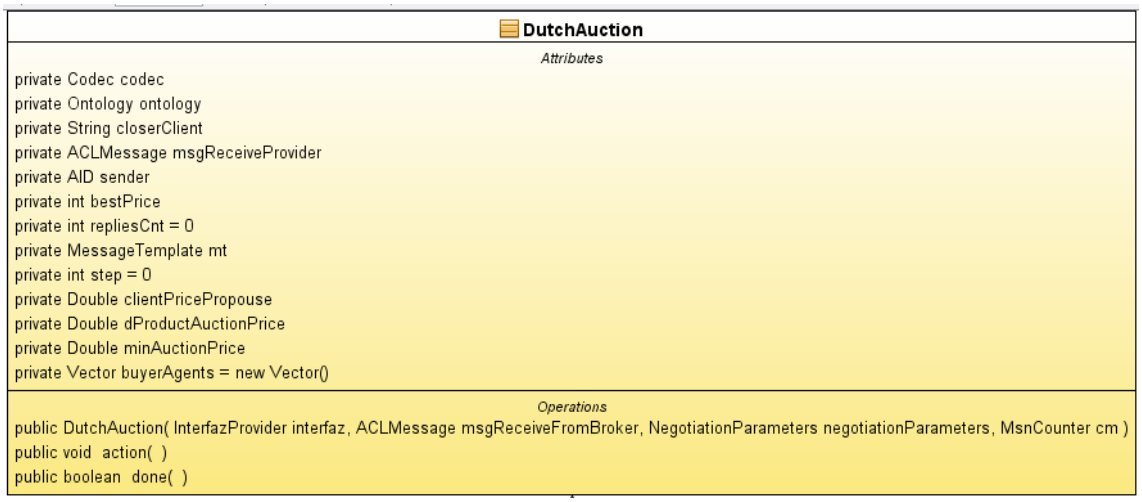

Figura C.13: Comportamiento *DutchAuction*

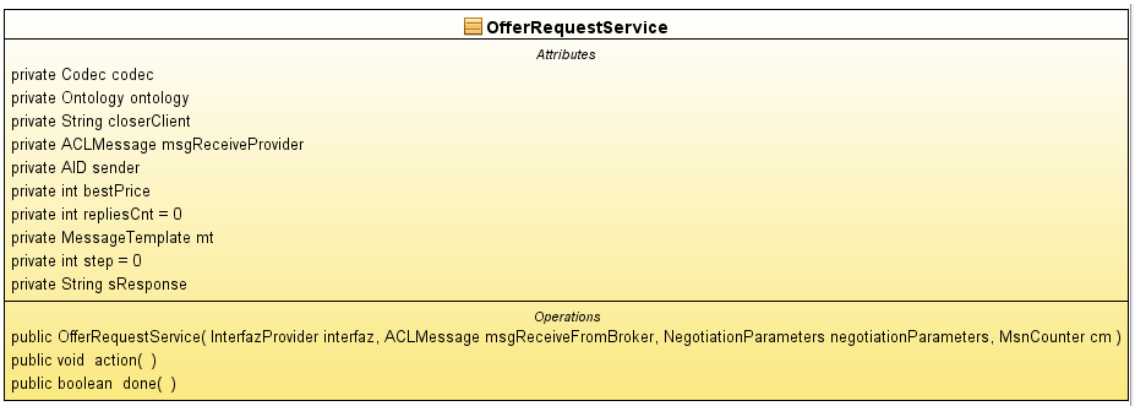

Figura C.14: Comportamiento *OfferRequestService*

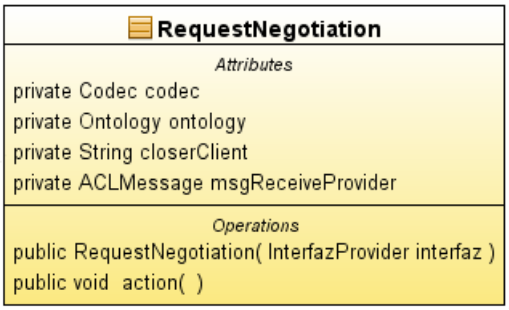

Figura C.15: Comportamiento *RequestNegotiation*

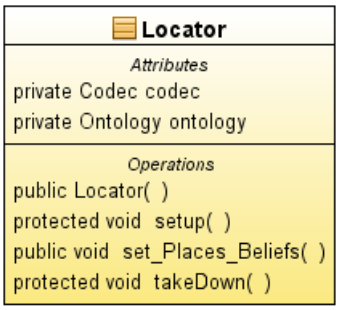

Figura C.16: Clase Locator representando al Agente Localizador

C.2.5 Diagramas de Clases para la representación del Agente Administrador de Usuarios

En esta sección se muestran las clases que pertenecen al Agente Administrador de Usuarios (Fig. C.23) y el comportamiento UserFiltering (Fig. C.24).

Figura C.17: Clase para la interfaz gráfica del Agente Localizador

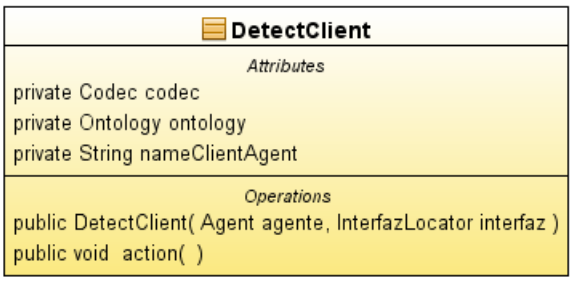

Figura C.18: Comportamiento *DetectClient*

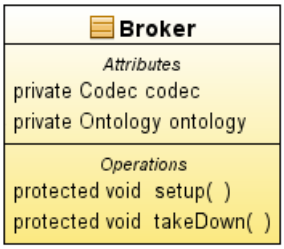

Figura C.19: Clase Broker representando al Agente Intermediario

Figura C.20: Clase para la interfaz gráfica del Agente Intermediario

Figura C.21: Comportamiento *ReplyToRegister*

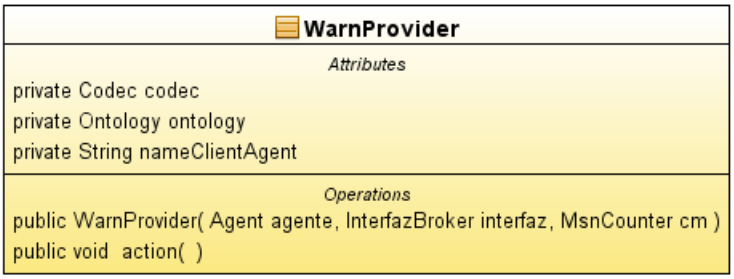

Figura C.22: Comportamiento *WarnProvider*

Figura C.23: Clase UserManagerAgent representando al Agente Administrador de Usuarios

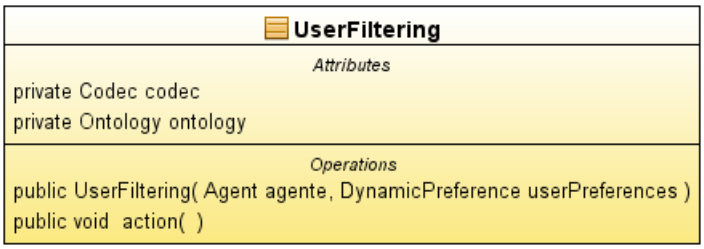

Figura C.24: Comportamiento *UserFiltering*

Bibliografía

Ahlbrecht, P. and Bothe, A. (2005). Leap-db: A mobile-agent-based distributed dbms not only for pdas. *Databases, Information Systems, and Peer-to-Peer Computing*, pages 203–210.

Andriatrimoson, A., Mimouni, N. A., Colle, E., and Galerne, S. (2012). An adaptive multi-agent system for ambient assisted living. In *The Fourth International Conference on Adaptive and Self-Adaptive Systems and Applications*, pages 85–92, Francia.

Bajo, J. and Corchado, J. M. (2005). Sistma multi-agente cbr-bdi para el estudio de la interacción mar-aire. In *CEDI*, pages 11–17.

Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *Int. J. Ad Hoc and Ubiquitous Computing*, 2(4):263–277.

Bellifemine, F., Caire, G., and Greenwood, D. (2007). *Developing multi-agent systems with JADE*. Wiley.

Berger, M., Fuchs, F., and Pirker, M. (2007). Ambient intelligence - from personal assistence to intelligent megacities. In Augusto, J. C. and Shapiro, D., editors, *Advances in Ambient Intelligence.*, volume 164, pages 21–35.

Berlanga, A., Isasi, P., and Segovia, J. (2001). Interactive evolutionary computation with small population to generate gestures in avatars. In Spector, L., Goodman, E., Wu, A., Langdon, W., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M., and Burke, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 823–828, San Francisco.

Bernardos, A. (2008). *Modelo de Integracion de Tecnologias para la provisión de Servicios Móviles basados en Localización y Contexto*. PhD thesis, Universidad Politécnica de Madrid.

Bernon, C., Cossentino, M., and Pavón, J. (2003). Agent-oriented software engineering. *The Knowledge Engineering Review.*, (2):99–11651–63.

Berrueta, D. (2007). Especificación de ontologías avanzadas para la descripción del perfil de usuario. Proyecto de Desarrollo Tecnológico en Cooperación. Fundación CTIC.

Bertino, E., Cuadra, D., and Martinez, P. (2005). An object-relational approach to the representation of multi-granular spatio-temporal data. In *Advanced Information Systems Engineering: 17th International Conference*.

Bertocco, C. (2009). *CONTEXT-DEPENDENT REPUTATION MANAGEMENT IN MULTI-AGENT SYSTEMS*. PhD thesis, UNIVERSITÁ DI PADOVA.

Bertocco, C. and Ferrari, C. (2008). Context-dependent reputation management for soft security in multi agent systems. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 77–81.

Bishr, M. and Mantelas, L. (2008). A trust and reputation model for filtering and classifying knowledge about urban growth. In Science, S., editor, *GeoJournal*, pages 229–237.

Bombara, M., Cali, D., and Santoro., C. (2003). Kore: a multi-agent system to assist museum visitors. In *Workshop from Objects to Agents.*, pages 175–178, Italy.

Botía, J. A. (2005). *Tutorial básico de JADE.* Universidad de Sevilla.

Bouck´e, N. and Holvoet, T. (2007). View composition in multi-agent architectures. *Special issue on Multiagent systems and software architecture. International Journal of Agent-Oriented Software Engineering.*

Boukerche, A. and Ren, Y. L. (2008). A trust-based security system for ubiquitous and pervasive computing environments. *Computer Communications*, 31(18):4343–4351.

Bravo, J., Hervás, R., Nava, S., Chavira, G., and Sanz (2005). Display-based services through identification: An approach in a conference context. In *In the I Ubiquitous Computing and Ambient Intelligence (UCAmI'05)*, Granada - Spain.

Bravo, J., Hervás, R., Sánchez, I., Chavira, G., and Nava, S. (2006). Visualization services in a conferences context: An approach by rfid technology. volume 12 of *3*, Bremen, Germany. Journal of Universal Computer Science.

Carroll, J. M. and Rosson., M. B. (1987). *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, chapter 5. Paradox of the Active User, Cambridge.

Cernuzzi, L., Juan, T., Sterling, L., and Zambonelli, F. (2004). *The Gaia Methodology. Basic Concepts and Extensions*, volume 11, chapter 4, pages 69–88. Multiagent Systems, Artificial Societies, and Simulated Organizations.

Chen, H., Finin, T., and Joshi, A. (2003). An intelligent broker for context-aware systems. In *Adjunct Proceedings of Ubicomp*, pages 183–184.

Chen, H., Finin, T., and Joshi, A. (2004a). An ontology for context-aware pervasive computing environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, 18(3):197–207.

Chen, H., Perich, F., Finin, T., and Joshi, A. (2004b). Soupa: Standard ontology for ubiquitous and pervasive applications. In *In International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston.

Chen, X., Cheng, L., Huo, J., Huo, Y., and Wang, Y. (2004c). Combining prediction through heuristic genetic algorithm on intelligence service system. volume 1, pages 407– 411. Networking, Sensing and Control, 2004 IEEE International Conference on.

Chmiel, K., Gawinecki, M., Kaczmarek, P., Szymczak, M., and Paprzycki, M. (2005). Efficiency of jade agent platform. *Scientific Programming*, (13):1–14.

Cilla, R., Patricio, M. A., García, J., Berlanga, A., and Molina, J. M. (2009). Recognizing human activities from sensors using hidden markov models constructed by feature selection techniques. *Algorithms*, 2:282–300.

Conte, R. and Paolucci, M. (2002). *Reputation in Artificial Societies: Social Beliefs for Social Order*. Kluwer Academic Publishers, Dordrecht.

Corchado, J. M., Bajo, J., de Paz, Y., and Tapia, D. I. (2008). Intelligent environment for monitoring alzheimer patients, agent technology for health care. *Decision Support Systems.*, 44:382–396.

Cozzolongo, G., De Carolis, B., and S.., P. (2004). A personal agent supporting ubiquitous interaction. In *WOA*, Torino, Italy.

Davidsson, P., Johansson, S., and Svahnberg, M. (2006). Characterization and evaluation of multi-agent system architectural styles. In *Software Engineering for Multi-Agent Systems IV, Lecture Notes in Computer Science*.

Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, 5:4–7.

Dogac, A., Laleci, G., and Kabak, Y. (2003). Context framework for ambient intelligence. In *Proc. of eChallenges*.

Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., and Burgelman, J. C. (2001). Scenarios for ambient intelligence in 2010. *ISTAG*, 3(1):100–110.

Esfandiari, B. and Chandrasekharan, S. (2001). On how agents make friends: Mechanisms for trust acquisition. In *Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies*, pages 27–34, Montreal, Canada.

Ferber, J. (1999). *Multiagent Systems - An Introduction to Distributed Artificial Intelligence*. Addison Wesley.

Fernández, R. F., Saenz, J. J. G., and Pavón, J. (2007). Managing contradictions in multi-agent systems. volume E90-D, Lisbon, Portugal. IEICE Trans. Inf. & Syst.

Finin, T., Labrou, Y., and Mayfield, J. (1995). Kqml as an agent communication language. In Bradshaw, J., editor, *Software Agents*, Cambridge. MIT Press.

Finin, T., Weber, J., Wiederhold, G., Gensereth, M., Fritzzon, R., McKay, D., McGuire, J., Pelavin, R., Shapiro, S., and Beck, C. (1993). Draft specification of the kqml agent-communication language.

FIPA (2000). Fipa personal travel assistence specification. Technical report, Foundation for Intelligent Physical Agents. Génova, Suiza.

FIPA (2002). FIPA ACL message structure specification. FIPA agent communication language specifications.

Flury, T., Privat, G., and Ramparany, F. (2004). Owl-based location ontology for context-aware services. In *In Proceedings of Artificial Intelligence in Mobile Systems*, pages 52–58, Nottingham, UK.

Fuentes, L. and Jimenez, D., editors (2005). *An Aspect-Oriented Ambient Intelligence Middleware Platform.*, volume 115 of *ACM International Conference Proceeding Series*, Grenoble, France. Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing.

Fuentes, V., Carbó, J., and Molina, J. M., editors (2006a). *Heterogeneous Domain Ontology for Location Based Information System in a Multi-agent Framework.*, volume 4224/2006. Intelligent Data Engineering and Automated Learning - IDEAL 2006.

Fuentes, V., Sanchez, N., Carbó, J., and Molina, J. M. (2006b). Reputation in user profiling for a context-aware multiagent system. Lisbon, Portugal. Fourth European Workshop on Multi-Agent Systems.

Fuentes, V., Sánchez, N., Carbó, J., and Molina., J. (2007). Generic context-aware bdi multi-agent framework with gaia methodology. In *International Workshop on Agent-Based Ubiquitous Computing*, volume 3, pages 100–110, Hawai, USA. Int Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007).

Gaggioli, A. (2005). User's experience in ambient intelligence systems. *Ambient Intelligence: Conceptual and Practical Issues*, 3(1):100–110.

Gambetta, D. (1990). *Trust: Making and Breaking Cooperative Relations*, chapter Can We Trust Trust?, pages 213–237. Basil Blackwell, Oxford.

Gasser, L. (2000). Mas infrastructure: Definitions, needs and prospects. In *Agents Workshop on Infrastructure for Multi-Agent Systems*, pages 1–11.

Glaser, N. (1997). The comomas approach: From conceptual models to executable code. In Bradshaw, J., editor, *Modelling Autonomous Agents in Multi-Agent Worlds*, Ronneby, Sweden. MIT Press.

Gómez Sanz, J. J. (2003). Metodologías para el desarrollo de sistemas multi-agentes. *AEPIA. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial.*, (18):51–63.

Gruber, T. (1993). A translation approach to portable ontology specification. *Knowledge Acquisition 5*, pages 199–220.

Gu, H., Shi, Y., Xu, G., and Chen., Y. (2006). A core model supporting location-aware computing in smart classroom. In *International Journal of Computer Science and Network Security*, volume 6, pages 161–168, China.

Gu, T., Pung, H. K., and Q., D. (2004a). A middleware for building context-aware mobile services. *IEEE Vehicular Technology Conference*, 5:2656–2660. IEEE Computer Society.

Gu, T., Wang, X. H., Pung, H. K., and Zhang, D. Q. (2004b). An ontology-based context model in intelligent environment. In *Proc. Communication Networks and Distributed Systems Modeling and Simulation Conf.*, London, UK.

Haya, P., Montoso, G., and Alamán., X. (2005). Un mecanismo de resolución de conflictos en entornos de inteligencia ambiental. *Actas del Simposio de Computación Ubicua e Inteligencia Ambiental, UCAmI*, pages 11–18.

Hofer, T., Schwinger, W., Pichler, M., Leohartsberger, G., and Altmann, J. (2003). Context-awareness on mobile devices - the hydrogen approach. In *Proc. Of the 36 th Hawaii International Conference on System Sciences*, page 10.

Hu, S., Ge, J. W., Yuan, Z. W., and Bae, H. Y. (2006). A mobile location-based architecture for intelligent selecting multi-dimension position data over internet. *Application Research of Computers*, 23:226–231.

Huget, M. P. (2003). Agent uml class diagrams revisited. In *Agent Technologies, Infrastructures, Tools, and Applications for E-Services*, pages 49–60.

Huget, M. P. and J., O. (2004). Representing agent interaction protocols with agent uml. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1244–1245, New York.

Huhns, M. N. (2004). Agent uml notation for multiagent system design. In *IEEE Internet Computing*, pages 63–71.

Hunyh, T. D. (2006). *Trust and Reputation in Open Multi-Agent Systems (FIRE)*. PhD thesis, University of Southampton.

Jae-Hyung, B., Chang-Soo, K., and Myung-Suk, K. (2005). A study on the ubiquitous computing service design system. *International Journal of Asia Digital Art and Design*, 3:5–78.

Jih, W., Hsu, J. Y., Wu, C. L., Liao, C. F., and Cheng, S. Y. (2003). A multi-agent service framework for context-aware elder care. In *AAMAS-06 Workshop on Service-Oriented Computing and Agent-Based Engineering*, pages 61–75, Italy.

Josang, A., Ismail, R., and Boy, C. (2006). A survey of trust and reputation systems for online service provision. *Decision Support Systems*, pages 618– 644.

Joumaa, H., Demazeau, Y., and Vincent, J. (2008). Evaluation of multi-agent systems: The case of interaction. In *Proceedings of the 3rd International Conference on Information and Communication Technologies: from Theory to Applications*, Damascus, Syria. IEEE Computer Society publisher.

Khedr, M. and Karmouch, A. (2004). Negotiating context information in context aware system. *IEEE Intelligent Systems*.

Kim, Y., Uhm, Y., Hwang, Z., Lee, M., Kim, G., Song, O., and Park, S. (2006). A context-aware multi-agent service system for assistive home applications. In *Ubiquitous Intelligence and Computing, Third International Conference*, pages 736–745, Wuhan, China.

Kjeldskov, J. and Paay., J. (2005). Just-for-us: A context-aware mobile information system facilitating sociality. Salzburg, Austria. Mobile HCI.

Korpipää, P., Mäntyjärvi, J., Kela, J., Keränen, H., and Malm., E.-J. (2003). Managing context information in mobile devices. *VTT Technical Research Centre of Finland. Published by the IEEE CS and IEEE ComSoc*.

Kwon, O., Shin, J. M., and Kim, S. W. (2006). Context-aware multi-agent approach to pervasive negotiation support systems. *Expert Systems with Applications*, 31(2):275–285.

Lee, W. P. (2007). *Deploying personalized mobile services in an agent-based environment.*, volume 32, chapter 4, pages 1194–1207. Expert Systems with Applications: An International Journal.

Malone, T. W. (1988). What is coordination theory. *National Science Foundation. Coordination Theory Workshop, MIT.*

Marghny, M. H. (2006). Evolutionary algorithm for learning the dynamics of the user profile. *The International Journal of Artificial Intelligence and Machine Learning*, 6:49–54.

Martin, D. L., Cheyer, A. J., and Moran, D. B., editors (1995). *The Open Agent Architecture: A Framework for Building Distributed Software Systems.*, volume 115, USA. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95).

Mas, A. (2004). *Agentes Software y Sistemas MultiAgente: Conceptos, Arquitecturas y Aplicaciones*. Pearson-Prentice-Hall.

Masaud-Wahaishi, A., Ghenniwa, H., and WEIMING, S. (2003). Brokering service in cooperative distributed systems: Privacy-based model. In *International conference on e-commerce and web technologies Nº 4*, volume 2738, pages 435–444, Prague , TCHEQUE REPUBLIQUE.

Masthoff, J., Vasconcelos, W. W., Aitken, C., and Correa da Silva, F. S. (2007). Agent-based group modelling for ambient intelligence. In *Proceedings of AISB*.

Merida Campos, C. and Willmott, S. (2007). Exploring social networks in request for proposal dynamic coalition formation problems. In *CEEMAS '07 Proceedings of the 5th international Central and Eastern European conference on Multi-Agent Systems and Applications V*.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs, Part I.*, volume 1. Springer, London, 3rd edn. edition.

Miraoui, M., Tadj, C., and ben Amar, C. (2008). Architectural survey of context-aware systems in pervasive computing environment. *Ubiquitous Computing and Communication Journal*, 3(3):68–76.

Molina, J. and Corchado, J. M. (2001). Introduction to the theory of agents and multiagent systems. *Agent Theory*.

Moraitis, P. and Spanoudakis, N. I. (2004). Combining gaia and jade for multi-agent systems development. In *Fourth International Symposium "From Agent Theory to Agent Implementation"*., Vienna, Austria.

Morariu, N. and Vlad, S. (2007). Using pattern classification and recognition techniques for diagnostic and prediction. *Advances in Electrical and Computer Engineering*, 7(1).

Moreno, A., Valls, A., and Viejo, A. (2003). Using jade-leap to implement agents in mobile devices. In *TILAB - EXP in search of innovation*, Italy.

Moreno, A. V. and Viejo., A. (2007). Using jade-leap to implement agents in mobile devices. grusma. Bremen, Germany. International Conference on Dynamics in Logistics.

Nakashima (2007). Cyber assist project for ambient intelligence. In Augusto, J. C. and Shapiro, D., editors, *Advances in Ambient Intelligence.*, volume 164, pages 1–20.

Nijholt, A. (2007). Capturing immediate interests in ambient intelligence environments. Lisbon, Portugal. IADIS International Conference Intelligent Systems and Agents.

Noy, F. N. and McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*.

Nwana, H. S., Ndumu, D. T., and Lee, L. C. (1998). Zeus: An advanced tool-kit for engineering distributed multi-agent system. In *In Proc. of PAAM'98*, London, UK.

Odell, J., Parunak, H. V. D., and Bauer, B. (2000). Extending uml for agents. In *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th NationaExtending UML for Agentsl conference on Artificial Intelligence.*, pages 3–7, Austin, TX.

Oravec, V., Budinská, I., and Frankovic, B. (2007). Coalition representation in ontology using various type of logic. In *Slovakian-Hungrian Joint Symposium on Applied Machine Intelligence and Informatics*, Poprad, Slovakia.

Paganelli, F., Bianchi, G., and Giuli, D. (2006). A context model for context-aware system design towards the ambient intelligence vision: Experiences in the etourism domain. In *Procedding of 9th ERCIM Workshop User Interfaces For All*, pages 173–191, Germany.

Patel, J. (2007). *A Trust and Reputation Model for Agent-Based Virtual Organisations (TRAVOS)*. PhD thesis, UNIVERSITY OF SOUTHAMPTON.

Pavón, J. and Gómez-Sanz, J. (2003). Agent oriented software engineering with ingenias. In *In Multi-Agent Systems and Applications II, 3rd International Central and Eastern European Conference on Multi-Agent Systems*, pages 394–403. Springer-Verlag.

Plaza., E. (2004). The computational interplay of physical space and information space. *Computer Science. Inhabited Information Spaces.*, 29:101–111.

Preuveneers, D., Van den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., and De Bosschere, K. (2004). Towards an extensible context ontology for ambient intelligence. In *Ambient Intelligence: Second European Symposium*, volume 3295, pages 148–159.

Ramchurn, S. D., Deitch, B., Thompson, M. K., de Roure, D. C., Jennings, N. R., and Luck, M. (2004). Minimising intrusiveness in pervasive computing environments using multi-agent negotiation. In *First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04)*, Boston, Massachussets, USA.

Ramchurn, S. D., Jennings, N. R., and Sierra, C. (2003). Persuasive negotiation for autonomous agents: a rhetorical approach. In *Proceddings of the IJCAI Workshop on Computational Models of Natural Argument*.

Rao, A. and Georgeff, M. (1991). Modeling rational agents within bdi architecture. In *Proceedings Second International Conference on Principles of Knowledge Representation and Reasoning.*, pages 473–484, Cambridge.

Räck, C., Arbanowski, S., and Steglich, S. (2006). Context-aware, ontology-based recommendations. Arizona, USA. IEEE Communications Society, International Symposium on Applications and the Internet.

Renders, J. M. and Bersini., H. (1994). Hybridizing genetic algorithms with hill-climbing methods for a global optimization: two possible ways. In IEEE World Congress on Computational Intelligence., P. o. t. F. I. C. o., editor, *Evolutionary Computation*, volume 1, pages 312 – 317.

Riva, G., Loreti, P., Lunghi, M., Vatalaro, F., and Davide, F. (2003). Presence 2010: The emergence of ambient intelligence. *Being There: Concepts, effects and measurement of user presence in synthetic environments.*, 3(1):100–110.

Roa, J., Jiménez, A., Seco, F., Prieto, C., Ealo, J., and Ramos., F. (2005). Primeros resultados en la optimización de la ubicación de balizas para localización utilizando algoritmos genéticos. Technical report, Instituto de Automática Industrial - CSIC, Madrid.

Robles, S., Vizcaíno, A., Sabater, J., Carbó, J., del Acebo, E., Hermoso, R., de la Hoz, E., and Alarcos, B. (2007). *Trust and Security*, chapter 4. Issues in Multi-Agent Systems - The AgentCities, Birkhauser, antonio moreno and juan pavón edition.

Sabater, J., Paolucci, M., and Conte, R. (2006). Repage: Reputation and image among limited autonomous partners. *Journal of Artificial Societies and Social Simulation*, 9(2).

Sabater, J. and Sierra., C. (2005). Review on computational trust and reputation models. *Artificial Intelligence Review*, 24:33–60.

Samper Márquez, J. J. (2005). *Estudio y evaluación de un sistema inteligente para la recuperación y el filtrado de información de Internet.* PhD thesis, Universidadde Granada-Departamento de Arquitectura y Tecnología de Computadores, España.

Sashima, A., Izumi, N., , and Kurumatani, K., editors (2004a). *Bridging Coordination Gaps between Devices, Services, and Humans in Ubiquitous Computing.* In Proc. of the Workshop on Agents for Ubiquitous Computing.

Sashima, A., Izumi, N., , and Kurumatani, K. (2004b). Consorts: A multiagent architecture for service coordination in ubiquitous computing. *Journal of Japanese Society for Aritifial Intelligence*, 19(4):322–333.

Schilit, B. N., Adams, N., and Want, R., editors (1994). *Context-Aware Computing Applications.*, volume 115, Santa Cruz, CA. IEEE Computer Society, In Proceedings of the Workshop on Mobile Computing Systems and Applications.

Shoham, Y. (1993). Agent oriented programming. *Artificial Intelligence*, pages 51–92.

Smailagic, A., Siewiorek, D. P., Anhalt, J., and Gemperle, F. (2007). Towards context aware computing experiences and lessons. *IBM T.J. Watson Research Center.*

Small, J., Smailagic, A., and Siewiorek, D. (2001). Determing user location for context aware computing through the use of wireless lan infrastructure. *Submitted for publication to ACM Mobile Networks and Applications*, 6.

Sánchez-Pi, N., Carbó, J., and Molina, J. M. (2008). Jade/leap agents in an aml domain. *Hybrid Artificial Intelligence Systems. Lecture Notes in Computer Science.*, 5271/2008:62–69.

Sánchez Pi, N., Fuentes, V., Carbó, J., and Molina, J. M. (2007). Knowledge-based system to define context in commercial applications. In *Proceedings of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 694–699, Qingdao, China.

Soldatos, J., Pandis, I., Stamatis, K., Polymenakos, L., James, L., and Crowley, b. (2006). Agent based middleware infrastructure for autonomous context-aware ubiquitous computing services. *Science Direct.*

Sorense, C., Wu, M., Sivaharan, T., Blair, G., Okanda, P., Friday, A., and Durand-Limón, H. (2004). A context-aware middleware for applications in mobile ad hoc environments. In *Proc. 2 nd Workshop on Middlerware for Pervasive and Ad hoc Computing*, pages 107–110, New York.

Sotela, F. C. (2010). *Fusión de Datos Distribuida en Redes de Sensores Visuales Utilizando Sistemas Multi-Agente*. PhD thesis, Universidad Carlos III de Madrid, España. Capítulo 5.

Spanoudakis, N. I. and Moraitis, P. (2006). Agent-based architecture in an ambient intelligence context. In *EUMAS*.

Sridharan, N. (1986). Report on the 1986 workshop on distributed artificial intelligence. *The AI Magazine*, pages 75–85.

Stefanidis, K. (2005). On supporting context-aware preferences in relation database system. 6th International Conference on Mobile Data Management.

Susperregi, L., Maurtua, I., Tubío, C., Pérez, M. A., Segovia, I., and Sierra, B. (2004). Una arquitectura multiagente para un laboratorio de inteligencia ambiental en fabricación. In *Desarrollo de Sistemas Multiagente DESMA-2004, en colaboración con IX Jornadas de Ingeniería del Software y Bases de Datos.*, Málaga.

Tewari, G., Youll, J., and Maes, P. (2003). Personalized location-based brokering using an agent-based intermediary architecture. In *Decision Support Systems archive. Special issue: Agents and e-commerce business models*, volume 34, pages 127–137.

Tiba, F. and Capretz, M. (2006). An overview of the analysis and design of sigma: Supervisory intelligent multi-agent system architecture. In *Information and Communication Technologies*, volume 2, pages 3052–3057.

Turhan, A.-Y., Springer, T., and Berger, M. (2006). Pushing doors for modeling contexts with owl dl - a case study. In *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*, pages 5 pp. –17.

Ujjin, S. and Bentley, P. (2002). Learning user preferences using evolution. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning*, Singapore.

Uschold, M. and Gruninger, M. (1996). Ontologies: Principles, methods and applications. *Knowledge Engineering Review 11*.

van Houten, H. (2005). The physical layer of ambient intelligence. *VLSI Technology, 2005. (VLSI-TSA-Tech). 2005 IEEE VLSI-TSA International Symposium on*, pages 9–12.

Venturini, V. (2009). Inteligencia ambiental y nanotecnología: el paso del bit al átomo. *Cuaderno Nº 4 de la Facultad de Ingeniería e Informática. Universidad Católica de Salta.*, pages 62–82.

Venturini, V., Carbó, J., and Molina, J. M. (2008a). Learning user profile with genetic algorithm in ami applications. In *Proceedings of Third International Workshop, HAIS*, pages 124–131, Burgos, Expaña.

Venturini, V., Carbó, J., and Molina, J. M. (2008b). Using temporal spatial data base in context-aware location-based system. In *Proceedings of the Second International Workshop on User-Centric Technologies and Applications, Madrinet*, pages 154–162, Salamanca, Expaña.

Venturini, V., Carbó, J., and Molina, J. M. (2010). An ambient intelligent platform based on multi-agent system. *XI Workshop of Physical Agents.*

Venturini, V., Carbó, J., and Molina, J. M. (2012a). Calor: Context-aware and location reputation model in ami environments. *Journal of Ambient Intelligence and Smart Environments - 2ª Rev.*

Venturini, V., Carbó, J., and Molina, J. M. (2012b). Methodological design and comparative evaluation of a mas providing ami. *Expert Systems With Applications Journal.*, 39(12):10656–10673.

Vázquez, I. and López, d. (2003). Servicios móviles sensibles al contexto. *MovilForum2003*, 3(1):100–110.

Wooldridge, M., Jennings, N. J., and Kinny., D. (2000). The gaia methodology for agent-oriented analysis and design. In *Journal of Autonomous Agents and Multi-Agent Systems*, volume 3, pages 285–312.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152.

Wright, D., Gutwirth, S., Friedewald, M., Vildjiounaite, E., and Punie, Y. (2008). *Safeguards in a World of AmI*. Springer.

Yamabe, T., Takagi, A., and Nakajima, T. (2005). Citron: A context information adquisition framework for personal device. In *Proc. Of the 11th International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 489–495, New York.

Zhang, Y. and Ji, Q. (2006). Active and dynamic information fusion for multisensor systems with dynamic bayesian network. *IEEE Trans. on Systems, Man, and Cybernectics*, 36(2):467–72.