CAEPIA-2003

Agentes Inteligentes en el Tercer Milenio

Organizado por:
Antonio Moreno, Universidad Rovira i Virgili, Tarragona

Con la ayuda de:
AgentCities.ES: Creación de un entorno innovador para la comunicación de agentes inteligentes (TIC2001-5108-E)

San Sebastián, 11 de Noviembre de 2003

Comité de programa

- J. Borrell, Univ. Autónoma de Barcelona
- J.A. Botía, Univ. de Murcia
- J.M. Corchado, Univ. de Salamanca
- U. Cortés, Univ. Politécnica de Cataluña
- P. Cuesta, Univ. de Vigo
- A. García Serrano, Univ. Politécnica de Madrid
- J.C. González, Univ. de Vigo
- C. Iglesias, Univ. Politécnica de Madrid
- V. Julián, Univ. Politécnica de Valencia
- B. López, Univ. de Girona
- J.M. Molina, Univ. Carlos III de Madrid
- S. Ossowski, Univ. Rey Juan Carlos
- C. Paniagua, Univ. de Murcia
- J. Pavón, Univ. Complutense de Madrid
- S. Robles, Univ. Autónoma de Barcelona
- J.A. Rodríguez, Instituto de Investigación en Inteligencia Artificial, CSIC
- A. Valls, Univ. Rovira i Virgili
- S. Willmott, Univ. Politécnica de Cataluña

Agenda

9:00 - 9:15 Presentación del taller
A.Moreno, URV

9:15 - 11:00 ***Conferencia invitada: estado actual de la tecnología de agentes y sistemas multi-agente en Europa***
Michael Luck (University of Southampton, UK)

11:00-11:30 Pausa

11:30-12:30 ***Sesión 1: Presentación de artículos***
Responsable: M.Garijo, UPM

Agentes TRACK-R para asistencia en carretera
A.M.García Serrano, V.D.Méndez Sáenz, J.Z.Hernández, F.Carbone (UPM)

Gestión de sensores mediante sistemas multiagente
J.M.Molina (UC3M)

Desarrollo de servicios turísticos a usuarios
M.Bätzold, M.Navarro, V.Julian, V.Botti (UPV)

HeCaSe: an agent-based system to provide personalised medical services
D.Isern, D.Sánchez, A.Moreno, A.Valls (URV)

Towards holonic multiagent systems: ontology for control tool boxes
S.Suárez, B.López, J.Melendez (UdG)

Ontological issues in agent-aware negotiation services
A.Giovanucci (iSOCO), J.A.Rodríguez-Aguilar (IIIA)

13:30-15:30 Comida

15:30-17:30 ***Sesión 2: Demostraciones de sistemas multi-agente***
Responsable: V.Julian, UPV

Sistema multiagente para gestión de reuniones
P.Cuesta, J.C.González, P.Raña, F.J.Rodríguez (U.Vigo)

HeCaSe: deployment of agent-based health care services
D.Isern, D.Sánchez, A.Moreno, A.Valls (URV)

Tools for the design and deployment of agent mediated electronic institutions
M.Esteva (IIIA)

INGENIAS environment for MAS generation
J.Gomez Sanz, R.Fuentes, J.Pavón (UCM)

Framework for developing mobile agent applications
J.Ametller, J.Borrell, J.García, G.Navarro, S.Robles (UAB)

Sistema multi-agente para la gestión de agendas personales
A.Rayón Alonso, M.A.Sánchez (UPM)

GenialChef
Agents Research Lab (UdG)

17:30-18:00 Pausa

18:00-19:30 ***Sesión 3: Mesa redonda "Uso de metodologías en el diseño de agentes y sistemas multi-agente"***

Responsables: J.Pavón, J.Gómez, UCM

Constructing deliberative agents using K-SIM case-based reasoning systems
J.M.Corchado, E.S.Corchado, M. González-Bedia, A. de Luis, L.F.Castillo (U.Salamanca, U.Burgos)

Towards a recursive agent oriented methodology
A.Giret, V.Botti (UPV)

A framework for evaluation of agent oriented methodologies
P.Cuesta, A.Gómez, J.C.González, F.J.Rodríguez (U.Vigo)

Contribuciones aceptadas

Agentes TRACK-R para asistencia en carretera

A. M. García Serrano, V. D. Méndez Sáenz, J. Z. Hernández y F. Carbone

Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid
Campus de Montegancedo s/n
28660 Boadilla del Monte (Madrid), España
{agarcia, vmendez, phernan, fcarbone}@isys.dia.fi.upm.es

Resumen. Se presenta el diseño de una arquitectura multiagente para la gestión de tráfico rodado[1] y su actividad para ayuda a la identificación de la ruta mas corta en términos de longitud y conflictos en el momento de atravesar determinados puntos de la misma. Además se aportan detalles sobre la implantación de 5 agentes TRACK-R (traffic city agent for knowledge based recomendation) utilizando el entorno JADE que proporciona compatibilidad con estándares. Asimismo se utiliza Ciao Prolog que facilita el desarrollo de los mecanismos de inferencia y la gestión del conocimiento de cada área concreta por parte de los agentes TRACK-R. Finalmente se incluyen algunas conclusiones y las líneas futuras a tener en cuenta.

1 Introducción

La necesidad de afrontar nuevos desarrollos para problemas complejos en entornos distribuidos ha motivado el auge de los sistemas multiagente. Tras una primera etapa de gran interés producto de la presentación de prometedores modelos de inteligencia artificial distribuida [3] y algunos casos de ejemplo, se pasó a una etapa de cierto desconcierto por la proliferación de aproximaciones, diferentes herramientas e incluso estándares divergentes. Ha sido a principios de la década actual cuando se ha comenzado una etapa mas realista y razonable ante la necesidad de utilizar en la práctica la tecnología multiagente, etapa en la que se espera se produzca la consolidación de esta tecnología y así se materialice su potencial para resolver problemas que se plantean cuando en un mismo entorno conviven distintos agentes software que deben coordinar sus actividades.

En este artículo se presentan los resultados actuales para el problema de la gestión del tráfico rodado [6] que se ha abordado con objeto de desarrollar un servicio o conjunto

[1] Trabajo financiado por los proyectos Agentcities.NET, IST-2000-28-384 (Traffic Agentcity for knowledge-based Recommendation (TRACK-R) y DAMMAD TIC-2000-1370-C04 (Diseño y Aplicación de Modelos Multiagente para Ayuda a la Decisión).

de funcionalidades en la que sea necesaria la coordinación de diferentes agentes. Este servicio consiste en encontrar la mejor ruta posible entre dos nodos de la red viaria (ciudades) teniendo en cuenta la distancia hasta el destino y la información del tráfico en el momento del recorrido concreto (predicción). Esta información es requerida tanto por agentes software como por usuarios para los que se activa un agente personal, que les asiste en el entorno software y obtiene la mejor ruta.

La definición de agente ha resultado ser muy controvertida y en la literatura se encuentran muchas, sin que ninguna de ellas haya sido aceptada plenamente por la comunidad científica. Una opción de concepto de agente bastante aceptada ha sido la que se realiza mediante un conjunto de propiedades que caracterizan a los agentes, aunque un agente no tenga por qué poseer todas ellas. Así, seguramente la definición más citada sea la propuesta por Wooldridge y Jennings [5] según la cual *un agente es un sistema informático que está situado en un entorno y es capaz de actuar de forma autónoma y flexible.*

El hecho de que un agente TRACK-R sea una entidad situada en un entorno implica que recibe información de entrada que proviene de una simulación de sensores o de información aportada por los centros de control de tráfico que estarían situados en las diferentes localidades. Estos agentes a partir de esta información realizan acciones que de alguna forma implican cambios en el entorno, como la variación en el flujo de coches provocada por las decisiones "aconsejadas" de los agentes asistentes o personales. La autonomía en nuestro caso se refiere a que los agentes TRACK deben ser capaces de actuar con o sin la llamada de otros agentes (característica prevista para su desarrollo posterior en DAMMAD [7]). Que un agente sea capaz de actuar de forma flexible para conseguir sus objetivos implica que el agente debe ser: reactivo esto es capaz de responder en un tiempo adecuado a cambios en el entorno en el que se encuentra situado, proactivo o capaz de exhibir un comportamiento oportunista y dirigido a obtener sus metas tomando la iniciativa cuando sea necesario y social o capaz de interactuar con otros agentes (humanos o no) a través de un lenguaje de comunicación entre agentes. En este momento los agentes TRACK-R son reactivos y sociales con un modelo simple de coordinación basado en la comunicación de información.

Se ha diseñado un sistema multiagente basado en varios agentes TRACK-R, que tienen información relacionada con el mapa de vías y carreteras principales de una ciudad así como del tráfico en el área correspondiente a los alrededores de una ciudad o puntos interiores de conocido conflicto (cruces en este momento). Otro tipo de agentes los PERSONALAGENT actúan como el interfaz de usuario del sistema, enviando las peticiones y mostrando las rutas adecuadas identificadas por el sistema de agentes.

Los agentes TRACK-R han sido desarrollados utilizando el entorno JADE [8,10] que proporciona compatibilidad con los estándares FIPA. Se ha utilizado Ciao Prolog [9]

para desarrollar los mecanismos de inferencia y la gestión del conocimiento de cada área concreta por parte de los agentes TRACK-R.

2 Arquitectura de los agentes TRACK-R

La selección del tipo de arquitectura de un agente debe realizarse de acuerdo con los requisitos de la aplicación a desarrollar. En el caso de los agentes TRACK-R se han identificado como necesarias tanto las arquitecturas deliberativas como las reactivas. El uso de arquitecturas deliberativas ha permitido que la gestión de tráfico sea un proceso de inferencia sobre de las Bases de Conocimiento y con la información recibida en tiempo real. Además como la adquisición de datos del entorno, en general, exige una transformación cuantitativa o cualitativa de carácter simple, por los TRACK-R dispondrán también de un componente reactivo para tareas en tiempo real y normalmente simples (como por ejemplo la comprobación del mantenimiento del estado de la carretera).

A continuación se presenta el contenido de los diferentes componentes de los agentes TRACK-R.

2.1 Bases de Conocimiento

Por la complejidad inherente de la gestión de tráfico en esta primera aproximación, el conocimiento distribuido de las rutas se reparte de manera que cada agente controle una ciudad y sus alrededores, aunque el sistema está diseñado para que se admitan otras posibles soluciones para repartir este conocimiento.
Los agentes TRACK-R disponen de conocimiento acerca de todos los elementos relevantes del área que controla, esto es, información de ciudades, carreteras, cruces de carreteras y puntos kilométricos. En la figura 1 se muestra el mapa de Bilbao y en la Figura 2 el de Barcelona. Además de estos dos nodos, actualmente tenemos las bases de conocimiento sobre Valencia y Madrid.

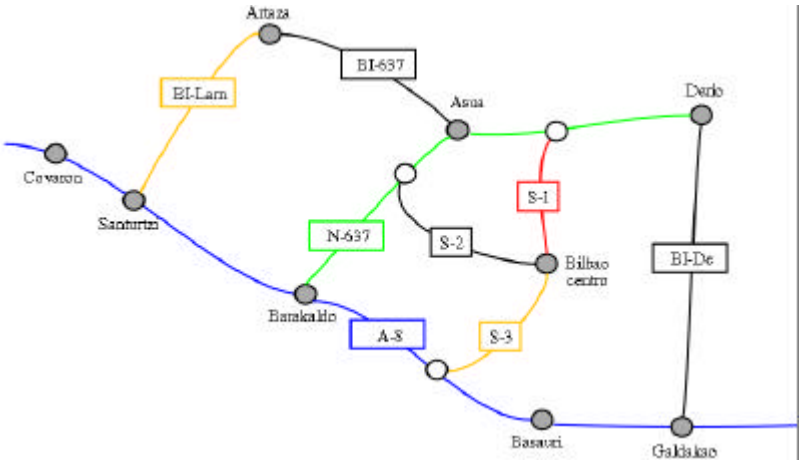


Figura 1: Nodo de Bilbao y sus alrededores

La Base de conocimiento de los agentes TRACK-R se ha codificado con Ciao Prolog [www.clip.dia.fi.upm.es] que facilita el desarrollo de los mecanismos de inferencia genéricos sobre el conocimiento de cada área concreta por parte de los agentes TRACK-R.

Se incluye a continuación la base de conocimiento parcial del agente de Bilbao, con objeto de clarificar el tipo de infomación que utilizan estos agentes:

```
% UrbanArea concept
urbanArea('Covaron').
urbanArea('Santurtzi').
urbanArea('Barakaldo').
...
% Road concept
% road(Name,roadCategory,speed)
road('A8',local,120).
road('BI-Lam',local,80).
road('N637',national,100).
…
road('S2',local,100).
road('S3',local,100).
% Crossroad concept
crossroad('N637','S1',bi1).
crossroad('N637','S2',bi2).
crossroad('A8','S3',bi3).
% RoadConnected relationship
%roadConnected(X,Y,Z):-
%  node(X),
%  road(Y),
%  kilometrePoint(Z).

roadConnected('Covaron','A8',148).
```

```
roadConnected('Santurtzi','A8',138).
roadConnected('Santurtzi','BI-Lam',7).
roadConnected('Barakaldo','A8',123).
…
% Adyacent relationship
adyacent('Covaron','Santurtzi').
adyacent('Santurtzi','Barakaldo').
…
```

Se tiene también información acerca de las incidencias que hayan podido surgir en las carreteras, así como de su gravedad. El nivel de inferencia utiliza toda esta información para calcular la mejor ruta que lleva a un pueblo o ciudad concreta del área del agente. El formato con el que se almacenan las incidencias es:

Incidencia (Carretera, Punto Kilométrico, Sentido, Gravedad)

El campo *Sentido* vale 2 si la incidencia afecta a ambos sentidos, 1 si la incidencia se encuentra en el sentido en el que avanzamos por los puntos kilométricos de la carretera y 0 si en el sentido en que se encuentra vamos retrocediendo por los puntos kilométricos de la carretera. El campo *Gravedad* contiene un número, que a mayor valor, indica una incidencia más grave. Por defecto asumiremos el valor 1 (sin incidencias) para un tráfico normal, 2 para un tráfico medio y 5 para un tráfico alto (incidencia grave). Estos valores han sido tomado arbitrariamente, en el caso de el campo gravedad, se podrá estudiar posteriormente el valor numérico a utilizar, para ajustar convenientemente el sistema (elección de la ruta más rápida, definición de rutas que el sistema intentará tomar por defecto…)

2.2 Módulo de acción: Inferencia

A partir de la base de conocimiento cada agente inferirá la ruta óptima entre dos ciudades, en concreto se utiliza un algoritmo de Dijkstra modificado que trabaja con el grafo contenido en el nivel de conocimiento. Este algoritmo calcula la ruta óptima entre dos nodos, usando para ello un grafo no dirigido.

El algoritmo recorre el grafo desde el origen hasta encontrar el destino, explorando siempre la rama del mismo que tenga menor peso total (los pesos en este caso representan las distancias entre nodos, por lo que al final obtendremos la ruta más corta)

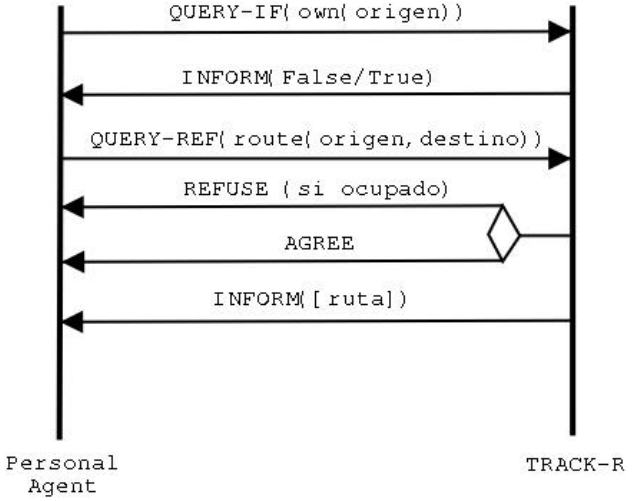
```
dijkstra(Current,Destiny,ActNode,Route):-
            findall(AD,ady(ActNode,AD),Adyacents),
            getRoute(Current,ActNode,RouteIn),
            getDist(Current,ActNode,Dist),
            distance(Adyacents,ActNode,Dist,RouteIn,AdyacentsDist),
            remove(Current,AdyacentsDist,CurrentX,Candidates),
            append(CurrentX,Candidates,Select),
            getBest(Select,(Best,DistBest),CandidatesAux),
            (equal(Best,Destiny) ->
                  getRoute(Select,Best,Route1),
                  append(Route1,[Best],Route)
                  ;
                  getRoute(Select,Best,Route2),
                  append(Route2,[Best],RouteAux),
                  append([(Best,DistBest,RouteAux,visited)],CandidatesAux,CA),
                  dijkstra(CA,Destiny,Best,Route)).
```

La distancia entre nodos se calcula restando los puntos kilométricos de los dos nodos en la misma carretera y teniendo en cuenta que si encontramos una incidencia entre dos nodos se sobrecargará esa distancia (multiplicando, por ejemplo, la distancia de entre los nodos por el valor de la incidencia) incitando así al sistema a que elija otro camino si es posible.

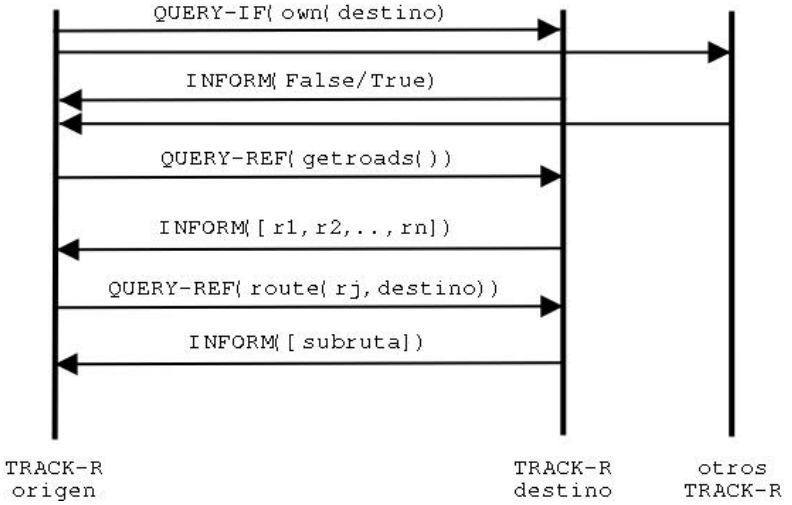
2.3 Módulo de Coordinación

Este nivel contiene actualmente los protocolos de coordinación entre los agentes y los procedimientos necesarios para encapsular e interpretar la información en los correspondientes mensajes ACL con los que se comunicarán los TRACK-R. Este módulo se ha desarrollado utilizando las librerías y las características del entorno JADE.

Hay dos tipos de comunicaciones necesarias en el sistema, la comunicación entre el PERSONALAGENT con el sistema para enviar peticiones y recibir resultados y la comunicación entre los TRACK-R necesaria para calcular rutas complejas (que atraviesen áreas pertenecientes a varios agentes). En el primer caso el PERSONALAGENT buscará el agente que tiene la información del nodo origen que se convertirá de esta manera en el agente encargado de hallar la solución, devolviéndola mediante el correspondiente mensaje INFORM.



La comunicación entre agentes TRACK-R plantea diversos escenarios, dependiendo de la información que compartan el agente que conoce el origen de la ruta y de la que conozca el destino. El primer caso se da cuando el agente que conoce el origen comparte un nodo (ciudad o cruce de carreteras) con el agente que conoce el destino, entonces la ruta entre el origen y el destino pasará forzosamente por este nodo. Si lo que comparten ambos agentes es una carretera (probablemente la situación mas común) la ruta estará compuesta por las dos sub-rutas desde el origen hasta esa carretera, y desde la carretera al destino.



También podría suceder que los agentes no compartieran nodos ni carreteras, entonces tendríamos que buscar un tercer agente que compartiera nodos y carreteras con cada uno de los dos agentes iniciales, dividiendo el problema en dos, y así con

todas las posibilidades de conexión entre los agentes. Este es el caso con el que estamos trabajando en este momento y que ya necesita de otros mecanismos para gestión de la coordinación, y que hagan posible la selección, negociación y aceptación de las tareas delegadas a otros agentes.

Además utilizamos un sistema básico de control de incidencias en el que cada TRACK-R lee de un fichero las incidencias que hay actualmente en sus dominios, y las tiene en cuenta para cada consulta. Para ello hemos añadido en el PersonalAgent un menú que permita modificar las incidencias en los dominios de cada agente.

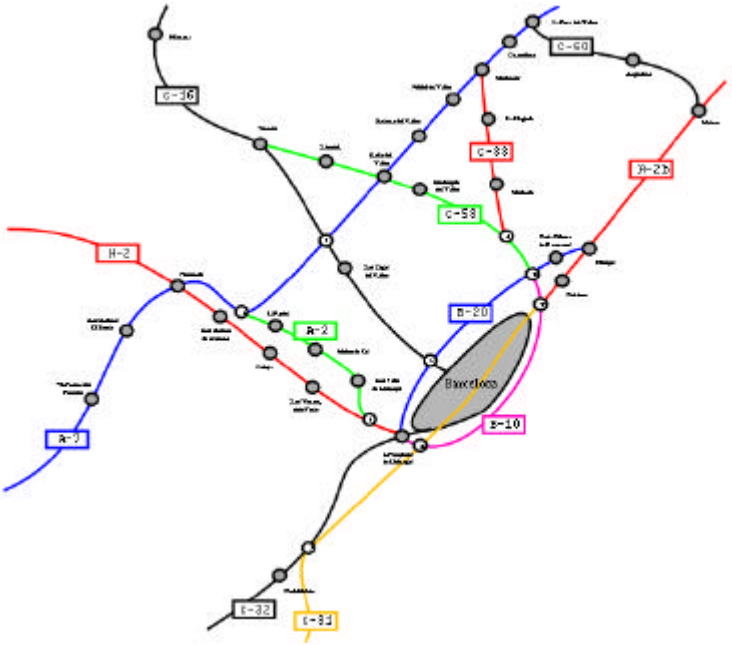


Figura 2: Nodo de Barcelona y sus alrededores

3 Conclusiones y Trabajo Futuro

Hasta el momento nuestro objetivo ha sido el planteamiento de los componentes del modelo de coordinación a desarrollar y el diseño tanto de los agentes TRACK-R como de los agentes personales (que se replican cada vez que hay un nuevo usuario). Actualmente se está evaluando este diseño mediante diferentes pruebas de coordinación así como depurando las bases de conocimiento de los mismos. También se está contactando con otros nodos de AgentCities para que distintos agentes se comuniquen con los TRACK-R.

A continuación se pretende identificar nuevas tareas necesarias para que los agentes inteligentes aumenten la calidad del servicio ofrecido hasta este momento. Finalmente

indicar que es nuestra intención aumentar la capacidad de interacción de los PERSONALAGENT dotándoles de nuevas funcionalidades para el diálogo hombre-agente, ya que actualmente son unas meras interfaces.

Referencias

1. Bueno F, García Serrano A., J. Correas, D. Teruel, Deployed architecture for an intelligent web-assistant, in Challenges in open environments, LNCS; Vol. : LNAI subseries, Springer 2003, en prensa.
2. Burmeister, B., Sundermeyer, K. Cooperative problem solving guided by intentions and perception. In Werner, E. and Demazeau, Y., editors, Decentralized AI 3 (MAAMAW-91) pages 77-92. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands. (1992).
3. Garcia-Serrano, A., Ossowski, S. Inteligencia Artificial Distribuida y Sistemas Multiagentes. Revista Iberoamericana de Inteligencia Artificial**.** Numero 6, Volumen 2, Otoño (1998)
4. Hernández, J., M.; Ossowski, S.; García-Serrano, A: Evaluating Multiagent Co-ordination Architectures - A Traffic Management Case Study In: 34th Hawaii International Conf. on System Sciences, Hawaii (HICCS), EEUU. (2000)
5. Jennings, N.R., Sycara, S., and Wooldridge, M. A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent Systems I. Kluwer, (1998) 275-306
6. Ossowski S., Cuena J. And García-Serrano A. : A case of Multiagent Decision Support: Using Autonomous Agents for Urban Traffic Control. Proc. Iberoamerican Conf. of Artificial Intelligence, IBERAMIA (1998).
7. M. Robledo, R. Fuentetaja, Ana M. García Serrano y J. Hernández, Propuesta de una arquitectura multiagente para gestión de tráfico, Actas de IBERAMIA Conferencia Iberoamericana de Inteligencia Artificial, págs:127-136, 2002
8. [www.agentcities.org]
9. [www.clip.dia.fi.upm.es]
10. [sharon.cselt.it/projects/jade]

Gestión de Sensores mediante Sistemas Multiagente

José M. Molina

GIAA. Departamento de Informática.
Universidad Carlos III de Madrid, Spain.
Avda. Universidad 30, 28911-Leganés (Madrid).
e-mail : molina@ia.uc3m.es

Resumen – Los sistemas de vigilancia actuales han evolucionado hacia complejos sistemas de información, siendo capaces de aportar al operador gran cantidad de datos obtenidos a través de una red de sensores de características heterogéneas, distribuidos espacialmente. Este trabajo pretende describir una arquitectura que soporta la distribución de tareas entre los sensores basada en sistemas multiagente.

1. Introducción

Los sistemas de adquisición de datos basados en múltiples sensores aparecen en los sistemas de vigilancia aérea [12], en los sistemas de vigilancia basados en cámaras móviles e incluso en los sistemas de robótica de última generación. Los sistemas de vigilancia aérea están compuestos por varios sensores (radares primarios y/o secundarios, sensores de escucha pasiva, sensores acústicos, sensores de imágenes, etc.) que proporcionan datos de cada móvil en el entorno (detecciones, atributos, pistas) que son fusionados en el centro de fusión [2]. En el diseño de estos sistemas aparecen dos tipos de problemas dependientes entre sí: el primer problema, denominado fusión de datos, define cómo se deben combinar de forma óptima los datos provenientes de las distintas fuentes [14]; el segundo problema, denominado gestión de multisensores, parte de los resultados obtenidos en el proceso anterior y determina cómo obtener el funcionamiento conjunto deseado del sistema controlando el funcionamiento individual de cada sensor [5]. Este trabajo se centra en el segundo problema, la gestión coordinada de los distintos sensores [10]. La gestión de un conjunto de sensores permite mejorar el proceso de adquisición de datos del entorno vigilado considerando dos tipos de información: la de carácter global o de alto nivel, de la que se dispone a través del proceso de fusión, y la de carácter local, que es interna a cada sensor. La cantidad y variedad de información de los modernos sistemas de vigilancia hace que la gestión sea irrealizable utilizando como núcleo central del sistema a un operador humano [7]. Para obtener una descripción del entorno más compleja se realiza una interpretación conjunta de toda la información recibida mediante técnicas de Inteligencia Artificial [11]. El desarrollo tecnológico ha llevado al aumento de la potencia de estos sistemas: por un lado se ha dotado a cada sensor de una capacidad de decisión que permite suministrar al centro de proceso una información semiprocesada localmente ; y por otra parte se buscan estrategias de coordinación que permitan al sistema global aprovechar conjuntamente la inteligencia que controla el funcionamiento de cada sensor [10].

El objetivo central de este trabajo es el desarrollo de un sistema de gestión que permita coordinar una red de sensores genéricos desplegados en un espacio de vigilancia. Se pretende que cada sensor presente un alto grado de autonomía, es decir, que la decisión última sobre las tareas a realizar sea tomada en su propio sistema de gestión. El proceso de coordinación debe organizar el flujo de informaciones de tal manera que la comunicación entre los distintos gestores locales a cada sensor y el centro de fusión se traduzca en una optimización en el uso global de los recursos. La arquitectura que da soporte al proceso de coordinación hará uso de técnicas de Inteligencia Artificial Distribuida [3], en particular de la teoría de multiagentes. Cada nodo de la red de sensores puede considerarse como un agente que colabora con el conjunto de agentes para la resolución de un determinado problema (gestión coordinada). La capacidad de decisión de cada uno de los agentes se aborda mediante sistemas basados en lógica borrosa [8] [11]. En particular, se desarrolla un sistema que permite priorizar el conjunto de tareas que debe realizar el sensor a partir de los datos medidos y de informaciones colaterales, como pueden ser informaciones meteorológicas o de terreno [6]. La elección de sistemas borrosos para representar la incertidumbre en el gestor se apoya en la robustez y la capacidad de generalización de estos sistemas.

2. Gestión de sensores mediante Agentes

Una red de vigilancia está compuesta por un conjunto de sensores, todos ellos conectados posiblemente a un único nodo al que envían los datos captados [2]. Los sensores que componen la red pueden ser a efectos de gestión pasivos o activos, es decir carecer o no de un gestor. Los primeros no gestionan el conjunto de tareas por lo que no adaptan su funcionamiento al entorno que los rodea. Los sensores activos son aquellos que permiten un control sobre el funcionamiento del sensor, de este modo las tareas que debe realizar dependen en gran medida del entorno actual en el que se encuentra el sistema. El nodo que centraliza toda la información captada por cada sensor es el centro de fusión. En una red de multisensores, además de las ventajas obtenidas mediante la integración de información proveniente de diversas fuentes, debe considerarse la posibilidad de coordinar todos los sensores para optimizar la respuesta global del sistema. Las funciones que pueden ser gestionadas para un sensor genérico son [14]: gestión espacial, gestión de modo, gestión temporal y la gestión de datos a comunicar. El criterio de gestión se define de manera que se utilicen los sensores de forma óptima para una determinada aplicación. El criterio de optimización se define en función de la aplicación. El funcionamiento del gestor se basa en parámetros cuantificables como son: probabilidad de detección, calidad de las pistas, identificación de los blancos [7], etc.. Una de las ventajas de la gestión coordinada de sensores es la posibilidad de establecer comunicaciones entre los sensores, de manera que la información de un sensor fluya hacia otro. En este sentido es necesario implementar un sistema que permita por un lado emplear las detecciones realizadas en un sensor para dirigir la atención de otro sensor hacia ese blanco, es lo que se denomina cueing, y por otro trasladar tareas desde un sensor hacia otro, es lo que se denomina hand-off, [1].

Los procesos involucrados en la gestión del sistema dependen de las posibilidades de gestión de los sensores y de la arquitectura del sistema. En general la arquitectura implementada para la gestión coordinada se basa en la centralización de todas las decisiones en el centro de fusión por ser éste el sistema con mayor información y por lo tanto con capacidad de tomar decisiones que engloban a todos los sensores del sistema. La existencia de sensores inteligentes capaces de desarrollar su propio esquema de toma de decisiones abre la posibilidad de acometer la arquitectura desde otro punto de vista, una arquitectura descentralizada. Este tipo de arquitecturas puede aplicarse de forma directa sobre una red de sensores ya que la propia red es distribuida espacialmente [13].

Una arquitectura multiagente permite gestionar una red de sensores inteligentes sin la necesidad de centralizar en un único punto el sistema gestor. La comunicación de las decisiones entre nodos provocará que la solución global sea encontrada mediante sucesivas propuestas en el tiempo. En el problema de coordinación de agentes, tan importante es el proceso de razonamiento interno como el proceso de comunicación. El proceso de razonamiento interno consistirá en la toma de decisiones y en la determinación de la información que debe ser compartida. El proceso de comunicación debe prefijar cómo y cuando debe producirse la comunicación entre los nodos. El proceso de comunicación en arquitecturas más centralizadas se limita a comunicar cada sensor con el centro de fusión. De este modo, la única comunicación posible entre los sensores se produce a través del centro de fusión. En la gestión descentralizada, sin embargo, cada sensor es capaz de gestionar la información que recibe, bien del centro de fusión, bien de otro sensor. Es decir, se establecen comunicaciones entre los sensores de manera que pueden tomarse decisiones entre los sensores en las que no toma parte el centro de fusión. La definición de las tareas en una arquitectura multiagente es igual que en aquellas con un grado mayor de centralización, existen tareas de sistema y de sensor. Las tareas de sistema son aquellas tareas que deben ser realizadas mediante la cooperación de varios sensores, mientras que las tareas de sensor son las internas a cada sensor. Cuando se trabaja con agentes, las tareas de sistema pueden ser enviadas por cualquier nodo cuando este infiere la necesidad de realizar una tarea de forma colectiva. Aunque el centro de fusión tiene la visión más completa del problema no se restringe al centro de fusión la capacidad de decidir las tareas de sistema.

En resumen, la comunicación, el qué y el cómo, entre los diversos agentes es muy importante a la hora de definir un sistema multiagente. En el caso de los multisensores toda la información que se puede transferir entre los nodos de la red es de dos tipos: de bajo nivel, numérica, resultado de las tareas de sensor correspondientes (medidas) o datos internos del sensor, de alto nivel, negociación de tareas (peticiones/aceptaciones para/de la realización de tareas de sistema) o evaluación de la situación del sensor.

3. Arquitectura Multiagente

La red de agentes está compuesta por dos tipos de agentes: los sensores y el centro de fusión. De aquí en adelante a cada sensor se le denominará agente sensor y al

centro de fusión agente fusión. Los agentes son nodos de la red con su propio proceso de toma de decisiones y del que puede surgir la necesidad de una comunicación con otro agente para la realización de tareas. El agente sensor tiene como competencias: la adquisición de los datos sobre las pistas y los sectores asignados, la gestión de sus propias tareas, la gestión de la comunicación con el resto de nodos de la red. El agente fusión tiene como competencias: la integración de todos los datos proporcionados por los agentes sensores, la gestión de un conjunto de tareas de sistema que deben ser realizadas por los agentes sensores, la gestión de la comunicación con lo agentes sensores de modo que el resultado final sea la obtención de unos objetivos de ejecución para las tareas de sistema.

La interconexión entre los agentes determina el conocimiento sobre otros agentes. Por ejemplo, si la conexión física de los agentes se realiza entre los agentes sensores y el agente fusión entonces resultaría imposible la existencia de conocimiento entre los agentes sensores y toda la información debería ser procesada en el agente fusión y posteriormente distribuida. Si se permite una interconexión global las posibilidades de comunicación son de todos los agentes con todos los agentes, y por lo tanto cada agentes deberá representar las capacidades de cada uno de los otros en la red. La importancia de la interconexión cobra especial relevancia a la hora de la comunicación de información entre agentes sensores adyacentes, puesto que en el límite de su cobertura existen gran cantidad de datos comunes. Es por ello que no se propone una interconexión global, que si bien podría proporcionar las mayores posibilidades de comunicación es, en la práctica, irrealizable, sino una interconexión parcial donde los sensores adyacentes se encuentren interconectados de manera que puedan apoyarse unos a otros sin necesidad de que el agente de fusión deba tomar esa decisión. Es decir, los agentes sensores conocen al agente fusión y a los agentes sensores con los que es posible que tengan informaciones comunes, el agente fusión tendrá un conocimiento de todos los agentes sensores.

En paralelo con la comunicación numérica de pistas y sectores que se realiza entre los agentes sensores y el agente fusión, se realiza una comunicación a nivel de concepto entre los agentes. El agente fusión tiene la visión más completa del escenario actual y por lo tanto es el único que puede trazar una estrategia de ejecución entre los agentes sensores. La capacidad de los sensores para tomar sus propias decisiones y de comunicarse entre ellos libera al sistema de muchas tareas inútiles. El agente fusión únicamente debe asegurar que la tarea de sistema se está realizando con los objetivos previstos y, no preocuparse de qué agentes sensores están realizando la tarea ni con qué objetivos. La coherencia del sistema, su comportamiento, se monitoriza desde el agente fusión donde se evalúa si el conjunto de agentes sensores está realizando las acciones precisas para alcanzar el nivel de objetivos deseado.

Las funcionalidades de los agentes de la red cubren la ejecución de tareas, la capacidad de gestionar a alto nivel dichas tareas y la posibilidad de comunicarse con otros agentes. Para reflejar estas tres funcionalidades la arquitectura de los agentes sensor y fusión de la red se ha subdividido en tres niveles: nivel de ejecución, nivel de planificación, nivel de comunicación. En el nivel de ejecución se encuentran las funciones que realiza el agente y que se corresponden con las tareas que puede llevar a cabo. Dependiendo del agente estas funciones son variables y definen sus

capacidades. En el nivel de planificación se realiza la interpretación de la situación y se decide que tareas son más importantes de todas aquellas que se pueden realizar. Este nivel realiza la gestión a alto nivel del agente y reproduce el comportamiento del agente cuando éste no se encuentra en una red de agentes. En general al nivel de planificación le es suministrada información colateral compuesta por toda aquella información ajena a la situación que se está monitorizando. Esta información incluye la proveniente del operador y de datos meteorológicos o de terreno y toda aquella información que es suministrada por otros agentes sobre estos aspectos.

El último nivel, de comunicaciones, se encarga de gestionar el proceso de comunicación entre los agentes a partir de los datos obtenidos en los otros dos niveles. El conjunto de mensajes que se pueden transmitir, así como las respuestas a los mensajes recibidos se deben gestionar para obtener un proceso de decisión distribuido que conduzca a la coordinación de los agentes. Los mensajes recibidos estarán compuestos de peticiones de ejecución o de informaciones de otros agentes. En el caso de las peticiones de ejecución, el nivel de comunicaciones debe transmitir al nivel inferior el resultado del proceso de negociación con otros agentes, es decir, los parámetros con los que se desea ejecutar la tarea. Las informaciones recibidas sobre tareas exclusivas del agente deberán incluirse en el proceso de razonamiento de esas tareas. Una vez comunicada, la gestión de la tarea es responsabilidad del gestor de tareas y resulta independiente de la comunicación. Del mismo modo que el nivel de comunicaciones actúa sobre el de toma de decisiones para modificar la decisión basándose en las peticiones de otros agentes, el nivel de planificación influye sobre el de comunicaciones informando de la situación en la que se encuentra el agente y las tareas. A partir de esta información, el nivel de comunicaciones gestiona las acciones que deben realizar otros agentes, o bien realiza una comunicación de información hacia otro agente. El nivel de comunicaciones considera las capacidades del agente para dar una adecuada respuesta a las peticiones que se le envían. Así la información acerca de la situación del agente permite al nivel de comunicaciones establecer si una petición de tarea desde otro agente puede ser realizada, y si existe la posibilidad de realizarla se encargará de llevar a cabo un proceso de negociación tras el cual desglosará el resultado de la negociación en informaciones para el nivel de planificación.

4. Negociación de Tareas

Cada agente sensor gestiona sus propias tareas y calcula la prioridad de las mismas a partir de los datos que se encuentran en las pistas locales. Ahora bien, el agente fusión se encarga de recoger todos los datos del entorno en el que se encuentra el sistema multisensor, a partir de esos datos es capaz de valorar la importancia que tiene cada tarea de sistema. En este caso las tareas de sistema son aquellas tareas que pueden ser ejecutadas por un conjunto de sensores, en el instante actual o en un instante futuro, el único nodo que puede gestionar estas tareas es el agente fusión. El conjunto de todas las tareas que se llevan a cabo en el sistema multisensor puede dividirse en tres conjuntos:

- Tareas de sensor aisladas, son aquellas tareas propias del agente sensor. En la realización de estas tareas no puede recibir ningún apoyo de otro agente. Por ejemplo, las tareas de vigilancia sobre sectores que no coinciden espacialmente con ningún otro sector.
- Tareas de sistema individuales, son aquellas tareas que pueden ser realizadas por un conjunto de sensores pero que, por la situación actual, únicamente pueden ser realizadas por un agente sensor. Este conjunto de tareas se compone de aquellas tareas de seguimiento e identificación que deben ejecutarse sobre blancos que únicamente se encuentran en la zona de cobertura de un sensor. En este caso el centro de fusión sabe que dicha tarea únicamente puede ser ejecutada por ese sensor. La misma situación se produce cuando la tarea puede ser ejecutada entre varios sensores pero todos ellos se encuentran limitados por la situación en la que se encuentran, *jamming*, alta carga, etc.., y que les imposibilita realizar la tarea. Cada agente sensor habrá informado al œntro de fusión de la imposibilidad de realizar la tarea y por lo tanto este determinará que la tarea sólo puede ser ejecutada por un sensor, pasando a ser una tarea aislada.
- Tareas de sistema conjuntas, son aquellas tareas que se realizan conjuntamente entre varios agentes sensores. Por ejemplo sucede así en el caso de las tareas de seguimiento e identificación sobre blancos que se encuentran en zonas cubiertas por varios sensores o tareas de vigilancia sobre sectores compartidos por varios sensores.

Las tareas, con sus correspondientes parámetros, son comunicadas entre los agentes para iniciar el proceso de negociación. Las tareas de sensor aisladas y las individuales no entran en el proceso de negociación entre agentes sensores. En el caso de las tareas aisladas ningún agente sensor puede colaborar en su ejecución y además el centro de fusión no tiene más información que el propio sensor sobre los parámetros de la tarea. El proceso de negociación de las tareas de sistema individuales se reduce al envío de un mensaje desde el centro de fusión al sensor requiriendo la ejecución de la tarea con unos objetivos generales determinados, OG(Ti), y una prioridad dada, P(Ti).

Cada una de las tareas conjuntas, Ti, sufre un proceso de negociación para determinar las tareas de sensor que cumplen conjuntamente con los objetivos impuestos por el sistema. El proceso de negociación comienza cuando el centro de fusión, A.F., envía a los sensores, Sj, las tareas de sistema que deben ejecutar y los sensores que pueden realizarlas, {Sj}. A partir de ese momento, el proceso de negociación se produce entre los agentes sensores para determinar los parámetros particulares de cada tarea de sensor, OP(Ti), y su prioridad local, PS(Ti). Las tareas de sistema conjuntas se gestionan en el agente fusión y en los agentes sensores. De nuevo al igual que en el caso de las tareas de sistema individuales, cada sensor valora la tarea desde los datos locales y respecto a la zona de cobertura propia. El agente fusión de forma asíncrona con el resto de valoraciones realiza su propia valoración de la tarea respecto de todo el sistema. Una vez que el centro de fusión ha determinado una prioridad y unos objetivos para la tarea de sistema, la transmite a todos los

sensores que pueden realizarla conjuntamente, de manera que cada sensor conozca el resto de sensores que comparten la tarea con él. A partir de ese momento el agente fusión se desentiende del proceso de coordinación y comienza la negociación entre los agentes sensores. Durante el proceso de negociación, los sensores, en función de la situación de cada uno, propone los parámetros con los que podría ejecutar la tarea, es decir los objetivos particulares y la prioridad de sensor. El siguiente esquema resume dichos pasos representado únicamente el proceso para cada uno de los sensores que pueden ejecutar la tarea:

1.Agente fusión: REQUEST.PROPOSE(T_i, S_j, $P(T_i)$, $OG(T_i)$).
2.Agente fusión: INFORM(T_i, $\{S_j\}$).
3.Transmisión del mensaje: < A.F., S_j, $RP(T_i, P(T_i), OG(T_i))$ >
4.Transmisión del mensaje: < A.F., S_j, $I(T_i, \{S_j\})$ >
5.Agente Sensor: REQUEST_TO_DO.PROPOSE(T_i, $P(T_i)$, $OG(T_i)$)
6.Agente Sensor: SUPPLY_INFO(T_i, $\{S_j\}$).
7.Proceso de negociación entre agentes....
8.Si se acepta
8.1.ACCEPT(T_i, $P(T_i)$, $OP(T_i)$)
8.2.Transmisión del mensaje: < S_j, A.F., $AC(T_i, P(T_i), OP(T_i))$ >
9.Si no se acepta
9.1.REJECT(T_i, A.F., $P_S(T_i)$)
9.2.Transmisión del mensaje: < S_j, A.F., $R(T_i, P_{SENSOR}(T_i))$ >

El proceso de negociación entre agentes involucra una serie de actos REQUEST, ACCEPT, REFINE, REJECT en los agentes que se transforman en mensajes según la situación del agente y las reglas de ejecución de los mismos. En el apartado siguiente se analizan el proceso de negociación detallando estas reglas.

El inicio del proceso de cooperación es el envío por parte del agente fusión de dos mensajes y su conversión en actos en el agente sensor:

Transmisión del mensaje: < A.F., S_j, $RP(T_i, P(T_i), OG(T_i))$ >
Transmisión del mensaje: < A.F., S_j, $I(T_i, \{S_j\})$ >
Agente Sensor: REQUEST_TO_DO.PROPOSE(T_i, $P(T_i)$, $OG(T_i)$)
Agente Sensor: SUPPLY_INFO(T_i, $\{S_j\}$).

En el instante que determinan las reglas de control de la agenda, el agente sensor trata el acto REQUEST_TO_DO.PROPOSE. Al ejecutar dicho acto se consideran, el conjunto de reglas de cooperación que definen el comportamiento del agente y los *skills*. Las reglas de cooperación que gestionan la agenda, y que determinan la manera de cooperar de cada uno de los agentes, están definidas de la siguiente manera:

1. Si la tarea de sistema tiene una prioridad de sistema similar a la prioridad calculada localmente al sensor, la tarea es aceptada por el sensor. En este caso el agente sensor, según lo recogido en los *skills*, crea un acto ACCEPT, para enviar un

mensaje de aceptación a todos los sensores que se encuentran en el acto SUPPLY_INFO y al agente fusión.

2. Si la tarea de sistema tiene una prioridad de sistema muy diferente de la asignada localmente al sensor, la tarea se rechaza, pero no totalmente. En este caso el agente sensor, según lo recogido en los *skills*, crea un acto REFINE, con unos objetivos particulares muy bajos a todos los sensores que se encuentran en el acto SUPPLY_INFO. Al mismo tiempo se genera un acto DO, que espera la recepción de una aceptación de la tarea.
3. La regla 1 no es válida en el caso de que el sensor se encuentre ejecutando muchas tareas. La regla 3 intentará rechazar la tarea para que la ejecute otro sensor. En este caso el agente sensor, según lo recogido en los *skills*, crea un mensaje REFINE para enviar un mensaje que contiene unos objetivos más bajos a todos los sensores que se encuentran en el acto SUPPLY_INFO. Al mismo tiempo se genera un acto DO, que espera la recepción de otra propuesta y la compara con la propuesta realizada.
4. Si el sensor rechaza una tarea cuya prioridad de sistema es similar a la de sensor debido a la regla 3 y ha generado un acto DO, aceptará una nueva propuesta si los objetivos que contiene la nueva propuesta son iguales a los suyos. Esto ocurrirá si todos los sensores se encuentran muy cargados y todos mandan un mensaje para refinar la tarea de manera que se realice de forma conjunta. En este caso el agente sensor crea un acto ACCEPT para enviar un mensaje de aceptación a todos los sensores que se encuentran en el acto SUPPLY_INFO y al agente fusión. Si no es así generará un acto REJECT.
5. Si tras ejecutar la regla 2, la tarea es rechaza por todos los sensores, según lo expuesto en los *skills*, se comienza una negociación mediante un acto REFINE, para llegar a una aceptación de la tarea. En este caso, si el agente sensor llega a recibir una propuesta con unos objetivos similares a los que ha propuesto crea un acto ACCEPT para enviar un mensaje de aceptación a todos los sensores que se encuentran en el acto SUPPLY_INFO y al agente fusión. Si no es así generará un acto REJECT.

Estas reglas no se activan en un único instante, sino que se van activando secuencialmente a medida que se produce el proceso de negociación. El tratamiento de un acto y no otro viene determinado por las reglas que controlan la agenda.

5. Breve revisión histórica

El problema de la integración de información de diversos sensores abordado desde una perspectiva distribuida aparece en las primeras publicaciones de Inteligencia Artificial Distribuida [15] y en particular para la interpretación conjunta de toda la información recibida [13]. El problema de la fusión de información también ha sido aboradado desde la disciplina de la Fusión de Datos con incursiones interesante en técnicas de IAD [16]. Los problemas de coordinación de entidades computacionales distribuidas son fundamentales y aparecen desde los primeros trabajos [17]. La negociación y en particular la red de ofertas y contratos de Smith [18] es utilizada

desde los primeros trabajos en sistemas distribuidos y está íntimamente ligada a los problemas de coordinación. Una revisión actualizada de toda esta problemática puede verse en [19], donde se sistematizan todos los problemas y se detallan las diversas soluciones que han ido apareciendo a lo largo de los años. Por último reseñar que el modelo de agente y el protocolo de comunicación/negociación utilizado está fundamentado en el modelo COOPERA de Lorenzo Sommaruga [20].

6. Referencias

[1] Bar-Shalom, Y., *Multitarget Multisensor Tracking. Vols. I-II*, Artech House Inc., Norwood, MA, 1992.

[2] Besada J., J. García, A. Varona, G. González, J. M. Molina, J. Portillo. "Image-Based Automatic Surveillance For Airport Surface", 4th International Conference on Information Fusion, Fusion 2001, pp (WeA1) 11-18. Montreal, Canadá, Agosto 2001.

[3] Decker, Keirh S., "Distributed Problem-Solving Techniques: A Survey", IEEE Transactions on Systems, Man, and Cybernetics, Sep/Oct 1987.

[4] Manyika J., Durrant-Whyte H., *Data Fusion and Sensor Management a decentralized information-theoretic approach*, Ellis Horwood, 1994.

[5] Molina J.M., J. García, F.J. Jiménez, J.R. Casar, "Cooperative Management of a Net of Intelligent Surveillance Agent Sensors",. International Journal of Intelligent Systems. vol 18, nº 3 pp 279 - 307, 2003.

[6] Molina J.M., J. García, F.J. Jiménez, J.R. Casar, "Surveillance Multisensor Management with Fuzzy Evaluation of Sensor Task Priorities". Engineering Applications of Artificial Intelligence. vol 15 nº6 pp 511 – 527, Diciembre 2002.

[7] Molina J. M., J. García, A. Berlanga, J. Besada, J. Portillo. "Automatic Video System for Aircraft Identification", 5th International Conference on Information Fusion, Fusion 2002, pp 1387-1394. Maryland, USA. Julio 8-11, 2002.

[8] Molina J. M., F. J. Jiménez, J. R. Casar. "Fuzzy Management of Surveillance Sensors", 37th IEEE International Conference on Decision and Control, pp 245-250. Florida, EEUU. Diciembre 1998.

[9] Molina J. M., F. J. Jiménez, J. R. Casar. "Negotiation in a MultiAgent System of Surveillance Sensors based on Fuzzy Priorities", 4th International Conference on Information Fusion, Fusion 2001, pp (ThB2) 25-32. Montreal, Canadá, Agosto 2001.

[10] Molina J. M., F. J. Jiménez, J. R. Casar. "Cooperative Management of Netted Surveillance Sensors", IEEE International Conference on Systems, Man and Cybernetics, pp 845-850. Orlando, EEUU. Octubre 1997.

[11] Molina J. M., F. J. Jiménez, J. R. Casar. "Fuzzy Reasoning for Multisensor Management", IEEE International Conference on Systems, Man and Cybernetics, pp 1398-1403. Vancouver, Canada. Octubre 1995.

[12] Portillo J., J. Besada, J. García, J. M. Molina, A. Varona, G. González. "MARIA: Preliminary Results of an AENA R&D Project on Airport Surface Surveillance Based on CCTV", International Conference on Airport Surveillance Sensors, pp 138-145. París, Francia, Diciembre 2001.

[13] Wesson R. et all "Network Structures for Distributed Situation Assessment", *Readings in Distributed Artificial Intelligence*, Ed. Alan H. Bond and Les Gasser, Morgan Kaufmann 1988.
[14] Waltz, E., J. Llinas, *Multisensor Data Fusion*, Artech House Inc., Norwood, MA, 1990.
[15] Bond, A., Gasser, L., (editores), *Readings in Distributed Artificial Intelligence*, P.Kaufmann CA, 1988.
[16] Llinas, A., "Blackboard Concepts for Data Fusion Applications", en *International Journal of Pattern Recognition and Artificial Intelligence*, vol 7, nº 2, pp 285-308, Abril 1993.
[17] Durfee E.H., Lesser V.R., Corkill D.D., "Cooperative Distributed Problem Solving", Cap. XVII, *The Handbook of AI (Vol IV)*, ed. Barr A., Cohen P.R., Feigembaum E.A., Addison-Wesley 1989.
[18] Davis, R., Smith, R.G., "Negotiantion as a Metaphor for Distributed Problem Solving", *Artificial Intelligence* vol 20, nº 1, pp 63-109, 1983.
[19] Ferber, J. *Multi-Agent Systems. An introduction to Distributed Artificial Intelligence*, Addison-Wesley 1999.
[20] Sommaruga, L., Avouris, A., Van Liedekerke, M. , "An environment for experimenting with interactive cooperating knowledge based systems", en *Reasearch and Development in Expert Systems, vol VI*, editado por Nigel Shadbolt, Cambridge University Press, Cambridge, UK, 1989.

Desarrollo de servicios turísticos a usuarios

M. Bätzold, M. Navarro, V. Julian and V. Botti

Departament de Sistemes Informàtics i Computació,
Universitat Politècnica de València, Spain
matthias.baetzold@rwth-aachen.de, {mnavarro,vinglada,vbotti}@dsic.upv.es

Abstract. En este artículo se presenta una aplicación enfocada al turismo mediante el empleo de la tecnología de agentes. En concreto, la aplicación permite a un usuario planificar su posible estancia en una ciudad para poder visitar distintos lugares de interés. La aplicación puede ser ejecutada, desde el punto de vista del usuario, tanto desde un ordenador personal como desde un dispositivo móvil, ofreciendo de esta forma una mayor flexibilidad.

Keywords: agentes, sistemas multiagente, turismo, scheduling.

1 Introducción

Imaginemos por un momento que fuésemos un turista en una ciudad extranjera. En ese caso nos gustaría por ejemplo, poder ir a ver monumentos, museos, o disfrutar de un buen restaurante. El problema es poder disponer de toda la información actualizada de los distintos lugares de interés desde el punto de vista del usuario como por ejemplo un restaurante que no sea muy caro o que sirva un determinado tipo de comida. Son evidentes las ventajas de disponer de una solución a nivel de usuario que permita tanto la búsqueda como la planificación de distintas actividades de tipo turístico para un día o un fin de semana. Partiendo de esta idea inicial en este artículo se presenta una solución empleando para ello el paradigma de sistemas multiagente.

En los últimos años, dentro de la inteligencia artificial, se está empleando el paradigma de los sistemas multiagente, tomando gran auge entre los investigadores del área. Dicho paradigma resulta especialmente adecuado para la construcción de sistemas complejos basándose en la utilización de entidades que pueden actuar de forma autónoma. Los sistemas basados en agentes y sistemas multiagente se han empleado desde su inicio en diferentes campos como control de procesos [1], robots móviles [2], aplicaciones comerciales [3], etc.,

Recientemente han aparecido iniciativas como la de AgentCities [4], en lo que se refiere a aplicaciones de sistemas multiagente relacionadas con el ofrecimiento de servicios a usuarios en entornos abiertos. Éste hecho ha favorecido la aparición de aplicaciones reales que emplean la tecnología de agentes y no meros prototipos o complejos estudios teóricos. Entre otros ejemplos, podemos destacar una

gran variedad de aplicaciones que proporcionan servicios de búsqueda de restaurantes, por ejemplo, la propuesta por la universidad de Girona [5], aplicaciones que recomiendan teatros [6], información sobre los posibles transportes públicos disponibles para poder ir de un lugar a otro determinado [7], servicios médicos [8], etc.

Partiendo de la problemática planteada al inicio, el artículo muestra un sistema multiagente capaz de ofrecer servicios turísticos a usuarios en una determinada ciudad incorporando la capacidad de planificar diferentes actividades en un determinado día de una forma flexible y dinámica. Las características fundamentales del sistema propuesto se muestran en el siguiente punto. De esta forma, el resto del artículo se estructura de la siguiente forma: En la sección dos se presenta el sistema multiagentes propuesto. En la sección tres se describen los principales servicios, ofrecidos dentro del sistema. Finalmente, en la sección cuatro se presentan las conclusiones.

2 Sistema

2.1 Arquitectura del sistema Multi-Agente

El sistema está dividido fundamentalmente en tres tipos de agentes: UserAgent, SightAgent y BrokerAgent. El UserAgent contienen la GUI para el usuario proporcionando la interfaz para el turista. El BrokerAgent es el encargado de ofrecer servicios al UserAgent y disponer actualizados los lugarés de interés exis- - tentes en la ciudad. En princípio, sólo se dispone de un BrokerAgent, pero podríamos disponer de varios especializados en distintos tipos de lugares de interés de la ciudad. Cada SightAgent almacena la información necesaria de un lugar de interés específico. Básicamente el sistema funciona de la siguiente forma: El BrokerAgent recibirá de los UserAgents las preguntas o acciones a realizar, y él se las trasmitirá a los SightAgents correspondientes. Los SightAgents compararán la información recibida con sus propios datos y en el caso de que tanto la información recibida, así como sus datos coincidan, enviará al BrokerAgent un paquete de datos, y el BrokerAgent será el encargado de hacersela llegar al UserAgent correspondiente.

En la figura 1 se muestran los tres tipos de agentes y la comunicación entre ellos. Esta comunicación puede sólo involucrar a los UserAgents y al BrokerAgent cómo por ejemplo el servicio *planificar día*, sólamente entre el BrokerAgent y los SightAgents como el servicio *ping*, o entre los UserAgents y los SightAgents como el servicio *reserva*. Detalles sobre los servicios ofrecidos dentro del sistema son presentados en la sección 3.

2.2 Ontología

Existen en la literatura una serie de ontologías que permite definir muchos aspectos relacionados con el turismo, como por ejemplo, restaurantes [9], pub's, etc., o que aportan conocimiento de scheduling [10]. En un principio se analizaron y

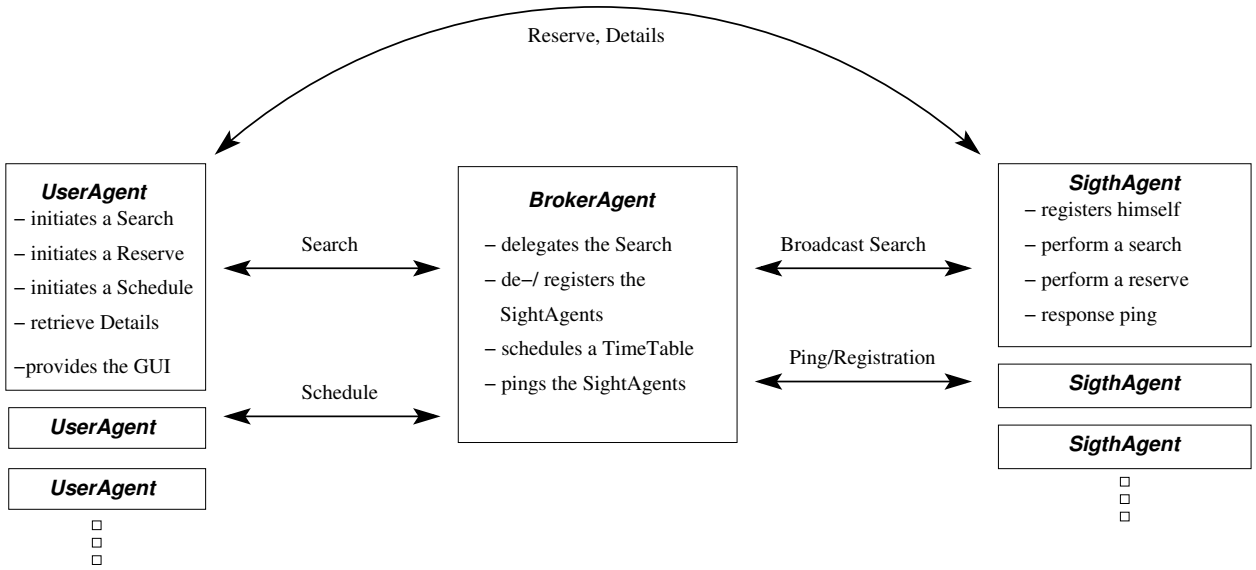


Fig. 1. Los tres tipos de agentes usados y sus respectivas comunicaciones

compararon distintas ontologías que disponían de conceptos turísticos necesarios en la aplicación, pero ninguna abarcaba nuestras necesidades de forma completa. En su lugar se adoptó el criterio de reutilizar parte de las ontologías conocidas y completar con nuevos conceptos. Además era imperativo añadir conceptos de scheduling a la ontología a utilizar, conceptos que en ningún caso se encontraron integrados en las ontologías turísticas vistas hasta ahora. La ontología es un elemento importante que permite que los distintos agentes del sistema compartan un conocimiento común de la información que van a manejar [11], por tanto su elección debe de ser cuidadosa. El sistema implementado en esta aplicación usa la ontología TurOnt, creada usando el editor Protégé [12] y que ofrece en primer lugar una descripción detallada de lugares turísticos, de ocio, hosteleros, etc., y a su vez un completo conocimiento sobre scheduling, necesario para realizar los servicios de planificación que proporciona el sistema. En la figura 2 se puede observar una parte de los conceptos que se definen en la ontología, en concreto los que hacen referencia a lugares de interés de tipo restaurante.

La ontología no sólo debe de aportar a los agentes un conocimiento común, si no también facilitar la comunicación entre ellos. Para ello, TurOnt aporta una serie de predicados y acciones que se intercambiarán mediante mensajes y que servirán para indicar cambios, pedir acciones sobre el sistema, etc.

Los predicados y acciones más significativos de TurOnt son:

– SimpleSearch : El UserAgent envía este predicado al BrokerAgent que a su vez lo reenvía a todos los SightAgent. El SightAgent comprueba que los parámetros del predicado coinciden con sus propios atributos, en el caso

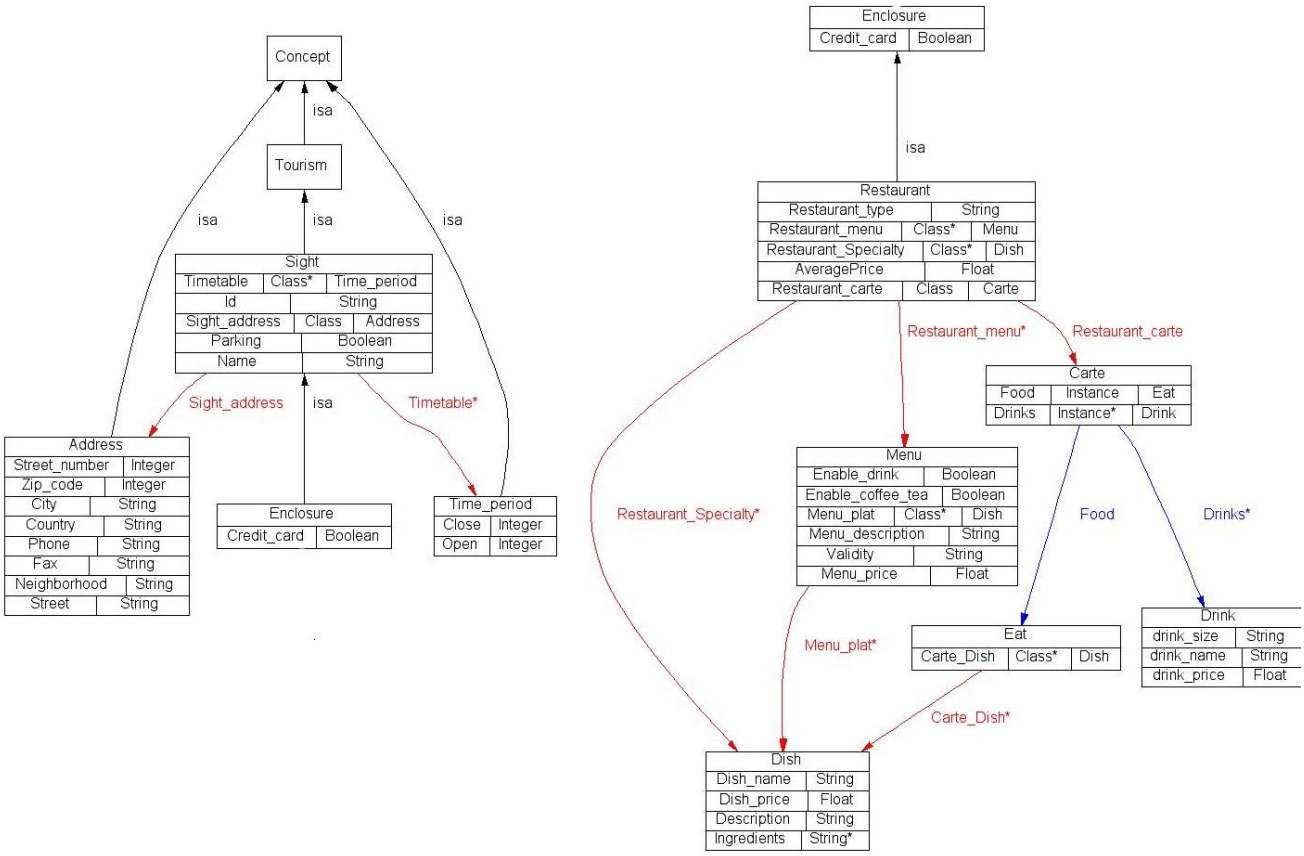


Fig. 2. Definición de la clase restaurant dentro de la ontología TurOnt.

de que concuerden, el SightAgent reenvía otro mensaje, con el predicado SimpleSearch dentro de él, de vuelta al UserAgent.

- SimpleSearchList: Este predicado es el mismo que el SimpleSearch pero se usa en el caso de que el objeto que se envía tenga múltiples instancias.
- SightDetails: Este predicado es enviado por el UserAgent en el caso de que desee ampliar la información recibida del SightAgent envía este predicado. Para ello usa como parámetro la información que se desea recibir.
- BookSight: Este predicado es usado por el UserAgent para preguntar al SightAgent si es posible poder realizar una reserva, por ejemplo en un restaurante.
- CreateTimeTable: Se emplea para iniciar el proceso de creación de un horario, el UserAgent envía al BrokerAgent un mensaje con el predicado CreateTimeTable y una serie de parámetros que serán usados para la realización del horario por parte del BrokerAgent.

– TimeTable: Una vez completado el horario, el BrokerAgent envía al UserAgent una lista ordenada con la planificación para que este se lo muestre al usuario.
– Reserve: Si la reserva es posible, es decir, el UserAgent ha recibido la confirmación del SightAgent de que está disponible, el UserAgent envía esta acción al SightAgent para que inicie la reserva.
– RegisterAgent: Esta acción es enviada por el SightAgent al BrokerAgent para que le registre en su lista de lugares de interés.

2.3 Implementación

La plataforma usada en este sistema es JadeLeap [13], una modificación de Jade [14] versión 3.0b11. JadeLeap permite la portabilidad de la plataforma Jade a dispositivos moviles, por ejemplo, PDA, telefonía móvil, etc. Para la comunicación se emplean los protocolos especificados dentro del estándar FIPA [15] y como lenguaje FIPA-ACL *(Agent Communication Language)*. Durante el desarrollo de la aplicación se usó una PDA, iPAQ 5450 de Hewlett Packard, con el sistema operativo Windows CE. La maquina virtual de Java usada fue CrEme [16].

Cada UserAgent del sistema puede ser instalado en un dispotivo móvil, como una PDA, o en un PC. En nuestras pruebas el UserAgent fue instalado en una PDA del tipo comentado anteriormente. Por otra parte, el BrokerAgent y cada SightAgent se han probado en distintos PC conectados todos ellos a una misma plataforma JADE.

3 Servicios

En este apartado se define los cinco servicios que por el momento ofrecen el sistema. En primer lugar, se describe los servicios principales de búsqueda, reserva y planificar día que son los que interactúan con el usuario. Los servicios restantes, ping y registrar, son totalmente transparentes al usuario y se muestran en el apartado 3.4.

3.1 Búsqueda

Ésta es la principal función del sistema, ya que proporciona el servicio de búsqueda de un sitio específico. Imaginemos que un usuario quiere encontrar un lugar especial. Para su búsqueda el usuario puede dar sus preferencias sobre los lugares que le gustan, los que son importantes para él, así, de ese modo, el resultado no es un conjunto de todos los lugares que puede visitar, sino de aquellos que realmente le interesan y están disponibles en el sistema.

La búsqueda es iniciada por un UserAgent, usando para ello la interfaz mostrada en la figura 3. Estos parámetros son fijos y dependen del lugar de interés que el usuario desea visitar. Por ejemplo, si un usuario desea visitar un

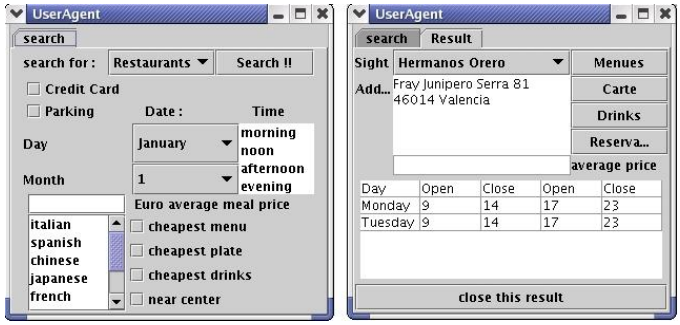


Fig. 3. Interfaz del servicio búsqueda y pantalla que muestra los resultados de la búsqueda

museo tendrá que especificar a que clase de museo desea asistir, o por el contrario, si lo que se desea es comer en un restaurante barato e italiano, especificar estos atributos mediante la interfaz de búsqueda de lugares de interés de tipo restaurante. En definitiva, cada lugar de interés que maneja la aplicación tendrá asociada una interfaz de búsqueda con unos parámetros específicos al lugar de interés. Después, el UserAgent enviará estos parámetros al BrokerAgent. El BrokerAgent identificará el lugar buscado por el usuario y solicitará información de los SightAgents que almacenen datos relacionados con ese tipo de lugares. El SightAgent comparará los parámetros de la búsqueda con sus propios datos y si coinciden enviará un mensaje al BrokerAgent con los datos solicitados, en caso contrario, el SightAgent enviará un resultado vacío. El BrokerAgent recogerá todos los resultados y mandará un mensaje al UserAgent que empezó la búsqueda con una lista de todos los SightAgent que cumplan las condiciones impuestas por él.

El sistema tiene asociado un plazo máximo que marca el fin de la espera del BrokerAgent de posibles respuestas por parte de los SightAgent. Cuando este tiempo llegue a su fin, el BrokerAgent enviará la lista al UserAgent sin esperar la respuesta de los SightAgent que aún no lo hayan hecho. La finalidad de este plazo máximo no es otra que en el caso de que un SightAgent haya caído, por el motivo que sea, un fallo en la conexión, el SightAgent se ha desconectado, etc., el sistema no se vea afectado por la espera de una contestación que nunca va a llegar. Una vez el UserAgent haya recibido la lista enviada por el BrokerAgent presentará los resultados de forma adecuada al usuario (ver figura 3).

3.2 Reserva

En este proceso, se sigue un protocolo Contract-Net. Primero, el UserAgent envía un mensaje CFP (Call For Proposal), conteniendo la hora y el número de personas para las cuales se desea hacer la reserva. Este mensaje puede ser enviado a un único SightAgent o a varios de ellos. Cuando el SightAgent recibe el mensaje compara la información que contiene con sus propios datos para ver

si la reserva es posible, es decir, comprueba que la reserva puede hacerse a esa hora y con esa cantidad de personas. En el caso de poder satisfacer al usuario, el SightAgent creará un mensaje PROPOSE conteniendo la misma información que recibió del UserAgent. Puede ocurrir que la hora solicitada esté ocupada, en ese caso el SightAgent buscará, si es posible, una hora donde pueda acomodar la petición del usuario y enviara un mensaje PROPOSE con la nueva hora. El UserAgent puede decidir si acepta cualquiera de las propuestas enviadas por los SightAgent o desestimarlas. En el supuesto de aceptar una de ellas, él enviará un mensaje ACCEPT-PROPOSAL y el SightAgent responderá con un mensaje INFORM indicando que la reserva está efectuada.

3.3 Planificar Día

El sistema también ofrece la posibilidad de que un usuario pueda planificar un periodo de tiempo con varias actividades, visitas a museos, cena en restaurante, cine, etc. Todo esto dentro de un entorno dinámico que permita al usuario cambiar los resultados propuestos por el sistema con la finalidad de modificar abiertamente el orden de las actividades, o simplemente, eliminarlas o añadir nuevas.

Internamente el servicio funciona de la siguiente forma: primero el UserAgent solicita que se realice un horario enviando para ello el predicado CreateTimeTable al BrokerAgent. Como parámetros lleva la hora inicial, final y una serie de características y restricciones que limitará la búsqueda dentro del sistema, por ejemplo, el tipo de comida que prefiere, la hora a la que suele comer, lugares que le interesa visitar, etc.. Una vez el BrokerAgent sustrae la información necesaria del mensaje recibido, enviará el predicado SimpleSearch a los distintos SightAgent del entorno solicitando que le manden sus datos. Para que el volumen de objetos que tenga que manejar el scheduler sea menor, el BrokerAgent tendrá en cuenta las características del usuario para limitar la búsqueda en los SightAgent. Por ejemplo, si el usuario tiene debilidad por la comida italiana y su deseo es ir a un restaurante donde sirvan esa comida, el BrokerAgent sólo solicitará información de restaurantes italianos omitiendo todos los demás. Cuando el BrokerAgent recibe la contestación de los SightAgent almacenará sus datos y realizará la planificación sobre ellos teniendo en cuenta las restricciones. Después de completar el horario, el BrokerAgent enviará el predicado TimeTable con la planificación como parámetro al UserAgent que a su vez presentará el resultado mediante su GUI (ver figura 4).

Después de recibir la información, el usuario mediante la interfaz vista en la figura 4, podrá modificar en ciertos aspectos la opción propuesta por el planificador, por ejemplo, si el planificador propone una visita el restaurante italiano 'Papardella' de 14:00 a 16:00, el usuario podría cambiar dicho restaurante por el restaurante 'Vella Roma' sin cambiar la hora que el planificador ha escogido. El cambio de hora no está permitido por el momento para no modificar el resto de planificación propuesta, pero eso si, se podría dejar libre cualquier hora ya que esa modificación no cambia el estado de las anteriores, simplemente deja al usuario sin actividad planificada durante ese intervalo de tiempo.

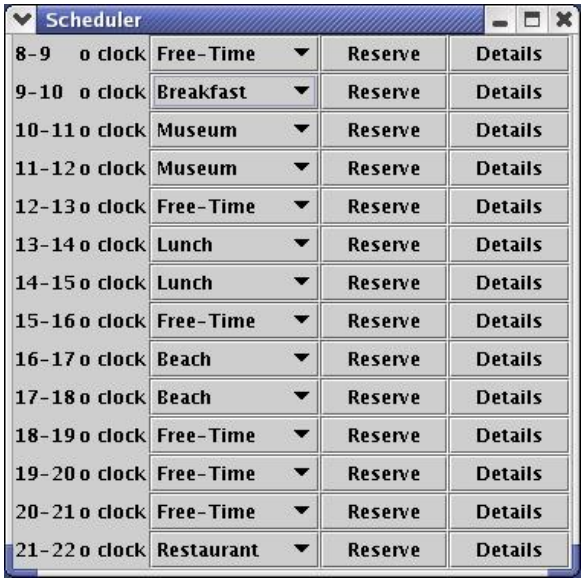


Fig. 4. Presentación de resultados del servicio planificar día.

3.4 Otros servicios

Registrar Esta acción es inicializada por el SightAgent. Si el SightAgent no se registrara, no existiría para el BrokerAgent. De esta forma sería imposible para el usuario localizarlo en el sistema. Por tanto lo primero que debe de realizar el SightAgent es mandar un mensaje SUBSCRIBE al BrokerAgent. Este mensaje contendrá como parámetros la información del tipo de lugar que maneja. Si el BrokerAgent recibe este mensaje y puede registrar al SightAgent, contestará con un mensaje CONFIRM. En este punto, habrá una conexión entre el BrokerAgent y el SightAgent, conexión que deberá comprobarse cada determinado tiempo usando para ello el servicio ping, expuesto a continuación.

Ping Este servicio, implementado por el BrokerAgent, envía cada 10 minutos una señal *PingSight* a los SightAgent. Con esta acción el BrokerAgent se asegura que los SightAgent todavía están vivos. Después de 3 intentos, si el SightAgent no ha respondido, se le borrará de la lista de SightAgent que maneja el BrokerAgent. Por otro lado, si el SightAgent no recibe un ping en 12 minutos, tratará de volver a registrarse al BrokerAgent.

En la figura 5 se muestra un ejemplo de comunicación entre los distintos agentes del sistema.

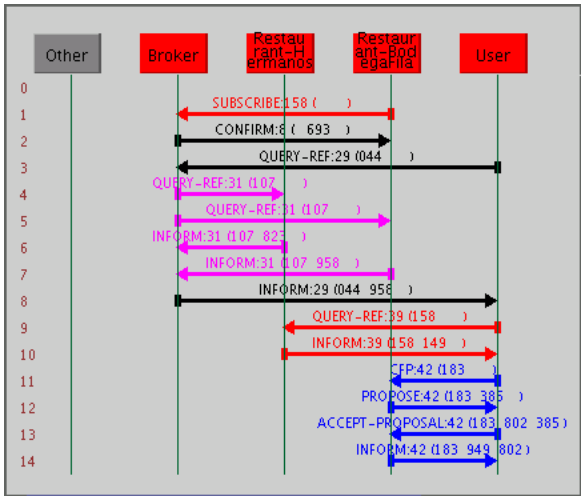


Fig. 5. Pantalla del Sniffer de JADE

4 Conclusiones

El empleo de la tecnología de agentes parece adecuada para el desarrollo de servíos automatizados a usuarios como puede ser el caso concreto de servicios turísticos. Existen actualmente diversos sistemas y prototipos que ofrecen servicios similares a los expuestos en este artículo, como por ejemplo restaurantes o museos. En estos casos se suelen tratar los servicios de una forma independiente y no integrada como aquí se plantea. Además, el sistema también permite la planificación de diversas actividades en un mismo día. Por otra parte, en la aplicación aquí presentada se da la posibilidad de acceder al usuario usando dispositivos móviles, dándole un valor añadido al sistema. Actualmente, se dispone de un prototipo del sistema totalmente funcional a falta de la incorporación de datos reales de lugares de interés de la ciudad de Valencia.

References

1. Jennings, N. R., Corera, J. M., Laresgoiti, I. *Developing industrial multi-agent systems. En Proceedings of the First International Conference on Multi-agent Systems*, (ICMAS-95), 423-430, 1995.
2. Neves, M., C. Oliveira, E. *A Control Architecture for an Autonomous Mobile Robot.* Agents'97, ACM, 1997.
3. Maes, P. *Agents that reduce work and information overload. Communications of the ACM*, 37(7), 31-40, 1994.
4. Página de Agentcities.org, http://www.agentcities.org
5. Montaner, M., López, B., Del Acebo, E., Aciar, S., Cuevas, I. *On the Integration of Restaurant Services*, workshop ID3 Barcelona (Spain). pp. 6-8 February, 2003

6. Imperial College (ICSTM) *Theatre Recommendation* http://www.agentcities.doc.ic.ac.uk/TheatreRecommendation.html
7. *Local Transport Information*, http://agentcities.agentscape.de/service/LocalTransportInformation.html
8. Moreno, A., Isern D. *Agent_based medical services within the AgentCities project.*, In Proceedings of Agents Applied in Health Care at 15th European Conference on Artificial Intelligence, 2002.
9. Ontología de Restaurantes, http://sf.us.agentcities.net/ontologies/restaurant.daml
10. Dean M., Ontología agenda, http://www.daml.org/ontologies/238
11. Noy, N. F., McGuinness, D. L. *Ontology Development 101: A Guide to creating Your First Ontology*, Stanford University
12. Página de Protègè, http://protege.stanford.edu
13. Lightweight Extensible Agent Platform (LEAP), IST-199-10211 http://leap.crm-paris.com
14. Java Agent Development Framework (JADE) , http://jade.cselt.it
15. The Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org
16. *CrEme, the Java Virtual Machine for Windows CE type devices*, http://www.nsicom.com

HeCaSe: An Agent-Based System to Provide Personalised Medical Services

D. Isern, D. Sánchez, A.Moreno, and A.Valls

Multi-Agent Systems Group (GruSMA)
Computer Science and Mathematics Department
Universitat Rovira i Virgili (URV)
ETSE. Av. dels Països Catalans, 26 (Campus Sescelades)
43007-Tarragona, Catalonia (Spain)
{disern,dsanchez,amoreno,avalls}@etse.urv.es

Abstract. In this paper we describe an application called HeCaSe, which is an agent-based system that provides medical services to its users (the citizens or the visitors of a city). This multi-agent system contains agents that have information about the medical centres, departments and doctors of a region. All these agents coordinate their tasks to provide a set of services to the user of the system, who can search for medical centres satisfying a given set of requirements, as well as access his medical record, or make a booking to be visited by a particular kind of doctor. Special care has been paid to provide personalised information using a dynamically updated user profile.

1 Introduction

The new information technologies allow people to access eficiently a wide range of data and services. Our aim is to use these tools to solve daily problems to improve the citizens quality of life. We focus our research in the health care domain using multi-agent systems technology (MAS) [12], that offers an added value in front of classical approaches such as web-based applications [3]. In this paper we describe the design and implementation of a MAS called HeCaSe that offers personalised health care services.

The rest of the paper is organised as follows. First of all, in §2 we give a general overview of the HeCaSe system. Section §3 explains how the system adapts dynamically its behavior to the user's profile. The paper finishes with a discussion of the work done and some future lines of research.

2 HeCaSe: Overview

HeCaSe is a MAS prototype that models the medical institutions of Catalonia [3]. The main design objectives are the following:

- To provide a decomposition of the problem that allows to model the real entities of the medical domain as agents, based on the structure of medical

centres in Catalonia [3] (each centre has a set of departments, and each department has a set of doctors).
- To provide an ontology for the medical domain [7, 9].
- To provide security measures that ensure the confidentiality and privacy of medical data such as Secure Socket Layer (SSL) for message ciphering and Public Key Infrastucture (PKI) and Hash functions for authenticating and user's access control [8].
- To make the developed agent services as reusable as possible by: (1) using standard languages - in this case FIPA-ACL ([6]) for agent language communication and Resource Description Framework (RDF, [4]) for content and representation of ontologies - and (2) providing detailed service models to describe the individual functioning and objective of each agent including descriptions of actions, protocols used and example messages [9].
- To implement mechanisms to allow the user to adapt his Personal Agent behavior based on his profile. This will be very useful to guide a negotiation or to make proposals according to the user's preferences (See §3).
- To provide ubiquitous access to the system from anywhere, anytime [9].

With these aims, we have implemented a MAS that offers the following features (more details in [9]):

- The user may request information about all the medical centres available in a particular geographical area.
- It is possible to book a visit to be examined by a doctor.
- As mandated by a Catalan law [2], the user has access to his medical record.
- It must be made sure that nobody can access the private medical information of the users of the system without proper authorisation.
- The user can manage his own timetable and a set of preferences in order to select the most appropriate proposal or sort a list of options in the booking negotiation protocol.
- The doctor can manage his working hours and patient's appointments. During an examination, he is able to access the patient's medical history and to update it with the result of the visit.

2.1 Architecture

The basic architecture of the MAS which has been developed in this work is shown in Fig. 1. This multi-agent system contains the following agents:

i) The user's personal agent (PA), that provides a graphical interface of the MAS to the user, allowing the access to the offered services (queries, bookings, agenda, preferences...).
ii) The broker agent (BA) is an agent that provides a gateway between personal agents and the rest of the agents in the system. It controls the access of users that are properly authenticated.

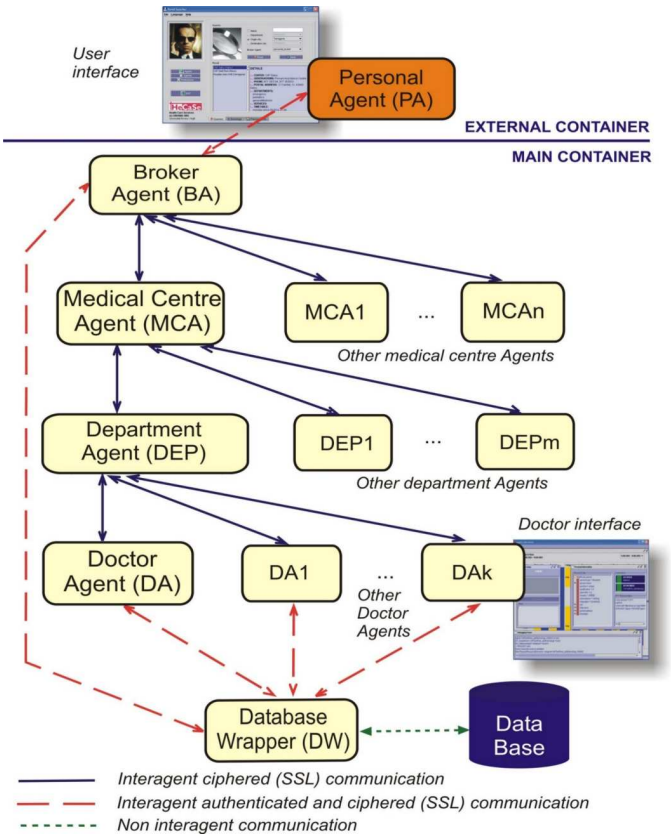


Fig. 1. Architecture of HeCaSe

iii) The information of a medical centre is divided in three levels. For each medical centre there is one medical centre agent (MCA), several department agents (DEPs, one for each department of the centre) and many doctor agents (DAs, one for each doctor of each department). The MCA has the general information of the centre (e.g. its address and opening times). Each DEP has the knowledge of a certain department (e.g. all the information of the Ophthalmology department). Each DA maintains the schedule of a given doctor, and is aware of the doctor's visiting times. They also implement a graphical interface to the doctor that is used to manage the list of pending visits with patients, to introduce the results of a medical visit and to look up the patient's personal data.

iv) The database wrapper (DW) is the agent that controls the access to a database that contains the medical records of the users.

The agents are stored in several containers [6]. The MCAs (with DEPs and DAs) and the DW are internal to the system and only the BA is accessible by

external agents. The PAs are running in the user's machine in an external container. The internal agents could be deployed physically around several machines in a real setting, e.g. each MCA (with its hierarchy of associated agents) could be running in a separate machine. Concretely, through our tests, we recommend not to run more than 100 agents on the same machine because the overhead produced by each one agent decreases the global performance.

3 Service personalisation

In order to provide more personalised services, each patient has his own profile which stores personal data (e.g. name, address, birth date, job), medical information (e.g. allergies, disabilities, medical antecedents), security elements (e.g. login, password), timetable (e.g. appointments, reminders) and preferences (e.g. ranking of doctors, medical centres and cities). The first three items are used to create the personal record that can be accessed by doctors or by the owner (after an authentication process, [8]). The timetable and the preferences are used to guide and personalise the booking negotiation process, as described in the following subsections.

3.1 Timetable plug-in

The *timetable plug-in* is a domain independent package for time management and scheduling visualisation. The list of features that this plug-in provides is the following:

- Management of appointments (an interval of time in which some kind of event occurs). Two types are distinguished: the *blocking ones* (shown on the left column of each day) that do not allow overlapping with others (like independent tasks) and the *non blocking ones* (shown on the right column) that can happen at the same time interval (like reminders).
- An intuitive GUI for user interaction (See Fig. 2) that allows introducing, checking, modifying and removing appointments as well as configuring the visualization properties (number of days, size, time intervals).
- The complete set of appointment configurations can be saved/loaded in a standard file format.
- Appointment booking negotiation that allows to reserve a set of intervals and confirm or refuse them later.
- A set of methods that allow searching the best free interval where an appointment fits, or retrieving the list of appointments contained in a given time range.

This plug-in has been included in both personal and doctor agents in order to manage their schedule during the booking negotiation. A detailed explanation of this process is made in §3.3.

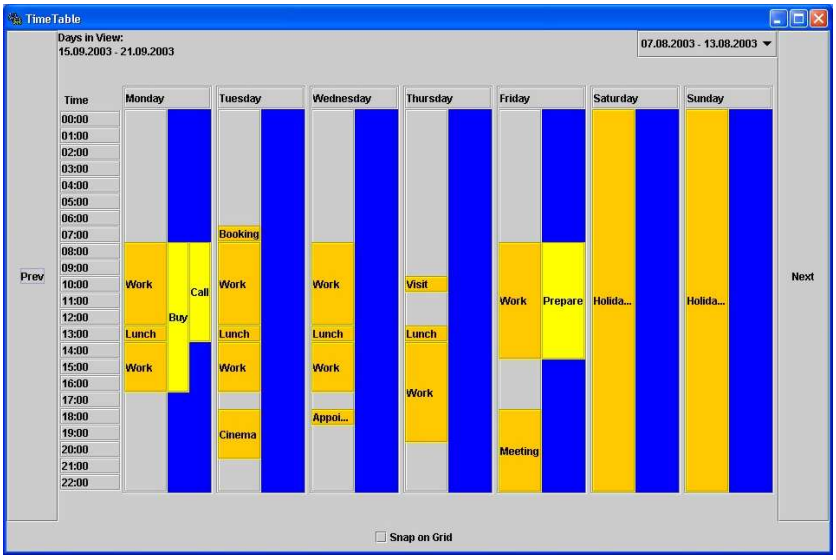


Fig. 2. Timetable plug-in GUI

3.2 Preferences plug-in

The *preferences plug-in* is a package which is able to manage multiple domains of preference. Its main goal is to allow to the user to obtain a ranking of a list of options (alternatives) according to the preferences of its corresponding domain.

Each domain (e.g. health care) has a set of criteria that corresponds to different properties or elements in the domain (e.g. doctors, cities). Each criterion can take different linguistic values to define the set of alternatives (e.g. doctors: Dr. Smith, Dr. Winston; cities: Tarragona, Barcelona). The user must express his preferences on these values. To do this, the user assigns a preference linguistic label to each of the possible values of the criterion (e.g. Tarragona=Best, Montblanc=OK, see Fig. 4). In addition, the user can indicate that one of the possible values of the criteria is not valid for him. For example, if the user does not want to be visited in a certain medical centre, he can assign a "Veto" label to this value (e.g. ConsultoriLocalAlforja=Veto).

Another feature of the plug-in is the possibility of giving different weights to each criterion. Moreover, some criteria can be temporally deactivated if the user decides not to take them into account for further decisions.

Once the user has specified his preferences, the plug-in can rank the alternatives taking into consideration all the criteria. To solve this problem a classical MCDM (Multi Criteria Decision Making, [11]) method has been implemented, which is based on multiattribute utility theory. This approach considers that each criterion is expressing a partial utility of each alternative. To aggregate these partial utilities, a weighted average is computed. Since the preferences are

defined using linguistic values, a pre-processing stage is performed in which labels are translated into numbers. The result is a sorted and rated list of alternatives.

The plug-in has been integrated in the Personal Agent to sort the list of received proposals. They are sorted according to some criteria (e.g: place of the treatment, the name of the doctor, the medical centre) after a query for bookings has been performed.

As the user cannot know in advance all the possible values of the criteria (e.g. which will be the names of the doctors of that city or which are the medical centres), the plug-in is able to discover new values each time that a query is performed (See Fig. 3). After each query, the plug-in adds the new values that appears in the proposals received. After that, the user could be able to rate these new criterions's values that will be taken into consideration in a future negotiation. In Fig. 4 we can see the names of the doctors, medical centres and cities that have made proposals to us. The labels beside them indicate our preference degree.

With this functionality the agent is able to learn from the user interaction and improve the quality of the rankings.

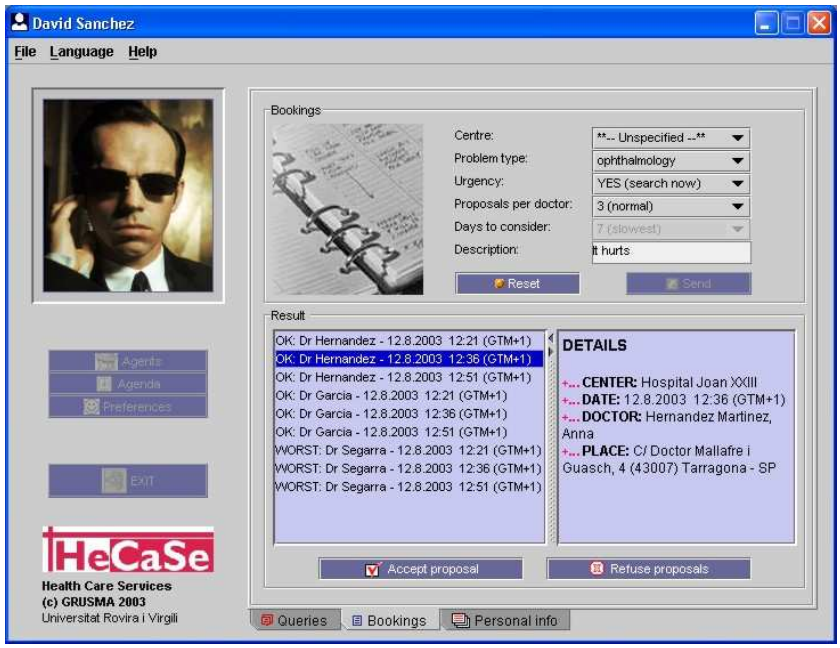


Fig. 3. Graphical interface to book a visit to a doctor

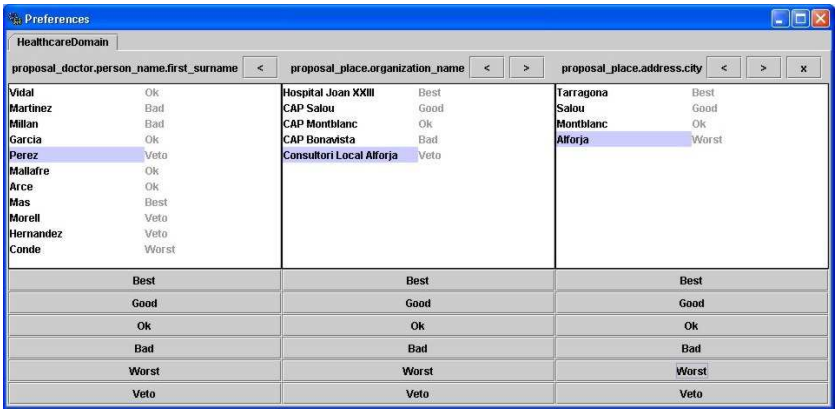


Fig. 4. Preferences plug-in GUI

3.3 Booking negotiation

When the user needs the care of a doctor, he can book a visit using HeCaSe. The booking process requires the negotiation of many agents. So, this is the most complex process in the system due to its high level of customisation according to the user's profile and the number of agents that interact.

The protocol (see Fig. 5) is divided in several steps on which the data is processed from the most general to the most specific agent in order to achieve the best proposal that fits with the user's preferences:

1) Firstly, the user initialises a set of search constraints (medical specialty, urgency, number of desired proposals per doctor, number of days to consider and, optionally, the desired medical centre). Then the system analyses the user's timetable through the Timetable plug-in in order to find the best time intervals in which an appointment with a doctor could be set. To do this, it divides the following X days (the number has been given by the patient),into three intervals that are sorted according to the amount of free minutes inside them, preserving their natural order (position in time) in case of a tie. This represents the patient's wish to get an appointment in an interval that is as soon as possible and not already full with other appointments. With all this information, the Personal Agent (PA) constructs a message.
2) Secondly, the message is sent to the Broker Agent (BA) that broadcasts it to all Medical Centre Agents (MCA) or to the one selected by the user. The MCA fowards the message to the Department Agent (DEP) that fits with the selected medical speciality and finally, the DEP broadcasts it to all its Doctor Agents (DA).
3) Each DA evaluates the booking preconditions included on the received message and constructs a list of proposal appointments which are exclusive to that patient until the protocol is finished (to avoid concurrency problems). The number of maximum proposals build by each DA is given by user in the

booking form (if there are enough free time in the doctor's agenda). In particular, the agent tries to fulfil the patient's wishes by matching the doctor's free time (determined by the timetable) with the sorted preferred intervals contained on the received message. If there is no possible slot in the most preferred interval or the user has requested more than one proposal, the doctor repeats the matching process with the next interval. The final proposed hours are reserved until the doctor agent receives the final accept or refuse message from the patient. This list of proposals of each doctor is returned to the DEP, which puts together all the doctors' proposals and sends them to the MCA. Then, all the MCAs involved on the protocol send their lists to the BA that joins them all and sends them to the user's PA.

4) Once this list is received, the PA evaluates the user's preferences (on the medical centres, cities and doctors). The Preferences Plug-in sorts and rates the proposals (as described in §3.2) and the result is shown to the user. If some of the proposed appointments overlap with the appointments in the patient's timetable, the PA gives him a warning.
   The maximum number of proposals received depends on the number of doctors found and the number of proposals per doctor that we have specified. The list of proposals could be empty if no doctors can find a free slot. In this case the user can change the parameters of the booking (e.g. selecting a different city) and repeat the process again (See Fig. 3).
5) Finally, the user can select his preferred proposal or reject all of them. In the first case, the appointment is fixed on his timetable and an Accept message is sent through the medical hierarchy of agents to the desired doctor, and at the same time a Refuse one is broadcasted to the rest of involved agents; finally a confirmation is returned to the PA. In the second case, all agents receive a Refuse message.

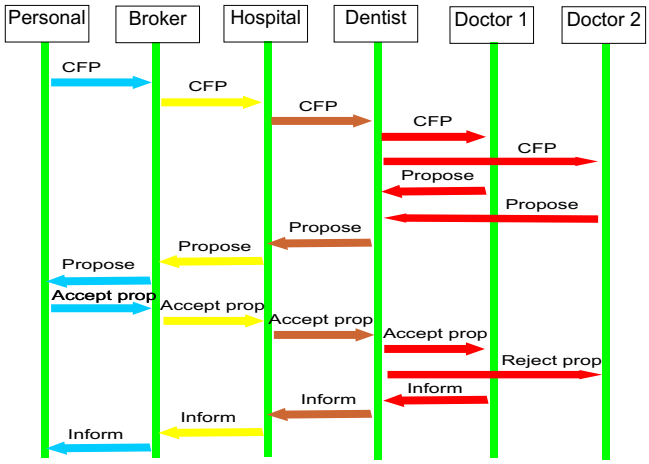


Fig. 5. 4-layer CFP Protocol for booking a visit

4 Conclusions and future work

HeCaSe is part of the Agent Cities initiative ([1]) whose main aim is to provide agent-based services that improve the quality of life of the citizens and the visitors of a city. The deployed application certainly contributes towards this objective, by giving to the users the possibility of accessing medical data in an easy and efficient way.

This system is fully implemented in a prototype version, with all the agents running in standard PCs. The system has been designed to benefit two kinds of users:

a) Citizens that need medical services, who could access their medical record from anywhere, make appointments with a certain doctor based on his preferences, and obtain any kind of information related to the medical centres, departments and doctors of a given city.
b) Health care personnel, who could automatically access and update the patient's medical record during the examination, and would have their workload decreased, due to the fact that citizens could make queries or appointments on their own.

The system described[1] in this paper has been implemented using JADE [10]. The content of all messages is written in RDF ([4]). The medical records database has been implemented using MySql. Moreover, we use the plug-in JADE-S[2] and the cryptographical library Cryptonite ([5]) to provide a secure infrastructure in the platform [8].

Finally, we summarise some research lines to follow in the future:

- The test of the system with real data of medical centres, departments and doctors.
- The implementation of user agents deployed in mobile environments (such as PDAs or mobile phones). The LEAP library [10], compatible with JADE, may be used for this purpose.
- The integration of new agents that simulate other medical entities like Medical Service Agents (X-rays, clinical analysis ...).
- The use of medical guidelines to ease the doctors labour by proposing a set of actions or treatments to be followed for a medical problem.
- The design of a hierarchy of Brokers that controls a subset of medical entities (e.g: located in the same city). We will study the coordination of these brokers in order to provide a more efficient solution instead of the centralised Broker approach.

[1] Our system is accessible live in the Agent Cities network of platforms (see access instructions at http://grusma.etse.urv.es/hecase/)
[2] JADE Security (JADE-S) is a plug-in and it can be obtained from the JADE's Website (http://sharon.cselt.it/projects/jade)

Acknowledgements

This system has been developed with the support of AgentCities.NET through the deployment grant "Deployment of agent-based health care services" and a Student Mobility Grant. In relation to this grant, we acknowledge Marek Jawurek from University of Aachen (Germany) for his contributions on the system. The authors also acknowledge the support of the Spanish thematic network "Creación de un entorno innovador para la comunicación de agentes inteligentes" (MCyT, TIC2001-5108-E).

References

1. Agentcities. IST project IST-2000-28384 Agentcities, 2000. http://www.agentcities.net/.
2. Generalitat de Catalunya. Law on the information rights concerning health, the autonomy of the patient and the clinical documentation. Law 21/2000, 29th December 2000. In *Quaderns de Legislació, 31*, pages 1–35. Health and Social Security Department, Generalitat de Catalunya, 2001. ISBN 84-393-5459-2.
3. Generalitat de Catalunya. Servei Català de la Salut (ICS), 2003. http://www.gencat.net/scs.
4. W3C Consortium. Resource Description Framework (RDF) - W3C Semantic Web Activity, 2003. http://www.w3.org/RDF/.
5. Cryptonite. Package of Java logi.crypto, 2000. More information available on http://logi.org/logi.crypto.
6. FIPA. FIPA Specification, 2003. http://www.fipa.org.
7. A. Moreno, D. Isern, and D. Sánchez. Provision of agent-based heath care services. *AI Communications. Special Issue on Agents in Healthcare*, 16:135–218, 2003.
8. A. Moreno, D. Sánchez, and D. Isern. Security Measures in a Medical Multi-Agent System. In *Sisè Congrés Català d'Intel.ligència Artificial (CCIA'03)*, Palma de Mallorca, Illes Balears, October 23-26, 2003 (in press).
9. A. Moreno, A. Valls, D. Isern, and D. Sánchez. GRUSMA1: Experience on the deployment of agent-based health care services. In *Applications of Software Agent Technology in the Health Care Domain*. To appear in WhiteStein series in Software Agent Technologies, Zurich, Switzerland, 2003 (in press).
10. TILab S.p.A. Java Agent Development Framework (JADE) v.3.01beta, March 2003. More information available on http://sharon.cselt.it/projects/jade.
11. P. Vincke. *Multicriteria Decision-Aid*. John Willey and Sons, 1992. ISBN: 0471931845.
12. M. Wooldridge. *An introduction to Multi-Agent Systems*. John Wiley and Sons, 2002. ISBN 047149691X.

Towards holonic multiagent systems: Ontology for control tool boxes

Silvia Suárez, Beatriz López, Joaquim Melendez
Universitat de Girona
Av. Lluís Santaló s/n
17071 Girona, Spain
{sasuarez, blopez, quimmel}@eia.udg.es

Abstract

During years of research in control field, some supervision, detection and diagnosis techniques have been developed. Now, advances on distributed and ubiquitous computing, networking and sensors provide new environments to integrate control techniques. However, this integration is not straigforward, reason of why new methods and approaches are required. In this sense, holons and agents seems a promising parading in which future control systems can be developed. The work presented in this paper is framed on this new paradigm. Particularly, we propose an ontology for dealing with control tool boxes integration. The ontology has been elaborated with Protegé2000.

1. Introduction

During the years of research in control field, a myriad of supervision, detection and diagnosis techniques have been developed. Now, advances on distributed and ubiquitous computing, networking and sensors provide new environments in which it is possible to integrate all such techniques. As a consequence, it will be possible to decide at any moment which technique is appropriate to the problem at hand, by taking into account all the possible sources of information.

However, the integration of control techniques rises on the main challenge of shifting from traditional control systems as processes with single controllers to control systems as collections of heterogeneous physical and information systems, with complex inter-connexions and interactions [1]. New methods and approaches are then required, and holons and agents, that have follow parallel research trends, have recently joined.

On one hand, a holon is “an autonomous and co-operative building block of a manufacturing system for transforming, transporting, storing physical and information objects [2], [3]. It consists on a control part and an optional physical processing part (see figure 1). A holon can itself also consist of other holons [3].

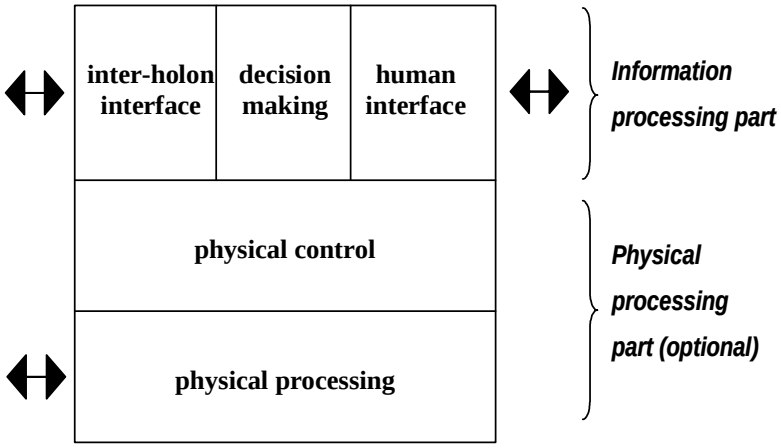


Figure 1. General architecture of a holon (from [3]).

On the other hand, agents provide autonomy with respect to the system capacity to a given environment [4]. Agent Technology, although broadly extended on open applications as Internet services, has been recently introduced on the control field. Oto Mayr, in his interview with S. Bennet [5], tells that "agency" is a concept that is present in all industrial, social and economic systems. Mayr defines "agency" as the acting capacity based on knowledge and information. From the control perspective, an agent is a feedback process that pursues certain goals; sets of learning agents (multi-agent systems) use feedback processes to modify their knowledge. Recently, Jennings and Bussmann analyse the suitable of agent-technology to engineering complex systems [6]. Moreover, they have several works related to methodological approaches to use agents in production control systems [7], [8].

Although both approaches, holons and agents, share many basic concepts, research in each area has been developed independently for the most part: holonic systems research has focused on manufacturing systems, and Agent Technology research on the development of interconnected systems in which data, control, expertise or resources are distributed. Being aware of the common interest, there are recent efforts to joint both communities [9], [10], [11], [12], [13]. This has lead to form the term Holonic Multiagent Systems, a novel paradigm for managing, modelling and supporting complex systems [9], [13]. This new paradigm provides two main benefits. On one hand, holons provides soundness and robustness, typical characteristics of the system engineering developments. On the hand, agents facilitates the integration of heterogeneous systems.

Our work is in line with this new approach to integrate heterogeneous control systems. More that building new control techniques, we focus on the integration of them. As a first step, we have participated in the development of tool boxes that encapsulates and describes control techniques [14]. Now, we are dealing with the problem of making tool boxes interconnection operational. In such challenge, one of the main drawbacks consists on information sharing, and so, ontologies play an essential role. Ontologies have established themselves as a powerful tool to enable knowledge sharing, and achieve semantic interoperability among heterogeneous, distributed agent systems [15]. Our ultimate goal is to use the ontology to integrate control tool boxes in a holonic multiagent system.

This paper is organized as follows: First, in section 2, we detail some related work. Then, in section 3, we provide our ontology. We end in section 4 by giving some conclusions.

2. Related works

There are some previous works on the definition of ontologies for diagnosis and fault detection. In [16], an ontology for fault diagnosis in Electrical Networks is presented. This paper remarks the clear need of a standard representation of the electrical network. In the past, each system has been build using a different representation of the network. However, the domain knowledge was the same in all cases. The works in this area are aimed at the representation of this knowledge in a way that is reusable by several applications. Here, the authors say that the obvious way to identify what is reusable and how to represent it is to look at the domain knowledge required in applications where the knowledge about the electrical network is crucial, e.g. fault diagnosis. The long term objective is to obtain a standard representation for the applications and electrical utilities. This paper is similar to our approach, since the authors aim's is to develop a standard ontological study. However, they are constrained to a domain (electrical), while we are concerned on control techniques, whatever domain it is.

In addition, in [17] an ontological analysis of fault process is presented in a general way. This study ontologically analyzes the fault process, i.e., causal chains in which the faults and the symptoms are induced, aiming at articulation of concepts which can categorize the fault processes and the faults. The ontology of faults provides a conceptual vocabulary to explicate the scope of a diagnostic activity performed by a reasoning mechanism using a kind of model. The objective of this paper is to identify richer vocabulary including such concepts from the viewpoints not of logic but of ontology of physical systems. Some questions are proposed by the authors, i.e.: How the observed symptom is induced by the initial fault, what type of phenomena appear in the fault process, and what categories of the deeper causes of faults exist. Our approach is broader than [17], due to the fact that our ultimate goal is to build an ontology for all the steps involved in control systems: supervision, fault detection and diagnosis.

On the other hand, in [18] a representation of fault cases for supporting fault diagnosis tasks is explained. The authors use an ontology for Representation of fault case information. The specified terms in this approach are limited to fault cases. This approach is different to our approach but it helps to observe and know the components of the fault cases presented.

3. Elaborated Ontology

In order to elaborate an ontology for control tasks, we have used the terms proposed in [19] by the area of supervision, fault detection and diagnosis. Therefore, main terms are organized in signals, states, functions, models and system properties (see Figure 2). Each term is defined in properties and relations, generating a complex network of classes, subclasses, instances and slots. Following we describe all the terms.

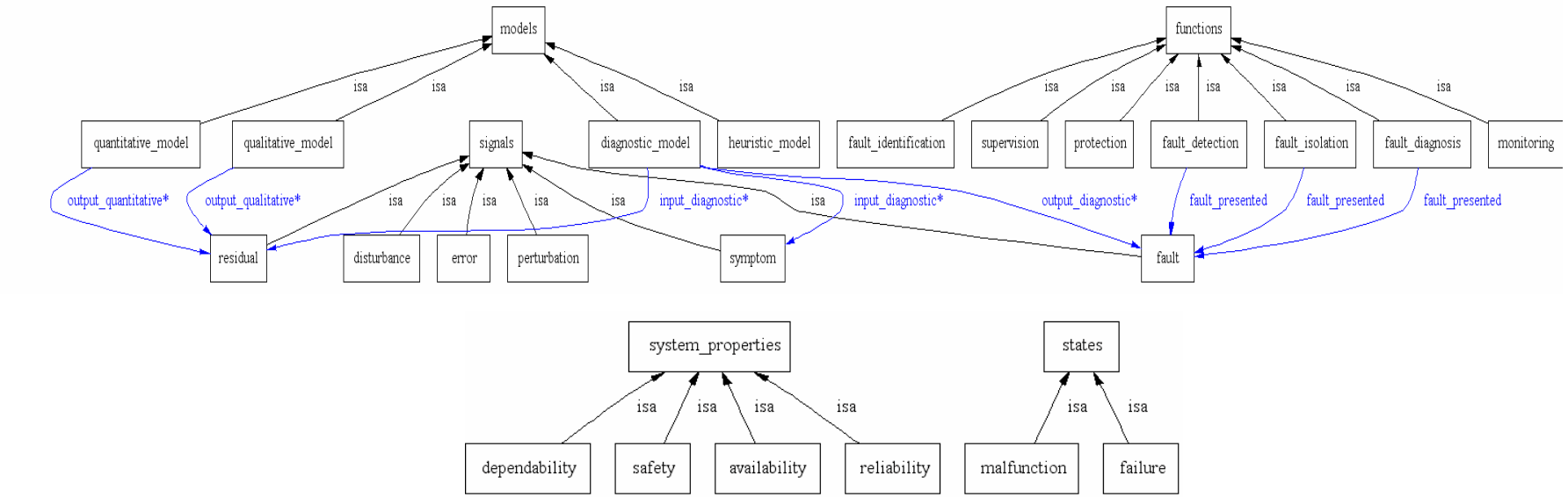


Figure 2. ontology general diagram

Signals

Signals are defined as physical measures or perceptible variables that communicate information and messages. There are six classes of signals:

- Fault: Unpermitted deviation of at least one characteristic property or parameter of the system from acceptable / usual / standard condition. It is composed of the following attributes: cause, duration, final_time, kind_fault, location_fault, name_fault, parameter, result, starting_time.
- Error: Deviation between a measured or computed value (of an output variable) and the true, specified or theoretically correct value. It presents the following attributes: cause, correct_value, duration, final_time, measured_computed_value, result, starting_time
- Disturbance: An unknown (and uncontrolled) input acting on a system. The attributes of this subclass are: cause, duration final_time, input_acting, result and starting_time.
- Perturbation: An input acting on a system which results in a temporary departure from current state. The attributes are: cause, duration, final_time, input_acting, result, starting_time
- Residual: Fault indicator, based on deviation between measurements and model-equation-based computations. It has the following attributes: cause, deviation, duration, final_time, result, starting_time.
- Symptom: Change of an observable quantity from normal behaviour. The attributes of this subclass are: cause, duration, final_time, result, starting_time.

States

States are the smallest set of information required to know the system situation. There two main states: failures, malfunction.

- Failure: Permanent interruption of a systems ability to perform a required function under specified operating conditions. The attributes of this subclass are: starting_time and state.
- Malfunction: Intermittent irregularity in fulfilment of a systems desired function. It presents the following attributes: period and state.

Functions

Functions are the different kind of operations that can be performed in order to control a given system. There are seven main kinds of functions:

- Fault detection: Determination of faults present in a system and time of detection. This subclass is composed of the following attributes: fault_presented and time_detection.
- Fault isolation: Determination of kind, location and time of detection of a fault. Follows fault detection. The attributes of this subclass are: fault_presented, kind_fault, time_detection.
- Fault identification: Determination of the size and timevariant behaviour of a fault. Follows fault isolation. It presents the attributes: size_fault and timevariant_behaviour.
- Fault diagnosis: Determination of kind, size, location and time of detection of a fault. Follows fault detection. Includes fault isolation and identification. The attributes of this subclass are: fault_presented, kind_fault, location_fault, size_fault and time_detection.
- Monitoring: A continuous real time task of determining the conditions of a physical system, by recording information recognising and indicating anomalies of the behaviour. The attributes are: anomalies_behaviour and information_recongnising.
- Supervision: Monitoring a physical system and taking appropriate actions to maintain the operation in the case of faults. It has the following attributes: actions_mantain_operation.
- Protection: Means by which a potentially dangerous behaviour of the system is suppressed if possible, or means by which the consequences of a dangerous behaviour are avoided

Models

For the purpose of engineering analysis and design, physical systems are usually represented in some mathematical form; this representation is also called the model of the system. The properties of the model reflect the nature of the system, though in many cases the model may just be an approximation of the true system behaviour. In [19], four classes of models are considered (see figure 3):

- Quantitative model: Use of static and dynamic relations among system variables and parameters in order to describe systems behaviour in quantitative mathematical terms. The attributes of this subclass are: description, input_quantitative, output_quantitative and parameters.
- Qualitative model: Use of static and dynamic relations among system variables and parameters in order to describe systems behaviour in qualitative terms such as causalities or if-then rules. It presents the following attributes: description, input_qualitative, output_qualitative and parameters.
- Diagnostic model: A set of static or dynamic relations which link specific input variables - the symptoms - to specific output variables - the faults. The attributes of this subclass are: description, input_diagnostic, output_diagnostic and parameters.
- Heuristic_model: The attributes of this subclass are: description, input_heuristic and output_heuristic.

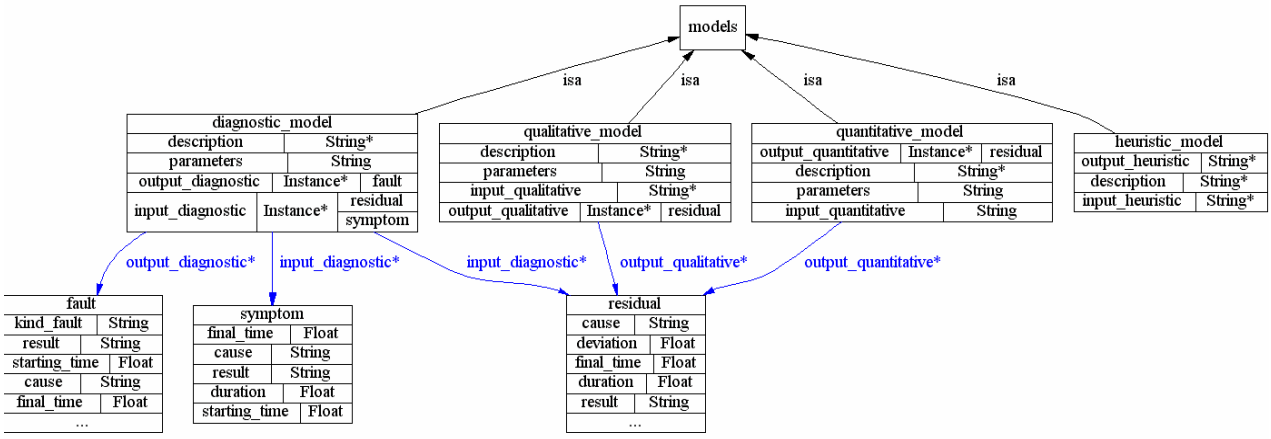


Figure 3. Ontology particular diagram.

System Properties

Systems properties relate particular characteristics of required for the system. Main properties are: reliability, safety, availability, dependability.

- Reliability: Ability of a system to perform a required function under stated conditions, within a given scope, during a given period of time. Measure: MTBF = Mean Time Between Failure. MTBF = 1\la; la is rate of failure [e.g. failures per year]. It has the following attributes: MTBF, period_time and required_funtion.
- Safety: Ability of a system to not cause a danger for persons, equipment or environment. It presents de following attributes: value_safety.
- Availability: Probability that a system or equipment will operate satisfactorily and effectively at any point of time measure: MTTR: Mean Time To Repair MTTR = 1/μ; μ: rate of repair. The attributes of this class are: MTTR and probability_availability.
- Dependability: A form of availability that has the property of always being available when required. It is the degree of which an item is operable and capable of performing its required function at any randomly chosen time during its specified operating time, provided that the item is available at the start of that period [RAM Dictionary]. It presents the following attributes: degree_dependability and time_dependability.

The ontology has been created using the tool protégé 2000 [20], (see figure 4).

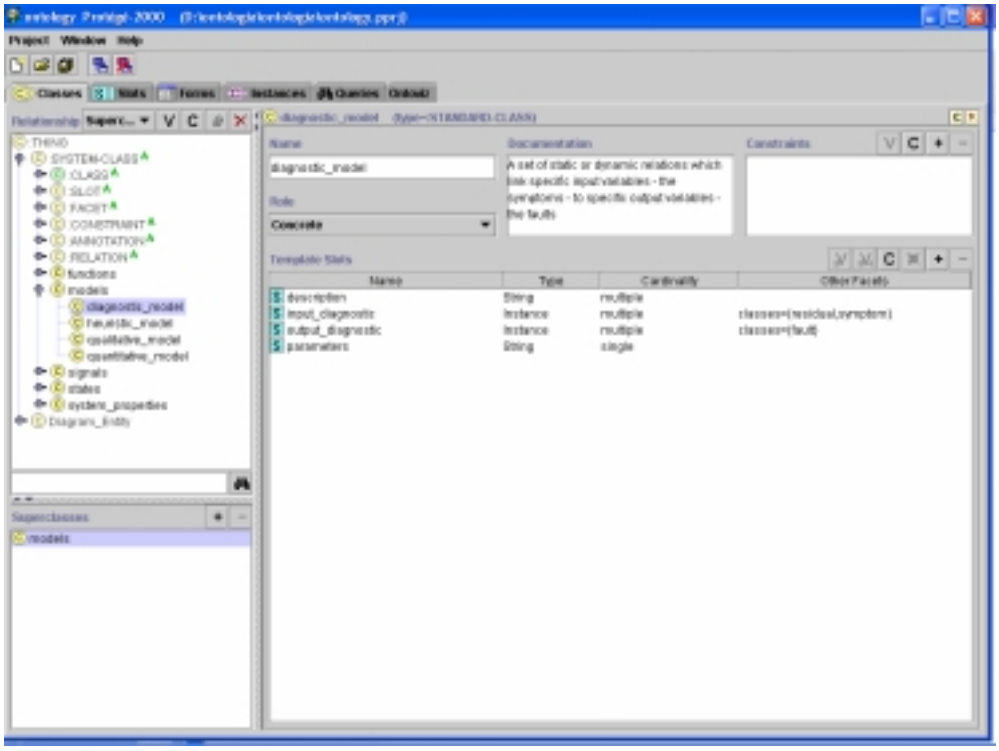


Figure 4. Screen of Protegé

4. Conclusions

In this paper we propose an ontology as first step to solve the problem of making operational the integration of several control techniques, so achieving an holistic control system. We have implemented a preliminary version of the ontology on Protege2000. Our next step involves the use of the ontology in a holonic multiagent system, in which several control agents (agentified tool boxes) interact.

References

[1] Murray, R.M., Aström, K.J., Boyd, S.P., Brockett, R.W., Stein, G. Future Directions in Control in an Information-Rich World. A summary of the report of the Panel on Future Directions in Control, Dynamics, and Systems. IEEE Control Systems Magazine, April 2003, pp. 20-33.

[2] Christensen, J. Holonic Manufacturing Systems – Initial Architecture and Standards Directions Proc. First European Conference on Holonic Manuyfacturing Systems, Hannover, Germany, 1994.

[3] MacFarlane, D., Bussmann, S. Developments in holonic production planning and control. Int. Journal of Production Planning and Control, 11 (6): 522-536, 2000.

[4] Weiss G. (Editor) "Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence". The MIT Press, 1999.

[5] Bennett, S. Otto Mayr: Contributions to the History of Feedback Control. IEEE Control Systems Magazine, 22 (2): 29-33, April 2002.

[6] Jennings, N.R., Bussmann, S. Agent-Based Control-Systems. Why They Suited to Engineering Complex Systems. IEEE Control Systems Magazine, June 2003, pp. 61-73

[7] Bussmann, S., Jennings, N.R., Wooldridge, M. On the Identification of Agents in the Design of Production Control Systems, In: P. Ciancarini, M.J. Wooldridge (eds.), Agent-Oriented Software Engineering, LNCS 1957, pp. 141-162. Springer-Verlag: Berlin, Germany, 2001.

[8] Bussmann, S., Schild, K. An Agent-based Approach to the Control of Flexible Production Systems Proc. 8th IEEE Int. Conf. on Emerging Technologies and Factory Automation (EFTA 2001), pp. 169-174. Antibes Juan-les-pins, France, 2001.

[9] HoloMAS 2003, 1st International Conference on Industrial Applications of Holonic and Multi-Agent Systems, September 1-3, 2003, Prague, Czech Republic.
http://gerstner.felk.cvut.cz/HoloMAS/2003

[10] Workshop on Holons: Autonomous and Cooperative Agents for Industry, at the 5th International Conference on Autonomous Agents, Montreal, Canada, 29 May 2001.
http://isg.enme.ucalgary.ca/workshop.htm

[11] Brennan, R.W.; Fletcher, M.; Norrie, D.H. An agent-based approach to reconfiguration of real-time distributed control systems. Robotics and Automation, IEEE Transactions on , Volume: 18 Issue: 4 , Aug. 2002. Page(s): 444 -451

[12] Fletcher, M.; Brennan, R.W.; Norrie, D.H.; Deadline control in holonic manufacturing using mobile agents. Database and Expert Systems Applications, 2001. Proceedings. 12th International Workshop on , 3-7 Sept. 2001. Page(s): 623 -627

[13] Schillo, M., Fischer, K. Holonic Multiagent Systems. In: Zeitschrift für kÑunstline, n. 3 (to appear).

[14] CHEM project. Advanced Decision Support System for Chemical/Petrochemical Manufacturing Processes. http://www.chem-dss.org/

[15] Working group on Ontologies. Agenticies.net.
(http://www.csc.liv.ac.uk/~valli/WG-Ontology/index.html)

[16] Bernaras A., Laregoiti I., Bartolomé N. And Corera J. An Ontology for Fault Diagnosis in Electrical Networks. Intelligent Systems Applications to Power Systems, 1996. Proceedings, ISAP '96., International Conference on , 28 Jan-2 Feb, 1996.

[17] Kitamura Yoshinobu and Mizoguchi Riichiro, An Ontological Analysis of Fault Process and Category of Faults. Proc. Of Tenth Interantional Workshop on Principles of Diagnosis (DX-99), pp.118-128, June 8-11, 1999.

[18] Kato, Yoshikiyo, Shirakawa Takahiro and Hori Koichi. Utilizing Fault Cases for Supporting Fault Diagnosis Tasks.

[19] Terminology in the field of supervision, fault detection and diagnosis.
http://w3.rt.e-technik.tu-darmstadt.de/institut/terminology.en.html

[20] Protegé 2000. http://protege.stanford.edu/

Ontological Issues in Agent-aware Negotiation Services

Andrea Giovanucci[1] and Juan A. Rodríguez-Aguilar[1,2]

[1] iSOCOLab
Intelligent Software Components, S. A.
Edificio Testa, C/ Alcalde Barnils, 64-68 A
08190 Sant Cugat del Vallès, Barcelona, Spain
{andrea,jar}@isoco.com
http://www.isoco.com
[2] Artificial Intelligence Research Institute, IIIA
Spanish Council for Scientific Research, CSIC
08193 Bellaterra, Barcelona, Spain.
jar@iiia.csic.es
http://www.iiia.csic.es

Abstract. In negotiation events involving multiple, highly customisable goods, buying agents need to express relations, constraints, and preferences over attributes of different items. Not forgetting the provider side, providing agents may also impose constraints or conditions over their offers. Thus, there is the need for providing all trading agents involved with a rich enough language to express their business rules. In this paper we focus on the ontological issues that need to be considered in order to empower the expressiveness offered by negotiation objects and offers to incorporate buyers' and providers' business preferences. We take the stance that a highly expressive ontology is compulsory to enact agent-aware negotiation services in actual-world procurement settings.

1 Introduction

Although negotiation is a key procurement mechanism, to the best of our knowledge most agent-based services deployed so far have primarily focused on infrastructure issues related to negotiation protocols. Furthermore, no special attention has been paid to developing negotiation ontologies that provide the expressiveness required in actual-world settings. Thus, the lack of agent-based decision support for trading agents that help improve current trading practices hinders the adoption of agent technology in procurement scenarios.

Consider the problem faced by a buying agent when negotiating with providing agents. In a negotiation event involving multiple, highly customisable goods, buying agents need to express relations and constraints between attributes of different items. On the other hand, it is common practice to buy different quantities of the very same product from different providing agents (multiple sourcing), either for safety reasons or because offer aggregation is needed to cope

with high-volume demands. This introduces the need to express business constraints on the number of providing agents and the amount of business assigned to each of them. Not forgetting the provider side, providing agents may also impose constraints or conditions over their offers. Offers may be only valid if certain configurable attributes (f.i. quantity bought, delivery days) fall within some minimum/maximum values, and assembly or packing constraints need to be considered. Once the buying agent collects offers, he is faced with the burden of determining the winning offers. Furthermore, there is the need for providing all trading agents involved with a rich enough language to express their business preferences.

In [5] we introduced *iBundler* an agent-aware decision support service that relieves buying agents from determining the winning offers based on the formal model thoroughly described in [6]. In this paper we primarily focus on the ontological issues that need to be considered in order to empower the expressiveness offered by negotiation objects and offers to incorporate buyers' and providers' business preferences. At such aim, our approach required the extension of state-of-the-art ontologies for automated negotiation[7, 8].

The paper is organised as follows. Section 2 introduces the market scenario where buyers and providers are to negotiate, along with the expressiveness requirements they may need. Next, a slight description of *iBundler*, an agent-aware decision support service for combinatorial negotiations, is provided in section **??**. Thereafter, the ontology required to enact the service is thoroughly described in 4. Finally, section 5 summarises our contributions and proposes future research lines.

2 Market scenario

Next we detail the capabilities required by buyers and providers in an actual-world procurement scenario. The requirements below are intended to capture buyers' constraints and preferences and outline a highly expressive bidding language for providing agents:

Negotiate over multiple items. A negotiation event is usually started with the preparation of a request for proposal (RFQ) form. The RFQ form describes in detail the requirements (including attribute-values such as volume, quality specifications, dates as well as drawings and technical documentation) for the list of items (goods or services) defined by the negotiation event.

Offer aggregation. A specific item of the RFQ can be acquired from several providers simultaneously, either because not a single provider can provide with the requested quantity at requested conditions or because buyers explicit constraints (see below).

Business sharing constraints. Buyers might be interested to restrict the number of providers that will finally trade for a specific item of the RFQ, either for security or strategical reasons. It is also of usual practice to define the minimum amount of business that a provider may gain per item.

Constraints over single items. Every single item within an RFQ is described by a list of negotiable attributes. Since: a) there exists a degree of flexibility in specifying each of these attributes (i.e. several values are acceptable) and b) multiple offers referring the very same item can be finally accepted; buyers need to impose constraints over attribute values. An example of this can be the following: suppose that the deadline for the reception of certain item A is two weeks time. However, although items may arrive any day within two weeks, once the first units arrive, the rest of units might be required to arrive in no more than two days after.

Constraints over multiple items. In daily industrial procurement, it is common that accepting certain configuration for one item affects the configuration of a different item, for example, when dealing with product compatibilities. Also, buyers need to express constraints and relationship between attributes of different items of the RFQ.

Specification of providers' capacities. Buyers cannot risk to award contracts to providers whose production/servicing capabilities prevent them to deliver over-committed offers. At this aim, they must require to have providers' capacities per item declared. Analogously, next we detail the expressiveness of the bidding language required by providers. The features of the language below are intended to capture providing agents' constraints and preferences.

Multiple bids over each item. Providers might be interested in offering alternate conditions/configurations for a same good, i.e., offering alternatives for a same request. A common situation is to offer volume-based discounts. This means that a provider submits several offers and each offer only applies for a minimum (maximum) number of units.

Combinatorial offers. Economy efficiency is enhanced if providers are allowed to offer (bid on) combination of goods. They might lower the price, or improve service assets if they achieve to get more business.

Multi-unit offering. Each provider needs to specify that they will only participate in trading if a minimum (maximum) amount of business is assigned to him.

Homogeneous combinatorial offers. Combinatorial offering may produce inefficiencies when combined with multi-unit offering. Thus a provider may wind up with an award of a small number of units for a certain item, and a large number of units for a different item, being both part of the very same offer (e.g. 10 chairs and 200 tables). It is desirable for providers to be able to specify homogeneity with respect to the number of units for complementary items.

Packing constraints. Packing units are also a constraint, in the sense that it is not possible to serve an arbitrary number of units (e.g. a provider cannot sell 27 units to a buyer because.his items come in 25-unit packages). Thus providers require to be capable of specifying the size of packing units.

Complementary and exclusive offers. Providers usually submit XOR bids, i.e., exclusive offers that cannot be simultaneously accepted. Also, there may exist the need that an offer is selected only if another offer is also selected. We refer to this situation as an AND bid. This type of bids allows to express volume-

based discounts. For example, when pricing is expressed as a combination of base price and volume-based price (e.g. first 1000 units at $2.5 p.u. and then $2 each).

Obviously, many more constraints regarding pricing and quantity can be considered here. But we believe these faithfully address the nature of the problem.

3 Agent-aware negotiation support service

The *iBundler* service has been implemented as an agency composed of agents and software components that cooperatively interact to offer a decision-support service for negotiation scenarios. *iBundler* can act as a combinatorial negotiation solver for both multi-item, multi-unit negotiations and auctions. Thus, the service can be employed by both negotiating agents and auctioneers in combinatorial auctions.

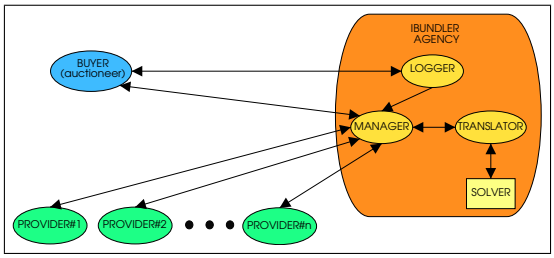


Fig. 1. Architecture of the *iBundler* agency

Figure 1 depicts the components of the agency, along with the fundamental connections of buying and providing agents with the service. Next we make explicit the main functionality of its members:

[**Logger agent**]**.** It represents the interface of the *iBundler* agency to the world. The Logger agent is in charge of facilitating registration and unregistration with the *iBundler* service to users (both buyers and providers) as well as their subsequent access to the service via log in and log out procedures.

[**Manager agent**]**.** Agent devoted to providing the solution of the problem of choosing the set of bids that best matches a user's requirements. There exists a single Manager agent per user (buying agent or auctioneer), created by the Logger agent, offering the following services: brokering service to forward buying agents' requirements (RFQs) to selected providing agents capable of fulfilling them; collection of bids; winner determination in a combinatorial negotiation/auction; award of contracts on behalf of buying agents. Furthermore, the manager agent is also responsible for: bundling each RFQ and its bids into a negotiation problem in FIPA-compliant[3] format to be conveyed to the Translator agent; and to extract the solution to the negotiation problem handled back by the Translator agent. Observe that figure 1 shows the interplay of buying

and providing agents with the Manager as the sole access point to the *iBundler* agency.

[Translator agent]. It creates an XML document representing the negotiation problem in a format understandable by the Solver departing from the FIPA-compliant description received from the Manager. It also translates the solution returned by the Solver into an object of the ontology employed by user agents. It is the bridge between the language spoken by user agents and the language spoken by Solver.

[Solver component]. The *iBundler* component itself extended with the offering of an XML language for expressing offers, constraints, and requirements. The XML specification is parsed into an MIP formulation and solved using available MIP solvers as described in [5].

4 Ontology

Although research on automated negotiation in multi-agent systems has concentrated on the design of negotiation protocols and their associated strategies, ontological aspects of negotiation protocols have recently started to attract researchers' attention (see [7, 8] and the results of the ADMIT project [2]). In [7, 8] we find an ontological approach to automated negotiation founded on the following concepts: *negotiation protocol* (rules followed by participants during a negotiation process), *party* (participants, be them either human agents, software agents or even organisations of agents), process (way to reach an agreement on some issue by modifying negotiation attributes), (negotiation) *object*, *offer* (possible combination of values associated to the negotiation attributes which represent an expression of will), *negotiation rule* (set of rules that govern a specific negotiation protocol). Although satisfactory enough for most concepts, particularly as to negotiation protocols regarded as processes and rules, in this work we had to enrich the concepts of *offer* and *object* in order to accommodate the expressiveness required by the actual-world constraints described in section 2 for bids and RFQs respectively. To the best of our knowledge, no ontology defined in prior work allows us the expressiveness that buying and providing agents require. In other words, there is no adequate ontology for multi-item, multi-unit combinatorial reverse auctions with side constraints. Thus we had to define an *ad-hoc* ontology for the *iBundler* service.

The ontology has been defined with the aid of *Protege 2000* [4]. Furthermore, the conversion from ontological objects to Java classes is realised via the *beangenerator* Protege 2000 plug-in[9]. The automatically-generated Java classes fulfill with the JADE specification in [1].

Figures 2, 3, and 4 provide graphical representations (as shown by the Ontoviz Protégé plug-in) of the core concepts in the *iBundler* ontology, namely, and respectively, the *RFQ*, *ProviderResponse*, *Problem*, and *Solution* concepts.

The *RFQ* concept is employed by buying agents to express their requests for bids. Figure 2 shows that an RFQ is composed of a sequence of *Request* concepts, one per requested item. A sequence of global constraints (*GlobalConstraint*

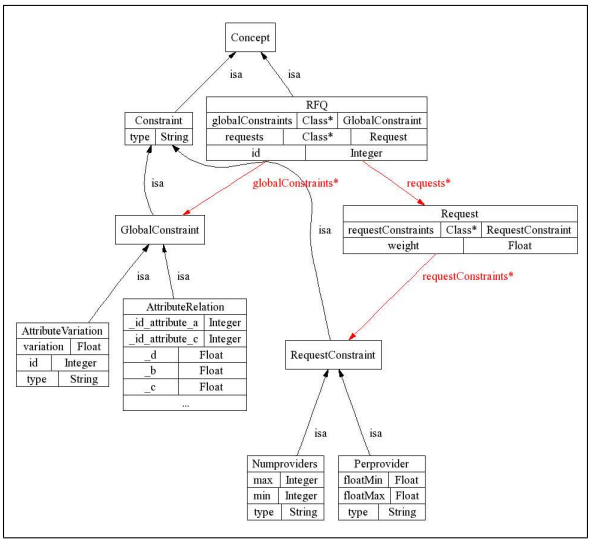


Fig. 2. RFQ concept representation

concept) relating separate, requested items may be part of an RFQ. There are two types of *GlobalConstraint* concepts: constraints that allow to express linear relationships between different attributes of the very same or separate item(s) (*AttributeRelation* concept) and constraints on the values of an item's attribute (AttributeVariation concept). A sequence of constraints on individual items (*RequestConstraint* concept) may be also part of an RFQ. Constraints on individual items can serve to limit the range of providers (*NumProviders* concept) to which the item can be awarded or the range of percentage of units to be awarded to the very same provider (*PerProvider* concept). Notice that all the constraints specified in an *RFQ* stand for the buyer's business rules.

On the provider side, providing agents express their offers in terms of the *ProviderResponse* concept, which in turn is composed of several elements: a list of *Bid* concepts (each *Bid* allows to express a bid per either a single requested item or a bundle of items); constraints on the production/servicing capabilities of the bidding provider (*Capacity* concept); and constraints on bundles of bids formulated with the *BidConstraint* concept (each *BidConstraint* in turn can be of exclusive –xor– or volume-based discount type –and–, corresponding respectively to the *XOR* and *AND* concepts). Whereas constraints on bundles of bids put into relation separate bids, constraints on individual bids (expressed as *SingleBidConstraint* concepts) allow to relate the values offered for separate items within the very same bid. As an example, homogeneity constraints can be declared by providers within some bid to make buyers aware that the quantity of items they can select per item must be the same, or else the provider will not

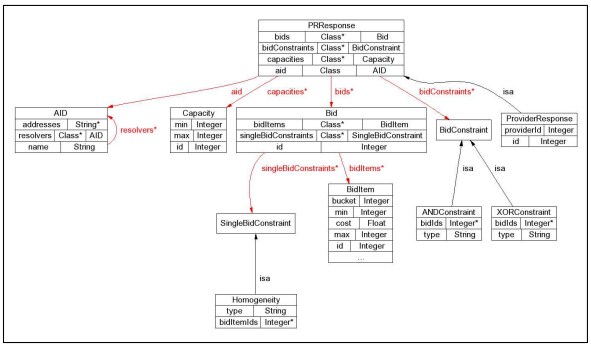


Fig. 3. Bid representation as part of the provider response concept

concede his bid. Such constraint maps to the *Homogeneity* concept, a particular type of *SingleBidConstraint*.

Once the manager collects all offers submitted by providers, he wraps up the *RFQ* concept as received from the buyer along with the offers as *ProviderResponse* concepts to compose the negotiation problem to be solved by the *Solver* component. The resulting concept, *Problem*, is depicted in figure 4.

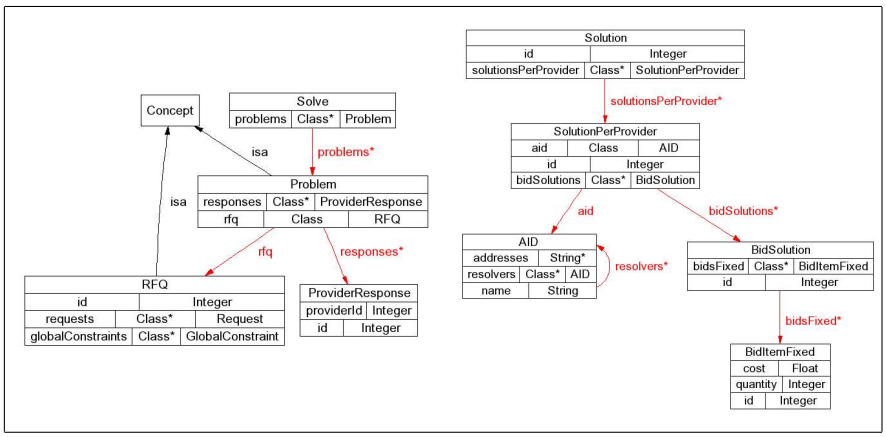


Fig. 4. Problem and Solution concepts

Finally, the solution produced by the Solver component is transformed by the translator agent into a *Solution* concept (see figure 4) that is handed over to the manager. The *Solution* concept contains the specification of the optimal set of offers calculated by Solver. Thus *Solution* contains a list of *SolutionPerProvider* concepts, each one containing the bids selected in the optimal bid set per provider, as a list of *BidSolution* concepts, along with the provider's agent

identifier, as an *AID* concept. Each *BidSolution* in turn is composed of a list of *BidItemFixed* concepts containing the number of units selected per bid along with the bid's total cost.

So far we have concentrated on ontological concepts referring to entities with a complex structure that can be defined in terms of slots. Hereafter we shall draw our attention to agent actions, i.e. the special concepts that indicate actions that can be performed by agents in the *iBundler* agency, as well as buyers and providers. Figure 5 depicts all available agent actions as descendants of a single common ancestor: *AgentAction*. Observe that the different agent actions correspond to the services offered by each agent.

Thus, the Logger agent offers the services associated to the following actions:

Login Action requested by trading agents when logging in with the *iBundler* agency.

Logout Action requested by trading agents when logging out of the *iBundler* agency.

Register Action requested by trading agents when signing for the *iBundler* agency. They must provide information about themselves. At the end of the registration, the Logger provides them with a user name and a password.

Unregister Action requested by trading agents when unregistering with the *iBundler* agency. They must provide information about themselves. All the brokering information associated to them is erased by the Logger.

As to the manager, it offers four core services via the following actions:

GetAllBids The buyer specifies an RFQ along with a list of providers. The manager agent forwards the query to all the providers and delivers back all the responses to the buyer.

Solve The buyer sends to the manager an *RFQ* along with a list of *ProviderResponse* concepts representing providers' offers. The manager composes a *Problem* out of the *RFQ* and *ProviderResponses* to subsequently request the translator agent for a *Solution*. In this way, the buyer is relieved from the intricate construction of a *Problem* involving the creation of crossed references between *RFQ* and the *Bid* concepts in each *ProviderResponse*. Once the *Solution* is received by the manager it is forwarded to the buyer.

Manage The buyer sends to the manager an *RFQ* along with a list of providers. The manager sends a filtered version of the *RFQ* (removing the buyer's private constraints) to available providers, collects all their offers, constructs a *Problem* to ask the translator for a *Solution*, which is conveyed to the buyer once calculated.

Buy The buyer constructs a *Solution* concept and subsequently asks the manager to the bids and providers in the list of *SolutionsPerProvider*. Notice that the buyer may employ the same *Solution* concept recommended by the manager as an optimal solution.

Finally, providers do offer their services through the following actions:

RequestForQuotations Request for offers received from the manager for a filtered version of the *RFQ* sent by the buyer.

BuySolution Order to buy selected offers received from the manager.

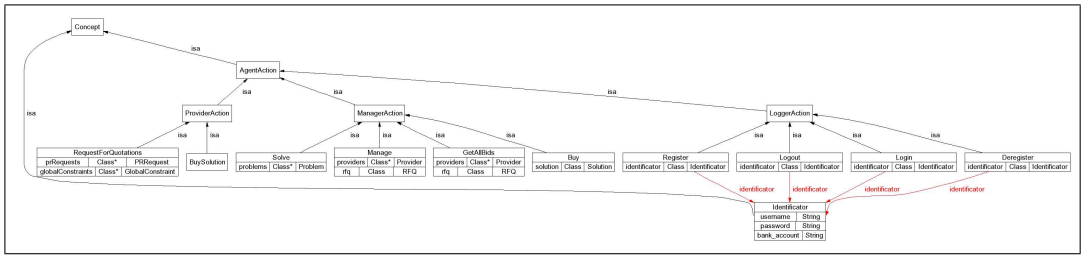


Fig. 5. Agent actions

5 Summary and Future Work

We believe that it is time to deploy agent services aimed at complex problem-solving so that they can be subsequently employed by other agents either to help them team up and cooperatively solve complex problems or to behave more efficiently in competitive scenarios. The *iBundler* service contributes along two main directions. On the one hand, it incorporated new, actual side constraints to the winner determination problem for multi-item, multi-unit, multi-attribute negotiations[6]. On the other hand, it includes a new ontology that accommodates both operational constraints and attribute-value constraints for buying and providing agents.

However, we have identified two important research issues as future work . In particular, it would be interesting to assess the overload added by the use of our ontology when employed in actual-world settings. On the other hand, it might be of interest to merge our proposal with the ontology in [8, 7], more focused on procedural issues of negotiation protocols.

References

1. Giovanni Caire. Jade tutorial. application-defined content languages and ontologies. Technical report, TILAB S.p.A., June 2002.
2. Agentcities Consortium. Demonstration documentation for checkpoint 1. Technical Report Deliverable D5.4, IST-2000-28385, August 2002.
3. Fipa. http://www.fipa.org.
4. Protege 2000. http://protege.stanford.edu.
5. Juan A. Rodríguez-Aguilar, Andrea Giovanucci, Antonio Reyes-Moro, Francesc X. Noria, and Jesús Cerquides. Agent-based decision support for actual-world procurement scenarios. In *2003 IEEE/WIC International Conference on Intelligent Agent Technology*, Halifax, Canada, October 2003.

6. Juan A. Rodríguez-Aguilar, Antonio Reyes-Moro, Andrea Giovanucci, and Jesús Cerquides Francesc X. Noria. Negotiation support in highly-constrained trading scenarios. In *Proceedings of the Conference of the Spanish Association for Articial Intelligence (CAEPIA)*, San Sebastian, Spain, November 2003. Forthcoming.
7. V. Tamma, M. Wooldridge, and I. Dickinson. An ontology for automated negotiation. In *First Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, Bologna, Italy, July 2002.
8. Valentina Tamma. An experience in using ontologies for automated negotiation. Presentation at the Agentcities Information Day 4, August 2003.
9. C. J. van Aart. Beangenerator. http://www.swi.psy.uva.nl/usr/aart/beangenerator.

Sistema Multiagente para Gestión de Reuniones*

Pedro Cuesta, Juan C. González, Pilar Raña, and Francisco J. Rodríguez

Dpto. de Informática (Universidade de Vigo)
Ed. Politécnico, Campus As Lagoas, Ourense 32004,
pcuesta,jcmoreno,franjrm@uvigo.es
WWW home page: http://montealegre.ei.uvigo.es/gwai/

1. Introducción

El objetivo fundamental que motivó el desarrollo de este sistema fue experimentar la aplicabilidad de la tecnología de agentes en problemas del mundo real.

Bajo esta perspectiva se buscó en primer lugar un problema tipo dentro del paradigma de agentes: gestor de reuniones, que permita convocar de manera autónoma reuniones a los miembros de una organización. A continuación, se evaluaron diferentes herramientas de desarrollo (ZEUS, JADE, BEE-GENT,...), optando por JADE [1] debido a que es una de las plataformas mas usadas, que dispone de suficiente documentación y se ajusta a los estándares (FIPA). Por último, se analizaron distintas propuestas de metodologías para comprobar la utilidad de su aplicación en problemas reales. Fruto de este último análisis se decidió utilizar MaSE [2] debido a su independencia del dominio, a que cubre el ciclo de vida completo, y a estar soportada por una herramienta gráfica: agentTool.

El sistema desarrollado proporciona mecanismos para registrar y agrupar a los diferentes miembros de una organización. Una vez registrados cada usuario puede gestionar su agenda (definir su horario de trabajo, incluir actividades, ...) y solicitar reuniones con otros miembros y/o grupos de la organización. El proceso de gestión de reuniones es el que realmente justifica la utilización de agentes en el sistema, ya que permite que sin intervención directa de otros usuarios se establezca una fecha y hora para la reunión acorde a las agendas de los implicados. Este problema es un ejemplo típico de la necesidad de llevar a cabo procesos de negociación automática en un sistema distribuido.

2. Descripción del sistema

La arquitectura del sistema está formada por tres tipos de agentes:

- *Interfaz de usuario*: este agente sirve de nexo de unión entre cada usuario y el sistema. Existen dos tipos de usuarios y por tanto dos agentes de interfaz.

* Research supported by the spanish national project TIC 2002-04516-C03-01

El agente de interfaz del administrador, permite registrar usuarios y distribuirlos en grupos. Por otra parte cada miembro de la organización a través de su agente de interfaz puede configurar su agenda e iniciar el proceso de petición para realizar una reunión con otros miembros de la organización.

- *Gestor de reuniones*: asociado a cada agente de interfaz se dispone de un agente de este tipo, encargado de atender las peticiones para consultar la agenda de su usuario (de esta forma el sistema es independiente de que los diferentes usuarios estén permanentemente conectados al sistema) y negociar la fecha y hora de la reunión.
- *Informador*: encargado de ofrecer información sobre los usuarios del sistema (ampliación del AMS de JADE para ofrecer información de grupos a los que pertenece, dirección, ...) a los agentes que soliciten dicha información.

Para el correcto funcionamiento del sistema los agentes gestor de reuniones de cada usuario y el agente informador tienen que estar siempre ejecutándose, por lo que su configuración exige la instalación en algún servidor, accesibles al resto de agentes de interfaz.

3. Conclusiones

La aplicación de una metodología concreta de agentes ha sido difícil debido en gran medida a la falta de documentación para su aplicación práctica. Además, aunque la aplicación de la metodología ha facilitado la definición de un modelo adecuado del sistema, la mayor parte de la especificación del modelo no sido de gran ayuda en la implementación con JADE. Por ejemplo, los diagramas de clase de comunicación tienen un nivel de detalle demasiado elevado (los estados por los que pasa cada agente durante la conversación necesitan un nivel de especificación excesivo), mientras que en JADE las comunicaciones se detallan a más alto nivel, no es necesario especificar estos detalles.

La plataforma de desarrollo utilizada, JADE, proporciona los mecanismos adecuados para la construcción del MAS, facilitando la tarea del programador en la implementación de cada uno de los agentes y de la comunicación entre ellos. Un aspecto a mejorar en esta herramienta sería disponer de utilidades de depuración adecuadas al modelo de agente.

Como trabajo futuro (siguiente versión del prototipo) se está pensando en dotar al agente gestor de reuniones de un comportamiento más proactivo para elegir la fecha y hora de una reunión de acuerdo a nuevas exigencias, capturar automáticamente preferencias de un usuario, etc.

Referencias

1. F. Bellifemine, G. Caire, T. Trucco, and G. Rimassa. Jade Programmer's Guide (JADE 2.6). http://sharon.cselt.it/projects/jade/.
2. S. A. DeLoach, M. F. Wood, and Clint H. Sparkman. Multiagent systems engineering. *The International Journal of Software Engineering and Knowledge Engineering*, 11(3), 2001.

HeCaSe: Deployment of Agent-Based Health Care Systems

D. Isern, D. Sánchez, A.Moreno, and A.Valls

Multi-Agent Systems Group (GruSMA)
Computer Science and Mathematics Department
Universitat Rovira i Virgili (URV)
ETSE. Av. dels Països Catalans, 26 (Campus Sescelades)
43007-Tarragona, Catalonia (Spain)
{disern,dsanchez,amoreno,avalls}@etse.urv.es

Overview

HeCaSe (Health Care Services), is an agent-based application that provides medical services. It contains agents (autonomous software entities) that have medical information and implement the behaviour of a set of health care entities of a sanitary region: medical centres, departments and doctors. They coordinate their tasks to simulate the operation of the whole medical organisation and the interaction between them, providing a set of services to the users (citizens or visitors of a city) and to the medical staff (doctors).

The system is part of the Agent Cities initiative, whose main aim is to provide agent-based intelligent services to improve the quality of life of citizens. It is currently running and accessible in the Agent Cities network of platforms. Extended explanations and access instructions can be found on the HeCaSe system website: http://grusma.etse.urv.es/hecase.

System features

- The structure, operation and interaction of medical centres in Catalonia are maintained.
- It provides security measures (encryption, authentication ...) that ensure the confidentiality and privacy of medical data and critical actions.
- There are mechanisms (schedule management and ranking of medical entities) to allow the user to adapt his/her Personal Agent behaviour to his/her profile in order to provide personalised information.
- It provides an ontology for the medical domain (based on standard ones) that eases the share of information between the different entities in an efficient way.
- It uses complex communication protocols that allow the implementation of complicated actions (e.g. personalised booking negotiations).
- It allows the access to the medical services from anywhere at anytime through a computer or a portable device (e.g. laptop) with an internet connection regardless of the OS or the architecture.

Functionalities

1. The patient may request information about medical centres available in a geographical area specifying a set of search constraints.
2. It is possible to book a visit with a doctor due to a medical problem by selecting a proposed appointment from a list of possible ones.
3. The patient has access to his/her own medical record securely.
4. Each patient has an agenda to manage his/her appointments and free time that will be taken into consideration in the booking process (personalised proposals).
5. He/She can also rank the possible doctors, centres or cities in order to teach the system which ones he/she prefers in a future booking negotiation (learning capabilities).
6. Health care personnel can manage their working hours and appointments with patients.
7. During an examination, doctors can access the patient's medical record and update it with the result of the visit (the patient will be able to consult it later).

http://grusma.etse.urv.es/hecase/

http://grusma.etse.urv.es/

Tools for the Design and Deployment of Agent-mediated Electronic Institutions

Marc Esteva

Artificial Intelligence Research Institute, IIIA
Spanish Council for Scientific Research, CSIC
08193 Bellaterra, Barcelona, Spain
marc@iiia.csic.es
http://www.iiia.csic.es

1 Introduction

Human institutions have been successfully mediating among humans interactions for centuries. Wwe advocate that electronic institutions may cope with a similar task within agent societies. We argue that open agent organisations can be modelled as electronic institutions where a vast amount of heterogeneous software and human agents participate playing different roles, and interacting by means of speech acts. Electronic institutions define the rules of the game in agent societies likewise human institutions in human societies. They define what agents are permitted to do and the consequences of such actions. Furthermore, institutions are in charge of enforcing their rules and punishing those agents that violate them.

Developing electronic institutions appear as a highly-intricate task for which support is required. Along this direction, we have developed the ISLANDER editor[2], a tool for the specification and verification of institutions. ISLANDER permits the complete specification of an electronic institution allowing, whenever possible, to specify its elements graphically. We believe that graphical specifications are extremely useful as they facilitate the designer work and they are also much easier to understand. The output of the tool is the specified institution on a defined textual specification language and on XML. Moreover, the tool also verifies the correctness of specifications. The verification process permits the designers to detect errors on their specification prior to the development stage. Once a specification has been produced and verified, the generation of the architecture for an agent-mediated electronic institution can be automated as detailed in [1]. External agents can thereafter log into the institution for participation. Furthermore, the activity of the institution as a whole can be observed via monitoring tools devoted to graphically represent and analyse the behaviour of institutions at both the macro (institutional) level and the micro (agent) level.

The purpose of this demo is to exemplify how the design, verification, development, deployment, and analysis of electronic insitutions can be largely streamlined with the aid of ISLANDER and the tools devoted to their automatic generation and monitoring.

References

1. Marc Esteva. PhD thesis, Institut d'Investigació en Intel.ligència Artificial, 2003.
2. Marc Esteva, David de la Cruz, and Carles Sierra. Islander: an electronic institutions editor. In *Proceedings of the First International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2002)*, pages 1045–1052, 2002.

INGENIAS Environment for MAS Generation

Jorge Gomez Sanz[1], Ruben Fuentes[1], Juan Pavón [2]

[1] *grasia!* Research Group, Facultad de Informática,
Universidad Complutense de Madrid
{jjgomez, jpavon}@sip.ucm.es

Abstract. This paper introduces the INGENIAS Integrated Development Environment, an application for the specification and generation of MAS. INGENIAS IDE is the support tool of the INGENIAS methodology, a methodology that defines notation, models, and a development process. INGENIAS IDE provides facilities that make MAS development closer to conventional software engineering practices.

1 Introduction

INGENIAS IDE [GRASIA 2003], see **Fig. 1**, is a tool designed to generate MAS specifications in a similar way as software engineers use object oriented modeling tools to model systems. INGENIAS IDE bases on meta-modeling techniques to describe different aspects of a MAS. Resulting meta-models were presented in [Gomez at al. 2002]. Meta-modeling techniques have proven to be a valuable tool to guide MAS specification.

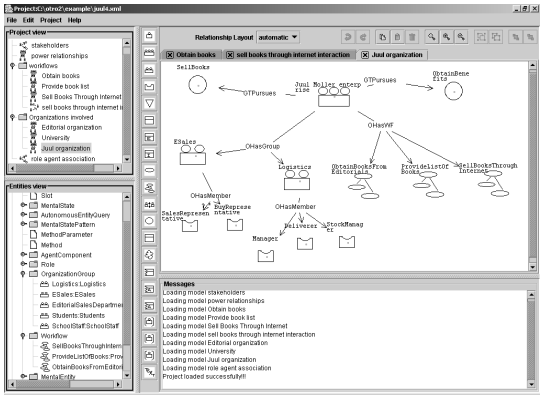


Fig. 1. A snapshot of the INGENIAS IDE applied to the design of bookseller services.

Meta-models are applied during the analysis when looking for a first version of the MAS. In this sense, a meta-model acts as a guideline for the analyst, by indicating which entities has to identify. During design, the meta-model is refined, by identifying new components and relationships among them, in order to achieve the appropriate level of detail. In parallel, the components of the meta-model are translated into

computational entities whose specification, design and implementation are already well known. At the end of the process there is a specification of the MAS with a design that is based on concrete and implementable entities. INGENIAS IDE supports this way of developing MAS with:

- **Diagram generation facilities**. These diagrams are visual representations of MAS meta-models. These diagrams can be reconfigured if the developer discovers new needs on studying the domain problem. Also, to deal with projects with a high number of diagrams, developer can organized them using packages as in object modelling tools.
- **Code generation facilities**. There are libraries and APIs that a developer can use to different purposes such as generate MAS code or documentation directly from the specification. These facilities allow traversing current diagrams from the tool or from a specification file generated from the tool. Other interesting applications of this technique are verification and validation procedures. It is easy to traverse a project's diagrams and determine whether specifications satisfy or not certain properties.

As an example of how this tool helps in a MAS development, readers can consult [Gomez et al. 2003]. Also, readers can access the main distribution site in [GRASIA 2003] if looking for examples of specifications and code templates.

2 Conclusions

INGENIAS IDE is a valuable tool for MAS development. It assists developers in the analysis, design, and implementation stages. Assistance consists in:

- **Making diagram generation simpler.** With help facilities that are accessible from the tool, document generation programs, and showing the developer what entities can be applied in each case and what relationships are allowed among them.
- **Simplifying MAS coding tasks.** It allows to generate MAS without writing a line of code, provided that appropriate libraries are present, or adapting the tool for any MAS platform or MAS framework that the developer is interested in.

As a result, INGENIAS IDE can be considered as flexible tool that can be used by beginners in this field as well as by experienced researchers.

References

[Gomez at al. 2002] Gomez-Sanz, Jorge J., Pavon, Juan, and Garijo, Francisco (2002). *Meta-models for building multi-agent systems*. In Proceedings of the 2002 ACM symposium on Applied computing, pages 37–41. ACM Press.

[Gomez et al. 2003] Gomez-Sanz, J. and Pavon, J. (2003). *Agent oriented software engineering with INGENIAS*. In Multi-Agent Systems and Applications III, volume 2691 of LNCS, pages 394–403, Prague, Czech Republic. 3rd International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003, Springer Verlag.

[GRASIA 2003] GRASIA! research group (2003). INGENIAS IDE. http://ingenias.source.forge.net.

Framework for Developing Mobile Agent Applications

J. Ametller, J. Borrell, J. García, G. Navarro and S. Robles
Departament d'informàtica - Universitat Autònoma de Barcelona
Joan.Ametller@uab.es

1 Problem Description

We present a scenario where Mobile Agents are used to construct Sea of Data applications. In this kind of applications data that must be processed is distributed across several hosts. This data cannot be processed outside the host they are because of legal or bandwidth restrictions. Furthermore, user could be not always on-line or connection could be suddenly interrupted. This is specially not desirable when remote data processing takes long time.

A Mobile Agent System consists of several agencies where agents are executed. This agencies are distributed over several hosts across the Internet and they offer some services to agents. In our scenario, there could be also private networks connected to the Internet through an access point containing also an agency here. Messaging between agents from private networks and agents from the Internet it is not feasible due network restrictions. An Internet agent must migrate to this access point before migrating to a private network agency or using any of its services. Therefore, communications, negotiation processes and resource access to private networks is not feasible without considering agent migration.

Trust problems arise when considering this type of applications. Finding a model for trust in this systems is not a trivial issue. From a mobile agent's point of view, trust regarding other entities in the MAS must be kept. In particular, they need to trust on other agents, service providers, among others. On the other hand, agencies must also use trust. Execution of all agents that arrive to the platforms cannot be allowed; neither service access. Some of this trust problems could be partially solved using cryptography mechanisms, but others are still open problems or are difficult to solve using traditional tools.

2 The Underlying Framework: MARISM-A

MARISM-A [1] (An Architecture for Mobile Agents with Recursive Itinerary and Secure Migration) is an agent platform based on JADE [2] that we have designed to solve problems like those described in previous section. This platform provides tools to easy develop applications based on mobile agents. The key components of MARISM-A are basically the following:

Agent Migration MARISM-A adds inter-platform agent migration to the JADE platform. This is the most important feature of the platform that brings us the possibility to send agents to be executed autonomously to remote places.

Agent Architectures Agent architectures are more than agent abstractions. Under this term we include the definition of the agent taxonomy and all the mechanisms used for agent control. An agency should not know about mobile agent structures. This feature permits that agents based on new architectures (implementing different agent structure designs) to be executed successfully without changing the platform at all.

Services Infrastructure Services provided by the platform are available to agents through an flexible interface. MARISM-A provides some off-the-shelf services (Cryptography Service, Directory Service and Data Store Service), but other can be dynamically installed by agents or the agency.

IDE The aim of this component is to help the user of the system an easy developing of mobile agents. Required user coding is significantly reduced. System specific agent code (initialisation, migration, or data management) is implicit in agent architecture and included by the IDE.

Agent Launcher Agencies are the usual way to introduce agents in an agent system. However, this creates a lot of fault-tolerance and CPU issues, specially in pervasive computing. Indeed, it is not necessary to use an agency to launch agents. Having a light-weight agent launcher, we can connect to a remote agency in order to launch our agents. This can be directly done from the IDE or by using the launcher as a stand alone process.

We have described the main features and components that MARISM-A brings us to solve those problems exposed previously. Now, we introduce a real application using MARISM-A.

3 Application

News agencies need information to publish reports about latest events. This information must be recollected from information sources spread all over the world. This sources are direct witnesses of the events or have obtained indirect information from other parties. Information can be sold after a negotiation to news agencies. News agencies and information sources can use our application to solve hard requirements arising. Information sources are isolated inside private networks. This can be solved using mobile agents as described in the first section.

Its easy to see that there are serious trust problems behind this scenario. News agencies need, in some way, to trust the sources offering information. Not only because they can lie us about the facts described in information, but because the original information could be false. From the other point of view, information sources may not trust on requesting agents.

Sometimes it will not be feasible to check all information that we obtain from the sources. Reputation mechanisms could be a very useful tool to make decisions about selecting sources with an acceptable level of trust. Reputation could be also useful for the information sources in order to get an estimation about the trust of requesting agents.

In this application we show how a mobile agent developed with the MARISM-A IDE is launched from a PDA using the Agent Launcher component. The Agent will start an itinerary composed of several agencies. The goal of this agent is to search information about some event. The agent will also analyse the information to avoid getting repeated copies. Reputation mechanisms will be used by mobile agents and information sources for evaluating the level of trust of other agents.

References

[1] CCD Research Group. An architecture for mobile agents with recursive itineraries and secure migration. `http://www.marism-a.org/`.

[2] Java Agent DEvelopment Framework (JADE) `http://jade.cselt.it/`.

Sistema multiagente para la gestión de agendas personales

Álvaro Rayón Alonso[1], Miguel Ángel Sánchez[2]

[1]arayon@gsi.dit.upm.es
[2]msalcazar@gsi.dit.upm.es

Abstract. Este artículo describe el sistema multiagente desarrollado como parte del proyecto europeo IST-2000-30045 COLLABORATOR [1]. Estos agentes se encargan de obtener un perfil de los usuarios del sistema analizando sus agendas personales y las reuniones que los usuarios han establecido en ellas. Una vez definido el perfil de cada usuario, los agentes son capaces de negociar entre ellos nuevas reuniones teniendo en cuenta las preferencias de los usuarios. En los diferentes apartados de este artículo se hace una descripción detallada de los procesos de obtención de perfiles y de negociación entre agentes.

1 Introducción

En el sistema COLLABORATOR cada usuario tiene una agenda personal y asociada a ella un agente encargado de analizar su contenido. Este agente personal se encarga de establecer el criterio seguido por el usuario a la hora de fijar las reuniones en su agenda personal. Para ello el agente clasifica las reuniones y almacena la tendencia que tiene el usuario a fijar un tipo de reunión en determinada fecha. Basándose en estos perfiles los agentes personales de los usuarios pueden negociar nuevas reuniones sin que los usuarios tengan que intervenir. Una vez fijada la reunión el sistema informa a cada usuario del resultado de la negociación para que estos puedan confirmar o rechazar su participación. Esta información sirve de realimentación a los agentes personales para corregir el perfil que poseen del usuario.
En el siguiente apartado se describe el método empleado para clasificar las reuniones en función de la información que el sistema posee sobre ellas y el proceso que los agentes siguen para obtener las preferencias de los usuarios. Seguidamente se describe el proceso de negociación entre los agentes.

2 Detección de perfiles de usuario

El perfil de cada usuario se compone de información encaminada a la categorización de reuniones e información referente a las preferencias temporales del usuario para fijar reuniones en función de la categoría a la que pertenecen. El usuario define el conjunto de categorías especificando propiedades de las reuniones tales como asistentes, palabras clave que definen el contenido de la reunión y lugar de la reunión. Por su parte el sistema genera automáticamente el conjunto de preferencias temporales observando los intervalos horarios en los que un determinado usuario fija, acepta o rechaza reuniones de una determinada clase. Esta información es expresada mediante hechos que conforman la base de conocimiento del motor de inferencias del agente personal. Un usuario puede visualizar en todo momento las preferencias temporales inducidas por su agente personal, pudiendo añadir nuevas restricciones o bien modificar las ya existentes.

Este perfil debe adaptarse automáticamente a los cambios que un usuario experimenta a lo largo del tiempo a la hora de planificar/clasificar nuevas reuniones. Esto se consigue monitorizando en todo momento las interacciones del usuario con el sistema. En el caso de que el usuario detecte como errónea la clasificación de una reunión, el sistema refuerza negativamente los términos comunes a la reunión y a la categoría seleccionada por el agente y positivamente esos mismos términos en la categoría seleccionada por el usuario. Además cada vez que una nueva reunión es introducida por el usuario, el agente verifica si en base a sus preferencias temporales sería capaz de planificar la reunión en esa misma fecha o en su entorno. En función del resultado obtenido, el grado de confianza en los hechos almacenados en la base de conocimiento del agente y que le han llevado a realizar dicha planificación es reforzado positiva o negativamente, eliminándolos si esta confianza está por debajo de un umbral.

3 Negociación de reuniones

El sistema COLLABORATOR define el concepto de sesión. Una sesión se compone de una serie de aplicaciones orientadas al trabajo en grupo y un conjunto de participantes. Para gestionar el ciclo de vida de estas sesiones se definen los agentes de sesión. Su principal tarea consiste en hacer de mediador entre los diferentes agentes de usuario a la hora de negociar la fecha en la que va a tener lugar una determinada sesión. El administrador de una sesión propone un conjunto de fechas en los cuales la sesión debe tener lugar así como un conjunto de invitados, los conocimientos deseables en dichos invitados, una descripción textual del tema de la reunión, etc. Toda esta información es transmitida desde el agente de sesión a cada uno de los agentes personales de los invitados. Éstos agentes responden al agente de sesión con información relativa a las preferencias temporales y disponibilidad de los usuarios para cada una de esas fechas. Para obtener dichas preferencias los agentes personales consultan el perfil de su usuario: clasificando primero la reunión en una de las categorías definidas para después obtener el grado de disposición del usuario a asistir a dicha reunión en función de la hora así como su disponibilidad. Con toda esta información el agente de sesión determina una fecha para la reunión maximizando una función que depende del número de asistentes y de sus preferencias. Este resultado se comunica a los agentes personales de todos los invitados y estos responden confirmando o rechazando la asistencia de sus usuarios. Si algunos de los agentes personales rechaza su asistencia de forma inesperada para el agente de sesión se busca un nuevo máximo comparándolo con el anterior. Si el nuevo máximo es menor que el anterior el sistema se queda con el resultado obtenido inicialmente. En caso contrario se repite todo el proceso hasta agotar todas las posibilidades.

Además de mediar en la negociación los agentes de sesión se encargar de informar a los agentes de usuario de cambios en el estado de la sesión tales como cancelación, inicio, un usuario decide no asistir, etc., con el fin de garantizar la sincronización de todas las agendas personales de los usuarios.

Referencias

1. COLLABORATOR (COLLABORAtive fRAmework for remoTe and mObile useRs) IST-2000-30045. Available at http://www.ist-COLLABORATOR.net

GenialChef

Recommender agents make recommendations to users according to the information about items (e.g., restaurants) as well as different profiles of other users on the system. Many recommender agents have been developed on the Internet where the typical architecture is a centralised server where users are connected in order to obtain their recommendations. However, when we apply the agents' theory to recommender systems, a standard centralised recommender system becomes a distributed world of recommender agents. Our work is in this latter line: GenialChef is an agent-based system that provides a distributed restaurant recommender service.

This multi-agent system consists of service agents and personal agents (PA). Among the service agents, we distinguish the restaurant server agent (RSA) and the personal agent facilitator agent (PAFA). The RSA provides information about any given restaurant found by name, by address or by location. It can also provide information from its case-based reasoning engine developed over the database. Due to the similarity functions among restaurant attributes, we can retrieve restaurants that are very similar to a given one as well as restaurants closest to what the user has in mind. There could be many RSAs in the system, each one providing information about the restaurants of a given city. The other service agent, the PAFA, acts as a broker agent: it is in charge of assisting PAs to find and contact other PAs.

The PAs are in charge of recommending restaurants to the user. Each user has their own PA in their local machine and the service agents placed in the server give assistance to them. Each personal agent encapsulates the user profile and filters recommendations based on this profile. Therefore, the privacy of the personal data is protected. A content-based filtering method based on case-based reasoning is applied in order to make recommendations to the user based on the content of the items. However, collaborative filtering cannot be applied since the user profiles are protected by each recommender agent and it is impossible to make direct comparisons among them. Collaboration is achieved by means of the exchange of opinions on restaurants. Opinions represent the general interests of the user without revealing detailed information. Therefore, a new information filtering method comes up: the opinion-based filtering method. It consists of reinforcing the recommendation process with opinions from other agents. In order to know which agents to enquire, PAs consider other PAs as personal entities on whom they can rely or. Reliability is expressed through a trust value. Therefore, a social model of trust is proposed in order to represent, initialise and evolve trust values. Thanks to this model of trust, an evolution of the collaborative filtering method can be applied. PAs only ask for advice to their reliable friends. The proposed collaboration generates a social network of PAs really useful in order to study social behaviours.

In order for our agents to communicate, an extension of the ontology of the Agentcities.RTD project has been developed. The ontology encompasses the communication between PAs and RSAs, between PAs and PAFAs and among PAs.

GenialChef has been supported by the Spanish MCYT project DPI2001-2094-C03-01 and the Agentcities.NET project ACNET.02.50 (IRES project).

GenialChef was put forward at the E-TECH 2003 Prizes and was awarded the prize for the best university project. The IRES project was presented to the AgentCities Technology Competition and was awarded the special prize for the best system deployed in the AgentCities Network. Visit http://arlab.udg.es for more information about the project and the prize.

Current Version: v1.0. under JADE 2.6

Publications describing the System:

1. M. Montaner, B. López and J. L. de la Rosa. Developing Trust in Recommender Agents. *In Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02). Cristiano Castelfranchi and W. Lewis Johnson (Eds). ACM Press. vol. 1, pp. 304-305. Bologna (Italy), 2002.*

2. M. Montaner, B. López and J. L. de la Rosa. Improving Case Representation and Case Base Maintenance in Recommender Agents. *In Proceedings of the 6th European Conference on Case Based Reasoning (ECCBR'02). Susan Craw, Alun Preece (Eds.), Lecture Notes in AI N2416. Springer-Verlag. pp. 234-248. Aberdeen (Scotland)*, 2002.

3. M. Montaner, B. López and J. L. de la Rosa. Opinion-Based Filtering Through Trust. *In Proceedings of the 6th International Workshop on Cooperative Information Agents (CIA'02). Matthias Klusch, Sascha Ossowski and Onn Shehory (Eds.), Lecture Notes in AI N2446. Springer-Verlag Berlin Heidelberg, pp. 164-178, Madrid (Spain)*, 2002.

4. M. Montaner, B. López, and J. L. de la Rosa. A Taxonomy of Recommender Agents on the Internet. *Artificial Intelligence Review*, Kluwer Academic Publishers. Volume 19, Issue 4, pp. 285-330, June, 2003.

5. E. del Acebo and J. L. de la Rosa. A Fuzzy System Based Approach to Social Modeling in Multi-Agent Systems. *In Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02). Cristiano Castelfranchi and W. Lewis Johnson (Eds). ACM Press. Bologna (Italy), 2002.*

6. G. González, B. López and J.L. de la Rosa. The Emotional Factor: An Innovative Approach to User Modelling for Recommender Systems. RPeC'02.

IRES

http://arlab.udg.es

AGENTS RESEARCH LAB

UdG

http://www.udg.edu

http://eia.udg.es/arl

Constructing Deliberative Agents using K-SIM case-based reasoning systems

J. M. Corchado[1], E. S. Corchado[2], M. Glez-Bedia[1],
A. de Luis[1] and L. F. Castillo[1]
[1]Departamento de Informática y Automática, University of Salamanca, Plaza de la Merced s/n, 37008, Salamanca, Spain
corchado@usal.es
[2]Dept. de Ingeniería Civil, University of Burgos, Esc. Politécnica Superior, Edificio C, C/ Francisco de Vitoria, Burgos, Spain

Abstract: This paper present a novel technique for constructing autonomous agents using a case-based reasoning system. The reasoning model incorporates a K-SIM algorithm for the identification of the appropriate information to construct the agent plan, in execution time, using variational calculus. The proposed methodology has been used to create several agent based systems. The paper describes the methodology and outlines a case study.

1 Introduction

Interesting case-based reasoning (CBR) models have been presented, over the last few years, for constructing deliberative agents. This paper presents a novel architecture that has been developed for constructing deliberative agents which can generate their plans using the framework of a case-based reasoning (CBR) system, that incorporate a Kernel Maximum Likelihood Hebbian Learning Scale Invariant Map (K-SIM) [1] for the identification of the information that later may be used to create a plan using variational calculus [2]. The robust analytical notation introduced in [3] is used to present the proposal.

Agents must be able to respond to events that take place in their environment, take the initiative according to their goals, interact with other agents (even human) and use past experiences to achieve current goals. The deliberative agent with BDI (Belief, Desire and Intention) architecture, uses the three attitudes in order to make decisions on what to do and how to achieve it [4, 5, 6]: their beliefs represent their information state (what the agents know about themselves and their environment); their desires are their motivation state (what they are trying to achieve); and the intentions represent the agents' deliberative state. These mental attitudes determine the deliberative agent's behaviour and are critical if a proper performance is to be produced when information about a problem is scarce [7, 8]. BDI architecture has the advantage that it is intuitive, it is relatively easy to recognize the process of decision-making and how to perform it. Moreover, it is easy to understand the notions of belief, desires and intentions. For several year we have been working on a mechanism to facilitate the effective implementation of autonomous agents using CBR systems

[9]. A robust analytical formalism for the definition of computationally efficient agents, which solves the first of the previously mentioned problems has been identifies and presented in [2, 3]. This paper presents an improved version of our approach in which a Kernel Maximum Likelihood Hebbian Learning Scale Invariant Map (K-SIM) is used for the identification of the cases required to construct the agent plan.

The analytical notation is reviewed and it is outlined how a CBR system is used to operate the mental attitudes of a deliberative BDI agent. Te K-SIM technique is introduced and an agent based system for the etourism domain that uses the methodology presented in the paper.

2 A strategy for constructing deliberative agents using case-based reasoning systems

A novel methodology for constructing deliberative agent has been presented in [2, 3], this section outlines it and explains why it has to be improved. This section identifies the relationships that can be established between deliberative agents and CBR systems, and shows how an agent can reason with the help of a case-based reasoning system. Case-based reasoning is used to solve new problems by adapting solutions that were used to solve similar previous problems [9, 10]. The operation of a CBR system involves the adaptation of old solutions to match new experiences, using past cases to explain new situations, using previous experience to formulate new solutions, or reasoning from precedents to interpret a similar situation.

The reasoning cycle of a typical CBR system includes four steps that are cyclically carried out in a sequenced way: retrieve, reuse, revise, and retain [9]. During the retrieval phase, those cases that are most similar to the problem case are recovered from the case-base. The recovered cases are adapted to generate a possible solution during the reuse stage. The solution is then reviewed and, if appropriate, a new case is created and stored during the retention stage, within the memory. Therefore CBR systems update their case-bases and consequently evolve with their environment. Each of the reasoning steps of a CBR system can be automated, which implies that the whole reasoning process could be automated to a certain extent [9,10]. This assumption has led us to the hypothesis that agents implemented using CBR systems could be able to reason autonomously and therefore to adapt themselves to environmental changes. Agents may then use the reasoning cycle of CBR systems to generate their plans.

The proposed model has identified a direct mapping between the agents and the reasoning model has been established, in such a way that the mapping between the agents and the reasoning model should allow a direct implementation of the agent and the final agents should be capable of learning and adapting to environmental changes.

The notation and the relationship between the components that characterise a BDI agent is defined as following. Let T be the set that describes the agent environment. If ?(T) is the set of attributes $\{\tau_1, \tau_2, \ldots, \tau_n\}$ in which the world's beliefs are expressed, then we define a belief on T, that is denoted "e",

as an m-tuple of some attributes of ?(T) denoted by e = (τ1, τ2,…,τm) with m≤ n.

We call set of beliefs on T and denote ?(T) to the set:

?(T) = { e=(τ1, τ2,…,τj) / where j =(1,2,…, m ≤ n)}

The operator "Λ of accessibility" between m beliefs (e_1,e_2,e_3,…,e_m), where we denote: $\Lambda(e_1, e_2, e_3,\ldots,e_m) = (e_1 \wedge e_2 \wedge \ldots \wedge e_m)$ indicates that exists compatibility among the set of beliefs (e_1, e_2, e_3,…, e_m). If any of the belief (e_1, e_2, e_3,…, e_m) is not accessible, or if there exists a contradiction, it will be denoted by:$\Lambda(e_1,e_2, e_3,\ldots,e_m) = \emptyset$. Moreover, an intention i on T is defined as an s-tuple of compatible beliefs, i = (e_1, e_2,…,e_s) with s∈ IN and $\Lambda(e_i, e_j) \neq 0$. Then, we call set of intentions on T and denote ?(T)={(e_1, e_2, …, e_k) where k∈IN}. Now a set of parameters will be associated to the space ?(T) that characterises the elements of that set. The set of necessary and sufficient variables to describe the system may be obtained experimentally. We call canonical variables of a set ?(T) any set of linearly independent parameters ℵ =(A_1, A_2,..., A_v) that characterise the elements $i \in$?(T).

In the same way, a desire d on T is defined as a mapping between

$$d : I(T) \longrightarrow ?\ (\aleph)$$

$$i = (e_1 \wedge \ldots \wedge e_r,) \rightarrow F(A_1, A_2,...., A_v)$$

where ? (ℵ) is the set of mappings on ℵ .

A desire d may be achieved constructing an intention i using some of the available beliefs, whose output could be evaluated in terms of the desired goals. We denote D(T) the set of desires on T: D(T)={d: I(T)→ ? (ℵ) / with I(T) set of intentions and ? (ℵ) set of mappings on ℵ }.

2.1 Analytical Notation

Now, after presenting our definition of the agent's beliefs, desires and intentions, it is reviewed the proposed analytical formalism for the CBR system. The necessary notation to characterise a CBR system is introduced as follows. Let us consider a problem P, for which it is desired to obtain the solution S(P). The goal of a case-based reasoning system is to associate a solution S(P) to a new problem P, by reusing the solution S(P´) of a memorised problem P´. P is denoted as P=(S_i,{ $?_j$ }, S_f) with S_i=initial state, S_f=final state and j=(1,…,m). S(P) is defined as S(P)= { S_1, $?_1$, S_2, $?_2$,..., $?_n$, S_{n+1} }={S_k, $?_h$} where k=(1,..,n+1) and h=(1,..,n ≤ m) , S_1=S_i and S_{n+1}= S_f.

The state S_k and the operator $?_j$ are defined as:

$$S_k = \begin{pmatrix} \{O_r\}_{r=1,\ldots,p} \\ \{R_s\}_{s=1,\ldots,q} \end{pmatrix} \qquad ?_j : S_k = \begin{pmatrix} \{O_r\} \\ \{R_s\} \end{pmatrix} \longrightarrow ?_j(S_k) = \begin{pmatrix} \{O'_r\} \\ \{R'_s\} \end{pmatrix}$$

where $\{O_r\}_{r=1,\ldots,p}$ and $\{R_s\}_{s=1,\ldots,q}$ are coordinates in which a state S_k is expressed.

The coordinates type $\{O_r\}_{r=1,\ldots,p}$ are introduced to express the objectives achieved. The coordinates type $\{R_s\}_{s=1,\ldots,q}$ are introduced to express the resources used. Through these definitions, the parameter effectiveness, ℑ, between two states S and S' can be defined, as a vector $\Im(S, S') = (\Im_x, \Im_y)$ which takes the form

$$\Im_x = \frac{O_r(S') - O_r(S)}{O_r \max} \qquad \Im_y = \frac{R_s(S) - R_s(S')}{R_s \max}$$

The definition implies that ($0 \leq \Im_x \leq 1$) and ($0 \leq \Im_y \leq 1$). In particular, if $S=S_i$ and $S'=S_f$, it is denoted $\Im(S_i, S_f) = \Im[S(P)]$ and we call it "effectiveness of a solution". In order to evaluate the rate of objectives achieved and resources used, between S and S', it is necessary to normalise every component of $\{O_r\}_{r=(1,...,p)}$, $\{R_s\}_{s=(1,...,q)}$. Then the expressions that have been defined to sum different objectives are:

If $\{O_r(S)\} = (O_1, O_2, ..., O_P)$ and $\{O_r(S')\} = (O'_1, O'_2, ..., O'_P)$

$$\Im_x = \frac{\sqrt{\left(\frac{O'_1 - O_1}{\max O_1}\right)^2 + \left(\frac{O'_2 - O_2}{\max O_2}\right)^2 + ... + \left(\frac{O'_p - O_p}{\max O_p}\right)^2}}{\sqrt{\left(\frac{\max O_1}{\max O_1}\right)^2 + \left(\frac{\max O_2}{\max O_2}\right)^2 + ... + \left(\frac{\max O_p}{\max O_p}\right)^2}} = \frac{\sqrt{\left(\frac{O'_1 - O_1}{\max O_1}\right)^2 + \left(\frac{O'_2 - O_2}{\max O_2}\right)^2 + ... + \left(\frac{O'_p - O_p}{\max O_p}\right)^2}}{\sqrt{p}}$$

As $\{R_s(S)\} = (R_1, R_2, ..., R_q)$ and $\{R_s(S')\} = (R'_1, R'_2, ..., R'_q)$ it is defined

$$\Im_y = \frac{\sqrt{\left(\frac{R_1 - R'_1}{\max R_1}\right)^2 + \left(\frac{R_2 - R'_2}{\max R_2}\right)^2 + ... + \left(\frac{R_q - R'_q}{\max R_q}\right)^2}}{\sqrt{\left(\frac{\max R_1}{\max R_1}\right)^2 + \left(\frac{\max R_2}{\max R_2}\right)^2 + ... + \left(\frac{\max R_q}{\max R_q}\right)^2}} = \frac{\sqrt{\left(\frac{R_1 - R'_1}{\max R_1}\right)^2 + \left(\frac{R_2 - R'_2}{\max R_2}\right)^2 + ... + \left(\frac{R_q - R'_q}{\max R_q}\right)^2}}{\sqrt{q}}$$

A new parameter is also introduced - efficiency - that measures how many resources are needed to achieve an objective. Given a target problem P, and a solution S(P), we define $\varsigma[S(P)] = \Im_x / \Im_y$, as the efficiency of the solution S(P). The definition implies that $\varsigma(S,S') \in (0, \infty)$. The meaning of this new parameter is explained later. In this domain, a case C is a 3-tuple *{P, S(P), ℑ[S(P)]}* where P is a problem description, S(P) the solution of P and ℑ[S(P)] the effectiveness parameter of the solution, and a CBR´s case base CB, denoted as: *CB={C_k / k=(1,...,q) and* $q \in IR$} that is a finite set of cases memorised by the system. Finally the relationship between CBR systems and BDI agents can be established, associating the beliefs, desires and intentions with cases. Using this relationship we can implement agents (conceptual level) using CBR systems (implementation level). So once the beliefs, desires and intentions of an agent are identified, they can be mapped onto a CBR system. First, a mapping is introduced that associates an index to a given case C_k.

idx:CB→ I(CB)

C → idx(C)= idx{P, S(P), ℑ[S(P)]}= **{** idx**(**S_i**)**, idx**(**S_f**) }=**

= **{** [S_i =(O_1,a_1), (O_2,a_2),...,(O_p, a_p), (R_1,b_1), (R_2,b_2),...., (R_q, b_q)],

[S_f=(O'_1,c_1),(O'_2,c_2),..,(O'_p, c_p),(R'_1,d_1),(R'_2,d_2),..,(R'_q,d_q)] **}**

with Oj, $R_k \in$?(CB) ,a_i, b_j, c_k, $d_l \in IR$ and p, $q \in IN$

where the set I(CB) is the set of indices of a case base CB that is represented by frames composed of conjunction of attributes of T(CB) and values of the domain. The abstraction realized through the indexing process allows the introduction of an order relation R in the CB that can be used to compare cases. Indices are organized in the form of a Subsumption Hierarchy.

(CB, R)={ [C_k / k=(1,..,q) and $q \in IN$], R}={(C_1 , .., C_q)/ idx(C_1) $\subseteq$ .. $\subseteq$ idx(C_q)}

Let us say that two cases C and C'∈ CB fulfill the relation

$$idx(C) \subseteq idx(C') \quad \text{if} \quad \begin{matrix} idx(S_I) \subseteq idx(S'_I) \\ idx(S_F) \supseteq idx(S'_F) \end{matrix}$$

And it is expressed in terms of their components,

$$idx(S_I) \subseteq idx(S'_I) \rightarrow \begin{Bmatrix} O_r(S_I) \geq O_r(S'_I) & \forall r = 1,\ldots p \\ R_s(S_I) \leq R_s(S'_I) & \forall s = 1,\ldots q \end{Bmatrix}$$

$$idx(S_F) \supseteq idx(S'_F) \rightarrow \begin{Bmatrix} O_r(S_F) \leq O_r(S'_F) & \forall r = 1,\ldots p \\ R_s(S_F) \geq R_s(S'_F) & \forall s = 1,\ldots q \end{Bmatrix}$$

Let us say that S(P') is a possible CBR solution of the target P,

$$\forall\ C' = (P', S(P'), \Im[S(P')]) / idx(C') \supseteq P$$

Given a canonical coordinate system $\aleph = (A_1, A_2,\ldots,A_v)$ on I(T), the set may be reordered, differentiating between:

$\{F_m\} = \{A_{j \text{ with } j \leq v} / A_j \text{ growing}\}$ and $\{G_n\} = \{A_{k \text{ with } k \leq v} / A_k \text{ decreasing}\}$ so,

$\aleph = \{F_m\} \cup \{G_n\}$ and m+n=v

Then, giving an $i \in I(T)$, a functional dependency relationship may be obtained in terms of the attributes $i = i[e_1(\tau_1, \tau_2,\ldots,\tau_j), e_2(\tau_1, \tau_2,\ldots,\tau_k),\ldots,e_s(\tau_1, \tau_2,\ldots,\tau_q)] =$

$= i(\tau_1, \tau_2,\ldots,\tau_n)$ and in terms of its canonical or state variables:

$i = i(A_1, A_2,\ldots,A_v) = i(F_1, F_2,\ldots,F_m, G_1, G_2,\ldots,G_n)$ which determines a functional relationship of the type $A_j = A_j(\tau_1, \tau_2,\ldots,\tau_n)$.

Now the fundamental relationship between the BDI agents and the CBR systems can be introduced. We define "state ? of an intentional process" and we denote as $? = \{e_1 \wedge e_2 \wedge \ldots \wedge e_{s-1} \wedge e_s\}$ to describe any of the situations intermediate to the solution $i = \{e_1 \wedge e_2 \wedge \ldots \wedge e_r$, with $r \leq s\}$ that admits a representation over $\aleph$. Moreover, the solution S(P) for a given problem $P = (S_I, \{?_j\}, S_F)$ can be seen as a sequence of states $S_k = (\{O_r\}_{r=1,\ldots,p}, \{R_s\}_{s=1,\ldots,q})$ interrelated by operators $\{?_h\}$.

Given a BDI agent over T with a canonical system, $\aleph = (A_1, A_2,\ldots, A_v)$ in the set I(T) that may be reordered as $\aleph = (F_1, F_2,\ldots,F_m, G_1, G_2,\ldots,G_n)$, we establish the relationship between the set of parameters:

$$\{F_m\} \longleftrightarrow \{O_r\} \qquad \{G_n\} \longleftrightarrow \{R_s\}$$

The identification criteria may be established among

- the intentional states, $?_i \in I(T)$, and the CBR states, $S_k \in T(BC)$.
- and a relationship may be established among the agents desires I(T) and the effectiveness operator $\Im[S(P)]$ of the CBR system.

Then the mathematical formalisation proposed can be used as a common language between agents and CBR system and solves the integration problem. The relationship presented here shows how deliberative agents with a BDI architecture may use the reasoning cycle of a CBR system to generate solutions S(P).

The relationship, presented here, shows how deliberative agents with a BDI architecture may use the reasoning cycle of a CBR system to generate solutions S(P). When the agent needs to solve a problem, it uses its beliefs, desires and intentions to obtain a solution. Previous desires, beliefs and intentions are stored taking the form of cases and are retrieved depending on the current desire. Cases are then adapted to generate a proposed solution, which is the agent action plan.

2.2 Variational calculus based planner

The proposed agents have the ability to plan their actions, to learn and to evolve with the environment, since they use the reasoning process provided by the CBR systems. CBR systems may be implemented and automated in different ways depending on the problem which must be solved. Several strategies may be implemented in the framework of CBR systems to automate the planning tasks [9, 10]. A variational calculus strategy has been proposed in [2] for the adaptation stage of CBR systems embedded in the agent. We have experimented with several planning strategies and observed that the Variational Calculus Based Planer (VCBP) by [2, 3] provides excellent results and can be compared very favorably with other planning strategies [3]. Variational Calculus allows re-planning at execution time, even in changing environments, where goals are to be achieved successfully in real-time.

Variational calculus allows the agents to plan and replan at execution-time because this formalism is used to model the cases during the reuse phase of the reasoning process to solve a given problem. Assuming that potentially significant changes can be determined after executing a primitive action, it is possible to control the dynamism of the new events of the domain and thus achieve an appropriate reconsideration of the problem [11]. When the plan proposed by the agent is stopped for any reason, variational calculus calculates a new plan. In this case the new initial state is the point at which the initial proposed route has stopped. If it is accepted that the environment may change, it is also necessary to define a reasoning mechanism capable of dealing with such changes by modifying the initial desires and intentions.

Nevertheless the reasoning process may be maintained since the general description problem remain constant. If at t_0, the function V(X, Y, Z) takes the form denoted by $V_0(X,Y,Z)$, at t_1, V is denoted by V_1 (X, Y, Z), with the associated surface $?_1$ (X, Y, Z) = 0 on the phase space, upon which it is possible to obtain the optimal curve between two new points, S_i and S_f where $S_i = S_1^{(0)}$, and $S_1^{(0)}$ is the second state of $?_0 = \{ S_i = S_0^{(0)}, S_1^{(0)},....., S_s^{(0)} = S_f \}$ and S_f is the final state or solution state of the global problem. Solving the Euler´s equations, $\mathbf{?_1 = ?_1(X, Y, Z)}$ is obtained, which may be used to calculate an expression for $\mathbf{?_1}$, denoted as $\mathbf{?_1 = \{ S_i = S_1^{(0)}, S_1^{(1)}, S_2^{(1)}, S_3^{(1)}, S_4^{(1)},...., S_m^{(1)}, ..., S_s^{(1)} = S_f \}}$ and the same can be done for any $\mathbf{t_j}$. From the previous equations, and based on variational calculus tools, an expression can be determined to identify the final solution of the agent. This expression, which represents the agent plan, can be obtained in execution-time and takes the following form:

$$\Psi_{final} = \begin{cases} \Psi_0, \ldots \ldots \; t \in (t_0, t_1) \\ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \\ \Psi_{s-1}, \ldots \; t \in (t_{s-2}, t_{s-1}) \\ \Psi s, \ldots \ldots \; t \in (t_0, t_1) \end{cases}$$

2.3 Kernel-SIM preprocessing strategy

The performance of the variational calculus strategy used in the reuse stage relies on the effectiveness of the retrieval algorithm. The more consistence and adequate the outlines cases, the better the performance of the planning strategy. This paper presents an algorithm, that has been developed for case indexing and retrieval and that has been presented in [1]. Kernel Maximum Likelihood Hebbian Learning Scale Invariant Map (K-SIM) is based on a modification of a new type of topology preserving map that can be used for scale invariant classification [12]. Kernel models were first developed within the context of Support Vector Machines [13]. Support Vector Machines attempt to identify a small number of data points (the support vectors) which are necessary to solve a particular problem to the required accuracy. Kernels have been successfully used in the unsupervised investigation of structure in data sets [11, 14, 15].

Kernel methods map a data set into a feature space using a nonlinear mapping. Then typically a linear operation is performed in the feature space; this is equivalent to performing a nonlinear operation on the original data set. The Scale Invariant Map is an implementation of the negative feedback network to form a topology preserving mapping. A kernel method is applied to an extension of the Scale Invariant Map (SIM) which is based on the application of the Maximum Likelihood Hebbian Learning (MLHL) method [16]. This method automates the organisation of cases and the retrieval stage of case-based reasoning systems. The proposed methodology groups cases with similar structure, identifying clusters automatically in a data set in an unsupervised mode. The features that characterize Kernel models can be used to cluster cases, to identify cases that are similar to a given one and to reuse cases. Then, the VCBP can generate a smoother representation of the problem and identify a more consistent plan.

3 Preliminary results and conclusions

The proposed system has been used to improve an agent based system developed for guiding tourist around a city. The initial system has been presented in [2, 3]. Basically the assistant agent can be contacted via Internet or wireless devices such as mobile phones, PDAs, etc. The initial advising agent has been improved using the methodology presented in the previous section. The agent shares information with other agents that maintain uptodate information about Salamanca, its monuments, restaurants, spectacles, etc. The agent interact with the user and provides them with a plan. The tourist may use a mobile device to contact the agent, and then introduces his/her login and password, and indicates to the agent his/her preferences (monuments to visit, visits duration, time for dinner, amount of money to spend, etc.). The agent then generates a plan for the user according to his/her preferences and sends it back to the visitor.

Once the agent is contacted and knows what are the interest of the tourist, it stars it reasoning process. First identifies what cases should be retrieved to create the model of the problem using the K-SIM algorithm and then to identify the plan using the

VCBP algorithm. If at any point the tourist decides to change his mind, the agent may run the K-SIM algorithm, and the VCBP planner to modify the initial plan in execution time, taking into consideration the initial constraints together with new ones.

The initial system, was tested from the 1st of June to the 15th of September 2002. The case base was initially filled with information collected from the 1st of February to the 25th of May 2002. Local tourist guides provided the agent with a number of standard routes and distributed among his clients Mobile phones, from which they could contact the agent and inform it about the progress of their plans: routes, times, evaluations, etc. As reported in [2], during this period the agent stored in its memory 540 instances. Which covered a wide range of all the possible options that offers the City of Salamanca. The system was tested during 115 days and the results obtained were very encourages. Three hotels of the City offered the option to their 4216 guests to use the help of the agent or a professional tourist guide, 7% of them decided to use to agent based system and 28% of them used the help of a tourist guide. The rest of the tourists visited the city by themselves. In this initial experiment the agent intentions were related to a one-day route (a maximum of 12). The degree of satisfaction of the tourist that used the help of the agent based tourist guide was very high. The new methodology has tested on bench, and we only have preliminary results, which can not be compared with the previously obtained and reported in [2], never the less we have observed that plans obtained with the improved system are more efficient in terms of the reduction of times spent between visits, identification of more convenient restaurants and monuments to visit with respect to the location and distance. The system will be operational in January 2004 and will be tested for a four month period.

Acknowledgements

This work has been partially supported by the CICYT projects TEL99-0335-C04-03 and SEC2000-0249 and the project SA039/02 of the JCyL.

Bibliography

1 Corchado J. M., Corchado E. S., Aiken J., Fyfe C., Fdez-Riverola F. and Glez-Bedia M. (2003) Maximum Likelihood Hebbian Learning Based Retrieval Method for CBR Systems. 5th International Conference on Case-Based Reasoning, Trondheim, Norway, June 23 to 26.

2 Glez-Bedia M. and Corchado J. M. (2002). Variational Calculus Bases Planning for Deliberative Agents. I Workshop on Planning, Scheduling and Temporal Reasoning. Sevilla, Spain, 12 November.

3 Glez-Bedia M., Corchado J. M., Corchado E. S. and Fyfe C. Analytical Model for Constructing Deliberative Agents, Engineering Intelligent Systems, Vol 3: pp. 173-185.

4 Glez-Bedia M. and Corchado J. M. Constructing autonomous distributed systems using CBR-BDI agents. Innovation in knowledge Engineering. R. J. Howlett and L.C. Jain (eds.) Advanced Knowledge International.
5 Jennings N.R. (1992) On Being Responsible. In Y. Demazeau and E. Werner, editors, Decentralized A.I. 3. North Holland, Amsterdam, The Netherlands.
6 Wooldridge M. and Jennings N. R. (1994) Agent Theories, Architectures, and Languages: A Survey. Procs. ECAI-94 Workshop on Agent Theories, Architectures, and Languages.
7 Bratman M. E. (1987) Intentions, Plans and Practical Reason. Harvard University Press, Cambridge, M.A.
8 Kinny D. and Georgeff M. (1991) Commitment and effectiveness of situated agents. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI'91), pp. 82-88, Sydney, Australia.
9 Corchado J. M. and Aiken J. Hybrid Artificial Intelligence Methods in Oceanographic Forecasting Models. IEEE SMC Transactions Part C. Vol. 32, No.4. pp. 307-313.
10 Corchado J. M. and Lees B. (2001) A Hybrid Case-based Model for Forecasting. Applied Artificial Intelligence. Vol 15, no. 2, pp 105-127.
11 Fyfe C. and Corchado J. M. (2001) Automating the construction of CBR Systems using Kernel Methods. International Journal of Intelligent Systems. Vol 16, No. 4.
12 Fyfe C. (1996) A Scale Invariant Map Network. Computation in Neural Systems, Vol. 7, pp. 269-275.
13 Vapnik V. (1995) The nature of statistical learning theory", Springer Verlag.
14 Scholkopf B., Smola A. and Muller K. R. (1998) Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation, Vol.10, pp. 1299-1319.
15 Fyfe C., MacDonald D., Lai P. L., Rosipal R. and Charles D. (2000) Unsupervised Learning with Radial Kernels. In Recent Advances in Radial Basis Functions, Editors R. J. Howlett and L. C. Jain.
16 Corchado E. S., MacDonald D. and Fyfe C. (2003) Maximum and Minimum Likelihood Hebbian Learning for Exploratory Projection Pursuit. Data Mining and Knowledge Discovery. In Press.

Towards a recursive agent oriented methodology

Adriana Giret, Vicente Botti

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia, Spain
46020 Valencia, Spain
{agiret, vbotti}@dsic.upv.es

Abstract. Current business trends, policy markets, production requirements, etc., have created the need for integrating Multi Agent Systems (MAS). In the agent-specialized literature, we have found very little work about MAS methodologies that allow us to build MAS which is made up of two or more MASs. We think that several difficult challenges for automated systems can be tackled by giving full meaning to the MAS concept: adopting a recursive definition of MASs. In this work we outline the basis for a recursive agent oriented methodology for large-scale MAS.

1 Introduction

Nowadays arises the need to integrate pre-existent Multi Agent Systems (MAS) in domains where these integration and/or cooperation are imposed by business trends, policy markets, production requirements, etc. The question arises as to whether a MAS can be a collection of several interacting MASs, a hierarchy, or some other type of organization.

A large-scale MAS encompasses multiple types of agents and may, as well, encompasses multiple MASs, each of them having distinct agency properties. A large-scale MAS needs to satisfy multiple requirements such as reliability, security, adaptability, interoperability, scalability, maintainability, and reusability. But, how can we specify and model the agency properties of a MAS made up of others MASs? We believe that MAS have to integrate recursive aspects to be able to comply with these requirements.

MAS offers powerful tools to realize complex problem spatialized solving or simulations systems. Some of these problems present hierarchical and multi-scale requirements and evolve in structured environments which posses recursive properties. In the intelligent manufacturing field, the need for some kind of hierarchical aggregation in real world systems has been recognized. These systems have to remain readable while they are expanded in a wide range of temporal and spatial scales. For example, a modern automobile factory, incorporates hundreds of thousands of individual mechanisms (each of which can be an agent) in hundreds of machines which are grouped in to dozens or more production lines. Engineers can design, build, and operate such complex systems by shifting from the mechanism, to the machine or to the production line (depending on the problem at hand) and by recognizing the agents of higher levels as aggregations

of lower-level agents. Also, in e-commerce applications, an enterprize is a legal entity which is independent of the individual people whose are its employees and directors.

In the agent-specialized literature, we have found very little work about methodologies which allow us to carry out recursive and dynamic analysis, design, and implementation of MAS. Most of the current approaches start from an atomic agent definition such as an indivisible entity and build MAS as compositions of interacting agents. Nevertheless, there are some works in which a nested MAS structure is included. Parunak and Odell, in [1], proposed UML conventions and AUML extensions to support nested agents' groups. Wagner, in [2], models an institutional agent which is made up of agents themselves.

In this work we try to define a set of concepts to help in the construction of large-scale MASs. The aim of this paper is to introduce a recursive agent model and an outline of a recursive MAS methodology. We are convinced that several difficult challenges for automated systems may be tackled by giving full meaning to the agent concept: adopting an abstract recursive agent definition. To this end we present the definition for abstract recursive agents in section 2. In section 3, we show an example of a large-scale MAS modelled as a recursive MAS. In section 4, we introduce an outline of a recursive agent oriented methodology. Finally we state our conclusions in section 5.

2 Abstract Recursive Agent

With a recursive approach to develop Multi Agent Systems as systems in which their components may be MASs themselves, the idea is as follows [1]. When we begin to analyze a group of agents (MAS) A, we identify the agents $\{a_1, a_2, ..., a_n\}$ which execute certain functions. These agents may encapsulate individual persons, physical, or software entities (agents). They may also be other groups of MAS, say B, so we can have $a_i = B_i$, which we treat as black boxes. We can take this perspective as long as our analysis can ignore the internal structure of the member groups (MAS). However, subsequent analysis generally needs to 'open' these black boxes and look inside them to see the agent components and their corresponding functions; for example, when analyzing B we have that $B = \{b_1, b_2, ..., b_m\}$. At this point, we insist on identifying which of B's member agents is actually responsible for filling B's role in A.

To support these ideas it seems appropriate to provide an abstract recursive agent (A-Agent) definition which will allow us to build Multi Agent Systems. This definition is based on the widely known agent definition of Wooldridge and Jennings [3].

Definition 1. *An A-Agent is a software system with a unique entity, which is located in some environment, which as a whole, perceives its environment (environment sensitive inputs). From these perceptions, it determines and executes actions in an autonomous and flexible way - reactive and proactive. These actions allow the A-Agent to reach its goals and to change its environment. From*

a structural point of view, an A-Agent can be an agent (atomic entity); or it can be a Multi Agent System (with a unique entity) made up of A-Agents which are not necessarily homogeneous.

An A-Agent is in a higher conceptual abstraction level than an agent. An A-Agent can be seen as a MAS, an organization, a federation or an institution with the added value that it can also be a composition of all this abstraction models. Furthermore, when we define two interacting A-Agent we could also be modelling two interacting organizations, federations, MAS or institutions. An A-Agent will exist only at modelling stages, in the end (at coding stages) it will be replaced possibly by a group of agents or also by a single agent.

Definition 2. *A Multi Agent System is made up of two or more A-Agents which interact to solve problems that are beyond the individual capabilities and individual knowledge of each A-Agent.*

Definition 2 extends the traditional notion of MAS when indicating that a MAS is made up of MASs. This will allow us to build systems in which the building blocks are interacting MASs that work together to reach one or several global goals (the global goal is the goal of the system as a whole).

From definition 1 we have:

- A-Agent of level 0 is an agent.
- A-Agent of level 1 is a traditional MAS.
- A MAS is an A-Agent.
- A-Agent of level n is a MAS made up of interacting MASs.

In [4], we have also proposed a formalization of MAS behaviours in terms of its constituent agent behaviour or MAS behaviour. In summary, the reactive behaviour of a MAS is determined by its perception which is defined as the union of the set of perceptions of its members, and by its actions, which in turn are defined as the union of the group actions executed by its members and the union set of the primitive actions carried out by each of its constituent entities. The intentional behaviour of a MAS is determined by its goals. These goals are defined as the union of the set of systems' goal derived from congruent patterns of interactions among its members and the set of joint goals of its constituent entities.

3 A large-scale MAS as a recursive MAS

One of the most difficult challenges for automated systems is scalability and adaptation. In life systems there are many useful concepts, including examples on how to scale up, evolve, adapt, interoperate, organize, and so on. Complex and adaptive life systems are large, intricate and require active autonomous entities. Life systems are recursive and they enable the construction of very complex systems from more simple entities [5]. What about MASs? Are MAS MASs arranged in clusters, a hierarchy, or some other type of organization? In this section, we present an example in which the usefulness of our definition can

be observed to describing complex large-scale problems with multiple levels of abstraction.

Let us suppose a Multinational company, called AG, which has different National companies distributed among different countries. The objective is to model the multinational as a MAS.

Each National company can be an A-Agent since it has got agenthood characteristics. The National company is autonomous in its national environment; it acts in the national market with its owns market and production rules. At the same time, it must be able to interact with others National companies to exchange materials, personnel, knowledge, etc. The National company, is also governed by the rules and norms of the Multinational for its international relations (other National companies).

The international companies' relationships define the rules, norms and policyies of the multinational. There are geographical areas in which the relationships among the national companies are narrower. In addition the commercial agreements among the different countries define new interrelation rules among the national companies of these zones. For example, in Europe, the European Union countries are governed by certain standards and norms of the community; and in South America the Southern Cone Common Market - MERCOSUR (Paraguay, Argentina, Chile, Brazil, Uruguay and Bolivia) and the countries of the Andean Community (Bolivia, Colombia, Ecuador, Peru and Venezuela). The relationships of the countries of these markets with other countries or regional markets are managed by their local market rules. Each market can be modeled as an A-Agent. It is important to note that Bolivia, as a National Company, belongs to two Regional Companies (MERCOSUR and the Andean Community).

Up to this point, we have identified 4 levels of abstractions (Figure 1(a)): the Multinational company, the Regional companies, and the National companies. We have been able to model the Multinational as a MAS, which is composed by A-Agents that are related to each other with certain behavior patterns that define the Multinational company. If the National companies will be made up of agents (A-Agents of level 0), we can think of a National company as a traditional MAS (A-Agent of level 1), the Regional companies as A-Agents of level 2 and the Multinational as A-Agent of level 3.

Apart from modeling the outside relationships, if the designer's interest is also to model the internal structure of each National company, the National company should be observed from inside. Inside each National company there would be new distributed companies in different cities or with autonomy for certain activities. In turn, each Local company is subordinated to the National company and each National one to the Multinational. Thus we have a new level of abstraction, the Local company as an A-Agent of level 1, the National company as an A-Agent of level 2, the Regional company as an A-Agent of level 3 and the Multinational as an A-Agent of level 4 Figure 1(b).

If the National company is not subdivided into city companies or autonomous companies. Then the National company is a traditional MAS composed of national domain-specific agents (A-Agents of level 0), which are interrelated agents

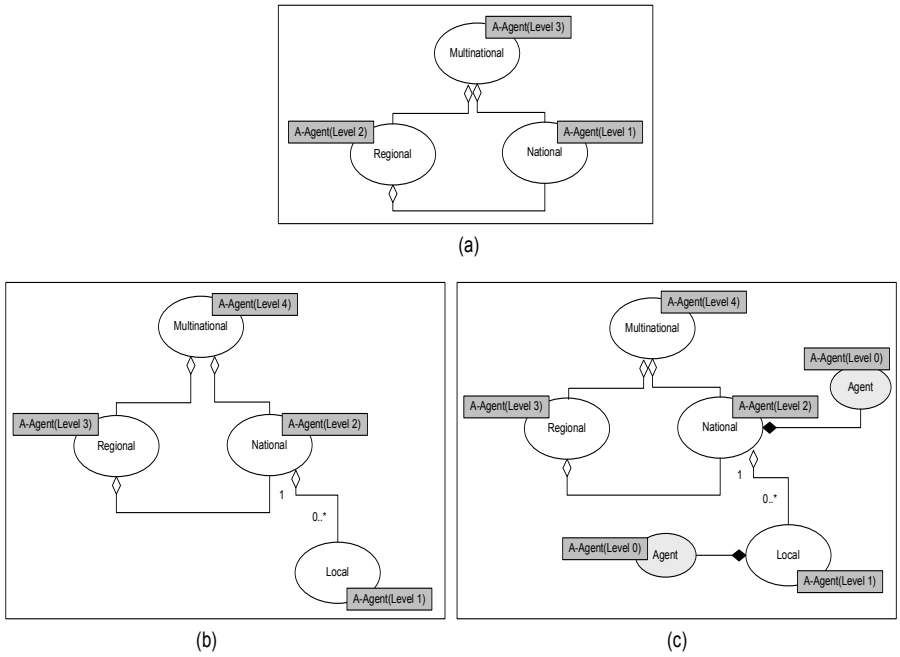


Fig. 1. Different levels of abstraction of the Multinational Company.

and carry out specific functions. These national domain-specific agents define the services provided by the National company inside the country and outside the country. This very same analysis should be made for each Local company until we reach the agents, which define and implement the company as a whole. In summary, the final result of the analysis should be similar to Figure 1(c). In Figure 1(c), it can be observed that the National company is composed of zero or more Local companies, and each Local company, in turn, is an A-Agent of level 1.

Again, the Multinational can be considered from outside as an A-Agent, since it is located in an environment, the world market; it is autonomous; it has its own economic and market policies; it is social, i.e. it interacts with other entities for purchasing, selling, recruiting, leasing, etc.; it is pro-active, since for example according to the world market trends it is able to modify its current market policies, etc.

4 Recursive Agent Oriented Methodology

In this section we will introduce an outline of a recursive agent oriented methodology. This methodology will help us in the construction of MAS made up of pre-existing MASs or MAS made up of interacting MASs emerged from groups of

agents which have close interactions and which implement a well defined functionality. As a result this group of close agents can be encapsuled as a new MAS (A-Agent) and hence modeled and implemented as an entity with its owns characteristics.

The recursive agent oriented methodology tries to reduce the complexity of large-scale MAS, dividing the domain problem in simpler sub-problems and considering every sub-problem as an A-Agent. Every such A-Agent can in turn be decomposed in simpler interacting A-Agents. Until we reach an abstraction level in which there is no more subdivision, that is all the constituent members of the MAS (A-Agent) are agents. We can think of the methodology as a MAS-driven methodology.

The recursive agent oriented methodology models each MAS dividing it in more concrete aspects that form different *views* of the system. This idea already appears in the work of Kendall [6], MAS-CommonKADS [7], and later in GAIA [8]. The way in which the views are defined is inspired by INGENIAS methodology [9].

To describe a MAS, the developer will use the following models:

– **A-Agent Model**: Describes agents and A-Agents, their task, goals, initial mental state, and played roles. Moreover, these models are used to describe intermediate states of agents and A-Agents. These states are presented using goals, facts, tasks, or any other system entity that helps in its state description. The constructs added to the INGENIAS notations for A-Agent and its associated abstract-tasks, abstract-goals and abstract-mental state are depicted in Figure 2. In contrast to INGENIAS in our approach a group (as well as an organization) of agents -an A-Agent- executes abstract-task, has abstract-goals and abstract-mental states. An A-Agent may play a role, but it can not execute task, can not have goals and mental states. All these abstracts entities have to be implemented by real entities. That is, for each A-Agent identified at every development step, there will be a group of agents that will implement the corresponding functionality, will execute the corresponding task, will have the corresponding goals and mental states. An abstract task will be implemented by a work flow. An abstract goal will possibly be replaced by a conjunction of goals or by a common goal and so on.

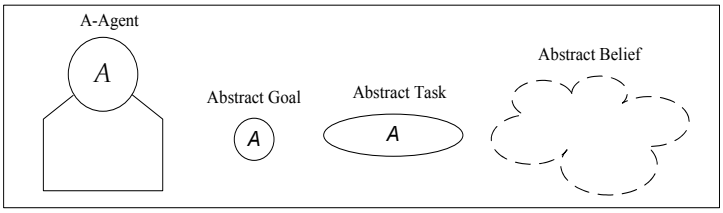


Fig. 2. Abstract Notations for the A-Agent Model.

In this work we focus on the A-Agent meta model. To define the A-Agent meta-model we add the gray entities and the bold lines identified in Figure 3 to the agent meta-model of INGENIAS. Following INGENIAS we use GOPRR (Graph, Object, Property, Relationship, and Role)[10] primitives in UML notation.

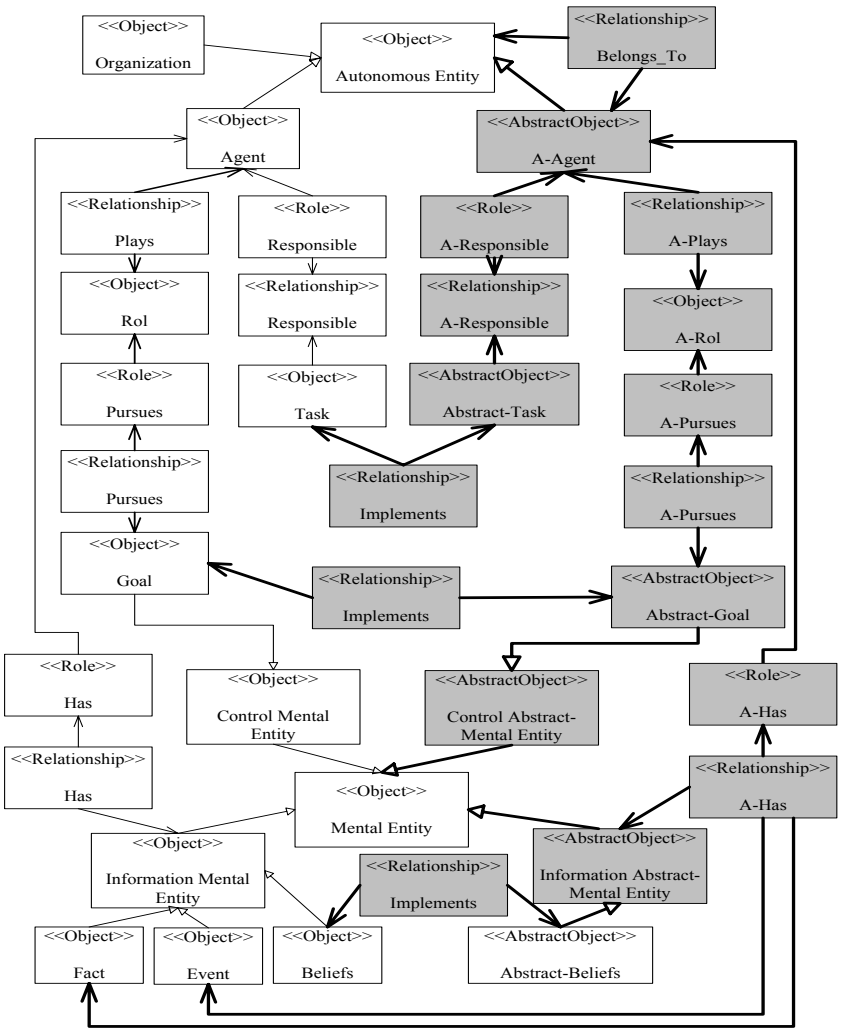


Fig. 3. Main Entities in the A-Agent meta-model.

– **Interaction model**: Describes how interaction among agents and A-Agents takes place. Each interaction declaration includes involved actors, goals pursued by interaction, and a description of the protocol that follows the inter-

action. An abstract interaction is an interaction in which at least one of the speakers is an A-Agent.
- **Task-goals model**: Describes relationships among goals and tasks, goal structures, and task structures. It is also used to express which are the inputs and outputs of the tasks and what are their effects on environment or on A-Agent's mental state. This model describes also the subdivision of the abstract task, identified in the A-Agent model, into a set of tasks.
- **Organization model**: Describes how system components (agents, A-Agents, roles, resources, and applications) are grouped together, which are executed in common, which goals they share, and what constraints exists in the interaction among agents and among A-Agents.
- **Environment model**: Defines agent's and MAS's perception in terms of existing elements of the system.

These models will help in the modelling of MASs of any level of abstraction (that is a conventional MAS -level 1- made up of agents, a MAS of level 2 made up of MAS of level 1, and so on). In every model the definition of the MAS behaviour is carried out following the work presented in [4].

The software development process guided by this methodology will be a recursive, incremental and concurrent MAS driven process, see Figure 4. It will have as many iterations as levels of abstractions are identified. The result of each iteration will be a MAS of level n in which its structure and functionality are defined in terms of the previous models. In each new iteration there will be as many concurrent processes as non defined MASs of level $n-1$ of the previous iteration (because, we can have pre-existing MAS of level $n-1$). Each iteration will be a recursive, incremental and concurrent process.

Each iteration will be conducted in the following way. In the analysis phase, organization models will be produced to sketch how the MAS looks like. In this step, we can identify potential constituent A-Agents (that is, group of agents with close interaction, with an identified functionality or goal, in which every member interacts to solve some problem, and with self-regulating rules of actions) or pre-existing constituent MASs. The models obtained in this phase will be refined later to identify common goals and relevant tasks to be performed by each A-Agent (task-goal model and environmental model). More details will be added specifying A-Agent interactions with interactions models and environmental models, and, as a consequence, refining A-Agents's mental state with A-Agent models. For each emerged MAS a new process is started, until we reach a step in which there are no more non specified MASs (A-Agents of level > 0).

5 Conclusion

In this work, we have presented a recursive approach to develop large-scale MASs as systems in which their components may be MASs themselves.We have illustrated an outline of a recursive, incremental and concurrent agent oriented methodology. The recursive methodology models each MAS dividing it in more concrete aspects that form different *views* of the system. The way in which the

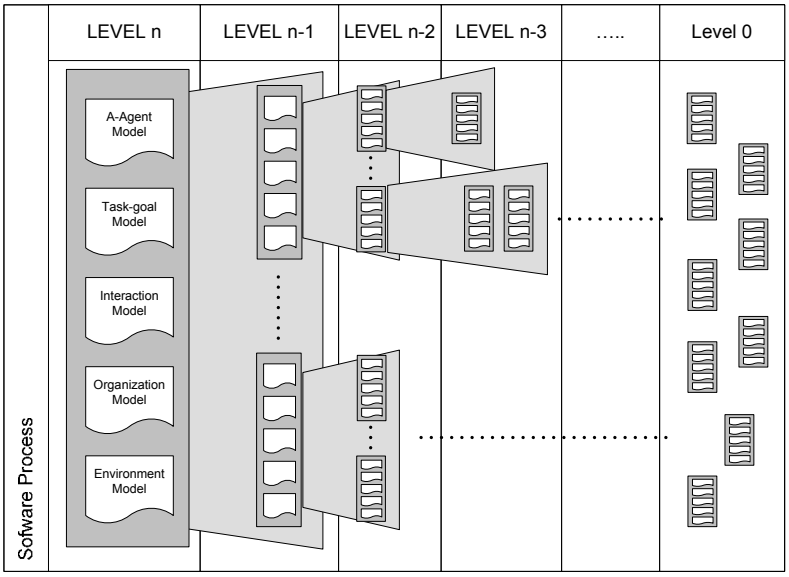


Fig. 4. Software Development Process.

views are defined is based in INGENIAS methodology [9]. We have presented a sketch of the A-Agent meta-model. The software process will have as many iterations as levels of abstractions are identified. The result of each iteration will be a MAS of level n in which its structure and functionality are defined in terms of: A-Agent models, Task-goal models, Interaction models, Organization models and Environment models. In each new iteration there will be as many concurrent processes as non defined MASs of level $n-1$ of the previous iteration (because, we can have pre-existing MAS of level $n-1$). Each iteration will be a recursive, incremental and concurrent process.

This paper is a preliminary report of our research. There are a lot of open problems. In [4], we pointed out a definition of MAS' behaviour in terms of its constituent members in a top-down fashion. We also need to formalize it in a bottom-up way to deal with the definition of the behaviour of MAS' emerged from pre-existing agents. We have to complete the definition of the four models and meta-models to comply with the recursive properties in A-Agent structures. We have to formalize the data flows relations among the four models. There is no CASE tool to help in the software recursive process.

References

1. Parunak, V.D., Odell, J.: Representing social structures in UML. In Agent-Oriented Software Engineering II, M. Wooldridge, G. Weiss, and P. Ciancarini, eds. Springer Verlag (2002) 1–16

2. Wagner, G.: Agent-oriented analysis and design of organizational information systems. in J. Barzdins and A. Caplinskas (Eds.), Databases and Information Systems. Kluwer Academic Publishers. (2001) 111–124
3. Wooldridge, M., Jennings, N.R.: Intelligent agents - theories, architectures, and languages. Lecture Notes in Artificia Intelligence, Springer-Verlag. ISBN 3-540-58855-8 **890** (1995)
4. Giret, A., Botti, V.: Recursive agent. In Proceedings of Iberagents 2002, Agent Technology and Software Engineering (2002)
5. Koestler, A.: The Ghost in the Machine. Arkana Books (1971)
6. Kendall, E.: Developing agent based systems for enterprise integration. IFIP Working Conference of TC5 Special Interest Group on Architectures for Enterprise Integration (1995)
7. Iglesias, C., Garijo, M., Gonzalez, J., Velasco, J.: Analysis and design of multi agent systems using MAS-CommonKADS. In Singh, M. P., Rao, A., and Wooldridge, M.J. (eds.) Intelligent Agentd IV LNAI, Springer Verlag **1365** (1998)
8. Wooldridge, M., Jennings, N.R., Kinny, D.: The Gaia methodology for agent-oriented analisys and design. Journal of Autonomous Agents and Multi-Agent Systems **15** (2000)
9. Gomez, J., Fuentes, R.: Agent oriented software engineering with INGENIAS. In Proceedings of Iberagents 2002, Agent Technology and Software Engineering (2002)
10. Lyytinen, K.S., Rossi, M.: METAEDIT+ - a fully configurable multi user and multi tool CASE and CAME environment. Springer Verlag LGNS 1080 (1999)

A Framework for Evaluation of Agent Oriented Methodologies*

Pedro Cuesta, Alma Gómez, Juan C. González, and Francisco J. Rodríguez

Dpto. de Informática (University of Vigo)
Ed. Politécnico, Campus As Lagoas,
Ourense E-32004 (SPAIN),
pcuesta,alma,jcmoreno,franjrm@uvigo.es
WWW home page: http://montealegre.ei.uvigo.es/gwai/

Abstract. Agent Oriented Methodologies have become an important subject of research in advanced Software Engineering. Several methodologies have been proposed, as a theoretical approach, to facilitate and support the development of complex distributed systems. An important question when facing the construction of Agent Applications is deciding which methodology to follow. Trying to answer this question, a framework with several criteria is applied in this paper for the comparative analysis of existing multiagent system methodologies. The results of the comparative over two of them, conclude that those methodologies have not reached a sufficient maturity level to be used by the software industry. The framework has also proved its utility for the evaluation of any kind of Agent Oriented Software Engineering Methodology.

1 Introduction

In the latest years, Agent Oriented Computing has emerge as an important issue in the Software Engineering field [2]. The ideas behind the concept of Agent (autonomy, social ability, reactivity/responsiveness, proactivity and intelligence: veracity, benevolence, rationality and cooperation) provide a powerful abstraction tool in the solution of complex distributed problems [12].

The interest, in the Software Engineering Approach, is focused in MultiAgent Systems (MAS) [17, 18]. A MAS is a set of autonomous agents which work cooperatively to achieve their goals. Each agent may interact with its environment or with other agents, using high level communication languages and protocols, in order to coordinate its activities and to obtain services and/or information.

Following the usual evolution in computer sciences, initially only the languages and tools which incorporate the new concepts were defined. Afterwards, the study of suitable methodologies for guiding the software development process was addressed. This evolution has implied that there are many agent oriented applications. But, until the moment, developers think that agent tools and methodologies have not achieved, the maturity level needed for being used under warranty in commercial software development.

* Research supported by the spanish national project TIC 2002-04516-C03-01

Nevertheless, agent oriented methodologies are essential in order to solve complex problems and lead the development process [10]. The construction of safe, flexible, scalable and specification adjusted MAS requires methodologies and modelling techniques which handle this complexity using the suitable supports: abstraction, structure, modularity, etc. Current Agent Oriented Software Engineering Methodologies (AOSEM) have evolved from Object Oriented Software Engineering Methodologies [7, 19] or from Knowledge Engineering Methodologies [11].

In this context we consider that the evaluation of current AOSEM will lead to important results that can afterwards be applied in the agent oriented software development: the necessity of a completely new methodology, the integration of the best characteristics of the current methodologies for obtaining a standard one, the necessity of additional documentation (samples, study cases, ...), etc.

This paper presents a framework for the comparative analysis of AOSEM. The proposed framework defines a set of relevant criteria for the evaluation of methodologies. These criteria have been introduced having in mind two different points of view. The first one is related to the software engineering issues (steps of development, models defined, ...). The other one has been the support provided to agent based orientation (definition of agent, features of communication, cooperation, ...). The comparison of AOSEM following these criteria will include the conceptual aspects mentioned as well as their practical application (in accordance with the documentation provided by each methodology).

Similar works [13, 15] have adapted existing frameworks in the field of classical software engineering for being used in agent methodologies. The most important drawback in the studied cases was that the defined criteria were difficult to apply in practice. These approaches introduce criteria which can only be evaluated in a subjective way. Trying to overcome these difficulties, this framework does not include subjective evaluation and introduces specific criteria for the agent oriented field. Moreover, we think that the framework could be used to evaluate future proposals; and also to compare them with the existent ones to decide whether the new methodology is useful or not in the context of agent development.

In [5] the framework has been applied to some of the most relevant proposals in AOSEM. The results obtained from this study will be used, afterwards, for achieving our global goal, that is, the definition of a methodology which integrates the best practices of these proposals and that covers the existing gap between theory and practice in Agent Applications [3, 4].

The reminder of the paper is organized as follows. First the evaluation framework is introduced and explained. After this, a case study in which we recover the evaluation results over MaSE [6, 7] and MAS-CommonKADS [9, 11] is presented. Finally, detailed conclusions about the framework and the results that can be obtained using it are shown.

2 The AOSEM Evaluation Framework

For better understanding, the criteria utilised in the evaluation framework have been classified in five groups according to their area of interest. The first group of evaluation criteria is called *Development Process* and incorporates general aspects of the methodology as well as other related with the stages in the construction of the system. The second one, *Model View*, tries to reflect the methodology concepts and their representation. *Agent* group, which is considered in third place, addresses the individual agent characteristics taken into account by the methodology under evaluation. Finally, the points of *Additional Features* and *Documentation* incorporate other issues of interest like the quality of the documentation provided or the extensions defined for mobility, ontologies, etc.

2.1 Development Process

In this section, aspects related to the construction of a MAS (how it is built) will be evaluated. These aspects can be summarised in the general questions: "which are the stages proposed by the methodology?" and "what kind of activities must be accomplished in each stage?". In particular, the following aspects are considered:

- *Application domain.* It evaluates if the methodology is supposed to address systems from any domain or if it is domain oriented.
- *Application areas.* This issue collects the information presented by authors about the application of the methodology in different areas. A list of the areas is showed, underlining for each one if it is only mentioned by the authors or if the results of the methodology in this particular area are well documented (i.e. papers, manuals, books, ...).
- *Open systems.* It tries to evaluate if the methodology is intended or not for open systems (i.e. it allows for dynamic addition or removal of agents, or their characteristics, while MAS is running).
- *Kind of lifecycle.* It describes what kind of lifecycle model is applied in the development process: cascade, spiral, incremental, etc.
- *Process stages.* This is the main point of the evaluation; here it is described how the lifecycle is covered by the methodology. The stages of the methodology and what kind of activities must be done in each stage are presented. For addressing activities, classical software engineering taxonomy has been used: Requirements, Analysis, Design, Implementation and Testing. Requirements activities are related with the initial specification of client necessities. Analysis includes any activity related to the description of WHAT the system must do. Design activities are oriented to define HOW the system does what it must do, and how the final model of the system is obtained. Implementation activities are those related with code generation. And, finally, Testing activities try to check and validate the system under construction. Each of these activities is qualified with the term:

- "Focused". If the methodology is fully oriented to that kind of activities in the stage under consideration.
- "Not Focused". If the stage does not consider activities of this kind.
- "Partially Focused", in other case.

In addition, a meaningful aspect to be considered is the implication of the user in the development process, and when this implication takes place. User implication is essential, in our opinion, in order to obtain the desired final product as it is pointed by different new software engineering methodologies [1]. This evaluation takes into account if the methodology relies on user in each stage, and the term unknown is used when this information is not clear or available.

– *Supporting methodological and code generation tools.* Another important issue is the availability of methodology supporting tools. The existence of tools is evaluated in relation to the activities in which the tool is used. The evaluation simply indicates if there are or not tools for each kind of activities, and how the activity is supported by the tool.

2.2 Model View

Model View section tries to evaluate the diagrams and techniques proposed by the methodology for modelling the system. The relationships between diagrams or between different views of the system, the model evolution through lifecycle and other aspects are taken into account.

– *Concepts and representation.* When modelling the system many concepts must be used and each of them must have a representation. This section presents a list of the concepts used and where (diagram or model) they are represented. This list is afterwards detailed for each stage of the methodology, relating the different diagrams and the concepts managed for each of them. It is though that a consistent definition of concepts through different models and a uniform representation will be a valuable feature of the considered methodology. The final point (human interaction) evaluates if human interactions with the system are specifically considered in the methodology.
– *Relationships between models.* The methodology may provide a way of deriving a new view of the model from an existent one, or may define rules for guaranteeing consistence and correctness when two views are related. In this point the gap between some relevant relations is underlined.
– *Deliverables.* In this point, the coverage provided by the different model views with respect to the whole lifecycle of the system is evaluated. By studying what are the deliverables provided by the methodology at different steps, it could be decided whether the model fulfils all the proposed stages or some of them are weakly covered. In this latest case, the models must be complemented with prototypes or with text descriptions wrote in formal, structured or natural language.

2.3 Agent

In this section, the way in which agent's characteristics and features are defined by the methodology is taken into consideration. A fundamental point in defining MAS must be the description of the individual components of the system, that is, agents. The features considered are:

- *Concept.* The concept of agent proposed by the methodology is relevant because it introduces the conceptual framework of development. Many characteristics of individual agents may be introduced in the definition, like if the methodology is intended for a particular agent architecture (by example BDI) or if only particular kinds of agents are to be defined, etc.
- *Agent attributes.* This point is in close relation with the concept of agent and handles the intrinsic characteristics that the methodology uses: autonomy, sociability, reactivity, proactivity, intelligence or others.

In addition, the goals achieved by MAS must be more than the sum of the individual goals of the agents implied due to agents interaction and co-operation. This aspect is studied and evaluated considering:

- *Communication Types.* In this point interaction among agents is evaluated at a high level of abstraction. Methodology may allow only communication from agent to agent (A-A) (heterogeneous or homogeneous) or other kinds like human-agent (A-H) or agent-other software systems (A-O).
- *Communication Protocols.* Communication may follow well-known protocols that can be predefined in the methodology.
- *Co-operation.* This criterion shows the kinds of interaction among agents which can be defined using the methodology: negotiation, delegation of tasks, etc.
- *Agent organization.* Agents may have an internal structure with influence in the communication. This organization may be hierarchical, peer to peer, etc.

2.4 Additional Features Modelling

This point will address the extensions that normally methodologies propose to deal with important aspects of MAS, like:

- Ontological aspects
- Mobility features
- Other additional features

2.5 Documentation

An important aspect when dealing with new proposals is how they are documented. To evaluate this point the following characteristics are addressed:

– *Available documentation.* The documentation (papers, web site, etc.) provided by authors is evaluated and qualified with: "Good", "Sufficient" or "Poor"; where, "Poor" means that the provided documentation is not enough to make the methodology completely understandable.
– *Study Cases presented.* It evaluates another important aspect when documenting a methodology, that is, the provided examples. It is evaluated to "Trivial" (if there is only little examples for showing particular aspects), "Partial" (in the case that examples are more complete but do not cover the whole lifecycle proposed) or "Complete" (if they cover the whole lifecycle).

3 A Case Study

Defining a framework for evaluation, as the one presented in the previous section, is useless unless it is used for comparing different methodologies. In a previous work [5] we have used the framework to evaluate the most relevant AOSEM which extend Object Oriented Methodologies, particularly Gaia, MaSE, Tropos and Message-INGENIAS. Here, the study will be centred in two different kinds of AOSEM. One of them pertaining to the class of AOSEM which are a evolution of Object Oriented Methodologies (MaSE) and the other corresponding to the field of Knowledge Engineering (MAS-CommonKADS). In this way, the capacity of the framework for being utilized for the evaluation of different kinds of Agent Methodologies is showed.

3.1 Development Process

Multiagent Systems Engineering (MaSE) [7] is a general purpose methodology for developing heterogeneous multiagent systems. MaSE covers the complete lifecycle of the system and uses a number of graphically based models, that are supported with a software engineering tool (AgentTool [6]). AgentTool also supports automatic verification of inter-agent communications and code generation. The lifecycle of MaSE is iterative. It is pretended that the analyst or designer moves among steps and phases freely such that, with each successive step, additional detail is added and, eventually, a complete and consistent system design is obtained. The purpose of the Analysis phase is to produce a set of roles whose tasks describe what the system has to do to meet its overall requirements. The goal of the Design phase is to define the overall system organization by transforming the roles and tasks defined during analysis into agent types and conversations.

MAS-CommonKADS is a multiagent methodology which extends a previous one: CommonKADS (oriented to Knowledge Based Systems development). This extension adds techniques from Object Oriented Methodologies and Responsibility Driving Design. For describing the agents protocols, methods from protocol engineering such as Specification and Description Language (SDL) and Message Sequence Charts (MSC) are used. The software process of the methodology combines the risk driven and the component based approaches in an iterative and incremental way.

Each iteration of MAS-CommonKADS is organized in the following phases: Conceptualization, Analysis, Design, Coding and Testing, Integration and Global Testing, and Deployment. During the Conceptualization phase, an elicitation task to obtain a preliminary description of the problem is carried out. The Analysis phase is oriented to get the requirements specification of the MAS through the development of different models in a risk-driven way. Analysis phase follows five steps: *Agent, Task, Coordination, Knowledge and Organization modelling.* To get the design model, the Design Phase consists of: *Agent network design, Agent design and Platform design.* The rest of the phases are expected to be approached as usually in Software Engineering Methodologies and are not explained in detail.

3.2 Model View

MaSE is based in the concept of goal. The models of analysis (*Goal Hierarchy Diagram, Use Cases, Sequence Diagram, Role Model, Concurrent Task Diagram*) capture the required organization, actions, and interactions among tasks, using the concepts of role and tasks. A role describes an entity that performs some function within the system. A task is a description of what the system has to do to fulfill the goals associated with a particular role. On the other hand, agent design captures roles and tasks. Roles are played by agent classes, conversations capture interaction and Actions are captured via methods. The diagrams used are *Agent Class Diagram, Communication Class Diagram, Agent Architecture Diagram and Deployment Diagram.* The relationship between models is well defined and the information provided is sufficient. MaSE does not explicitly model human computer interaction; it suggests that a specific role would be created to encapsulate the user interface.

The application of MAS-CommonKADS methodology is based on the development of seven different models. Each model shows the entities to be defined (called "constituents") and the relationships among them. Each model has a textual template to define and describe each "constituent". The models, diagrams and notation used are:

- *Agent model.* It specifies the agent characteristics using use cases, CRC (Class Responsibility Collaboration) cards, and the agent textual template.
- *Task model.* It describes the tasks that the agents can carry out: goals, decompositions, ingredients and problem-solving methods, etc. Although there is not a mandatory notation, the description of a task must include its name, a short description, input and output ingredients, task structure, ...
- *Expertise model.* This model describes the knowledge needed by the agents to achieve their goals. It covers the development of the application knowledge (consisting of domain knowledge, inference knowledge and task knowledge) and problem solving knowledge. In order to get the desired model the Conceptual Modelling Language (CML) or the Object Model Diagrams, the inference structures and the task-method inference decomposition structures are used.

- *Organization model*. The organizational environment of the MAS and the social organization of the agent society are defined. The graphical notation of these models is based on the Object Model, adding a special symbol to distinguish agents from objects. The aggregation symbol is used for expressing agent groups.
- *Coordination model*. It models the conversations between agents: "their interactions, protocols and required capabilities". It has two milestones: (1) definition of the communication channels and building of a prototype; (2) analysis of the interactions and determination of complex interactions (with coordination protocols). The model is presented using MSC notation, flow diagrams, state transition diagrams of SDL interactions, cooperation protocols library and HMSC (High level Message Sequence Charts) to describe and define new protocols.
- *Design model*. It consists of three submodels: "network design" for designing the relevant aspects of the agent network infrastructure; "agent design" for dividing or composing the agents of the analysis; and "platform design" for selecting the agent development platform for each agent architecture.

MAS-CommonKADS suggest to include one interface agent for each human interaction detected in the application.

3.3 Agent

With respect to the concept of agent, for MaSE, agents are a specialization of objects, which coordinate with each other via conversations and act proactively to accomplish individual and system-wide goals. Moreover, agents can be viewed as convenient abstractions, which may or may not possess intelligence. Designers have the choice of designing their own architecture or using predefined architectures such as BDI. Likewise, a designer may use predefined components or develop them from scratch.

MAS-CommonKADS is intended to be used to model any kind of agents; with independence of the concept of agent considered. Moreover, the methodology covers a strong concept in which the agent is an entity with mental state [16] and also a weak one, that views agents as entities that could interchange messages using a communication agent language.

3.4 Additional Features Modelling

Some extensions of MaSE have been proposed for integrating Ontologies [8] and for designing and specifying Mobility within the MaSE Methodology [14].

MAS-CommonKADS does not propose to include mobility aspects, but ontological aspects have been taken into account when modelling agents and their relationships.

3.5 Documentation

The principal MaSE available documentation are research papers. One important restriction of MaSE is that it does not provide a complete study case.

For MAS-CommonKADS only research papers and a Phd. Thesis are available. The travel agency case study (not fully developped) is the only application that is documented enough.

4 Conclusions

In this work, a new framework for the evaluation of AOSEM has been presented. This framework has proved its utility in the evaluation of different AOSEM, even if these methodologies are of very different nature. Particularly, the framework has been successfully applied for comparing a methodology which evolved from Object Orientation with other one that came from the Knowledge Engineering field.

In relation to the conclusions obtained from the comparison, it must be underlined that the current AOSEM are not really suitable for their application in industrial software projects, because most of them are still experimental projects. In addition, the studied proposals cover only a portion of the lifecycle or are focused exclusively on specific aspects of the development process, forgetting the rest of the activities that must be carried out.

Another important issue that must be highlighted is the lack of quality supporting tools for aiding in the system construction. Although some tools are proposed, the support provided is partial and must be further developed; particularly, the tools must be extended to handle all phases and steps, including code generation. Even more, the methodologies are presented as independent from any tool for agent development but some of them are oriented to a particular development tool.

Finally, we consider a very important restriction in the methodologies studied that they are only suitable to model close systems. Nowadays, the increasing importance of Internet applications has led to open dynamic systems. So, new proposals for been of real utility must incorporate these characteristics to systems.

References

1. K. Beck. *eXtreme Programming Explained.* Addison-Wesley, 1999.
2. P. Ciancarini and M. Wooldridge, editors. *Agent-Oriented Software Engineering. First International Workshop, AOSE 2000*, volume 1957 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.
3. P. Cuesta, A. Gómez, J.C. González, and F. J. Rodríguez. The MESMA approach for AOSE. In *Fourth Iberoamerican Workshop on Multi-Agent Systems (Iberagents'2002), a workshop of IBERAMIA'2002, the VIII Iberoamerican Conference on Artificial Intelligence*, 2002.

4. P. Cuesta, A. Gómez, J.C. González, and F. J. Rodríguez. The MESMA methodology for agent-oriented software engineering. In *Proceedings of First International Workshop on Practical Applications of Agents and Multiagent Systems (IWPAAMS'2002)*, pages 87–98, 2002.
5. P. Cuesta, A. Gómez, J.C. González, and F. J. Rodríguez. A framework for evaluation of agent oriented methodologies. Technical Report IT1/2003, Departamento de Informática (Universidade de Vigo), 2003.
6. S. A. DeLoach and M. Wood. Developing multiagent systems with agenttool. In Y. Lesperance C. Castelfranchi, editor, *Intelligent Agents VII. Agent Theories Architectures and Languages, 7th International Workshop (ATAL 2000)*, volume 1986 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.
7. S. A. DeLoach, M. F. Wood, and Clint H. Sparkman. Multiagent systems engineering. *The International Journal of Software Engineering and Knowledge Engineering*, 11(3), 2001.
8. J. Dileo, T. Jacobs, and S. DeLoach. Integrating ontologies into multiagent systems engineering. In *Proceedings of Agent Oriented Information Systems (AOIS'2002)*, 2002.
9. C. A. Iglesias. *Definición de una Metodología para el Desarrollo de Sistemas Multiagente*. PhD thesis, Departamento de Ingeniería de Sistemas Telemáticos. Universidad Politécnica de Madrid, 1998.
10. C. A. Iglesias, M. Garijo, and J. C. González. A survey of agent-oriented methodologies. In J. Müller, M. P. Singh, and A. S. Rao, editors, *Intelligent Agents V. Agents Theories, Architectures, and Languages*, volume 1555 of *Lecture Notes in Computer Science*, pages 317–330. Springer-Verlag, 1999.
11. C. A. Iglesias, M. Garijo, J. C. González, and J. Velasco. Analysis and design of multiagent systems using MAS-CommonKADS. In M. P. Singh, A. S. Rao, and M. J. Wooldridge, editors, *Intelligent Agents IV. Agents Theories, Architectures, and Languages*, volume 1365 of *Lecture Notes in Computer Science*, pages 312–326. Springer-Verlag, 1998.
12. N. R. Jennings. Agent-based computing: Promise and perils. In Dean Thomas, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99-Vol2)*, pages 1429–1436. Morgan Kaufmann Publishers, 1999.
13. A. Sabas, S. Delisle, and M. Badri. A comparative analysis of multiagent system development methodologies: Towards a unified approach. In *Third International Symposium From Agent Theory to Agent Implementation (AT2AI-3), Volume 2*, pages 599–604, 2002.
14. A. Self and S. DeLoach. Designing and specifying mobility within the multiagent systems engineering methodology. In *Special Track on Agents, Interactions, Mobility, and Systems (AIMS) at The 18th ACM Symposium on Applied Computing (SAC 2003)*, 2003.
15. O. Shehory and A. Sturm. Evaluation of modeling techniques for agent-based systems. In *Agents-01*, pages 624–631, 2001.
16. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.
17. K. P. Sycara. Multiagent systems. *The AI Magazine*, 19(2):79–92, 1998.
18. G. Weiss. *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.
19. M. Wooldridge, N. R. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.